Subject: Re: Inverse palette conversion algorithm...
Posted by mr_ped on Fri, 07 Apr 2006 15:30:47 GMT
View Forum Message <> Reply to Message

correction from my previous post
SAR vs SHR conflict, SAL vs SHL is no problem, as both instructions give the same result even with negative numbers.

Now ... as I really enjoyed the task you were trying to accomplish, and I love graphics effects and speed optimalisations, I choosed to spend this lovely day polishing my skills and to give you another code to think about...
(well, just half of the day spend and finally I did use something from U++ (containers) .. so far I always checked TheIDE only on some simple "stdio.h" console application, so now I tried to do something more complex with it)

Beat this :

```
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,
 RASTER_MAP_MAX_SHIFT = 2,

 RASTER_MAP_R_ADD = (1<<RASTER_SHIFT_R),
 RASTER_MAP_G_ADD = (1<<RASTER_SHIFT_G),
 RASTER_MAP_B_ADD = (1<<RASTER_SHIFT_B),
 RASTER_MAP_MAX_DIST = 3*((RASTER_MAP_MAX-1)*(RASTER_MAP_MAX-1))
};

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

struct sCubePoint : Moveable<sCubePoint> {
 RGBA mycol;
 byte index;
```

```
};

struct sPalCv2 {
 PaletteCv&  cv_pal;
 const RGBA *palette;
 int         ncolors;

 void AddPoint(byte r, byte g, byte b, byte idx, Vector<sCubePoint> *feed);

 sPalCv2(const RGBA *palette, int ncolors, PaletteCv& cv_pal);
};

void sPalCv2::AddPoint(byte r, byte g, byte b, byte idx, Vector<sCubePoint> *feed)
{
 sCubePoint pt;
 int dr = (int(palette[idx].r)-int(r))>>RASTER_MAP_MAX_SHIFT;
 int dg = (int(palette[idx].g)-int(g))>>RASTER_MAP_MAX_SHIFT;
 int db = (int(palette[idx].b)-int(b))>>RASTER_MAP_MAX_SHIFT;
 int dist = dr*dr + dg*dg + db*db;
 pt.mycol.r = r;
 pt.mycol.g = g;
 pt.mycol.b = b;
 pt.index = idx;
 ASSERT(dist <= RASTER_MAP_MAX_DIST);
 feed[dist].Add(pt);
}

sPalCv2::sPalCv2(const RGBA *palette, int ncolors, PaletteCv& cv_pal)
: cv_pal(cv_pal), ncolors(ncolors), palette(palette)
{
 int ii, jj;
 sCubePoint cubpt;
 Vector<sCubePoint> feed_me[RASTER_MAP_MAX_DIST+1];
 byte filled[RASTER_MAP_R * RASTER_MAP_G * RASTER_MAP_B];
 ZeroArray(filled);

 ii = ncolors;
 while (ii--) {
  cubpt.index = ii;
  cubpt.mycol = palette[ii];
  feed_me[0].Add(cubpt);
 }
 for (ii = 0; ii <= RASTER_MAP_MAX_DIST; ++ii) {
  while ( !feed_me[ii].IsEmpty() ) {
   cubpt = feed_me[ii].Pop();
   jj = cv_pal.GetIndex(cubpt.mycol);
   if (filled[jj] != 0) continue;
   filled[jj] = 1;
```

```
   cv_pal.cv[jj] = cubpt.index;
   if ( int(cubpt.mycol.r)+RASTER_MAP_R_ADD <= 255 )
    AddPoint(cubpt.mycol.r+RASTER_MAP_R_ADD, cubpt.mycol.g, cubpt.mycol.b, cubpt.index,
feed_me);
   if ( int(cubpt.mycol.r)-RASTER_MAP_R_ADD >= 0 )
    AddPoint(cubpt.mycol.r-RASTER_MAP_R_ADD, cubpt.mycol.g, cubpt.mycol.b, cubpt.index,
feed_me);
   if ( int(cubpt.mycol.g)+RASTER_MAP_G_ADD <= 255 )
    AddPoint(cubpt.mycol.r, cubpt.mycol.g+RASTER_MAP_G_ADD, cubpt.mycol.b, cubpt.index,
feed_me);
   if ( int(cubpt.mycol.g)-RASTER_MAP_G_ADD >= 0 )
    AddPoint(cubpt.mycol.r, cubpt.mycol.g-RASTER_MAP_G_ADD, cubpt.mycol.b, cubpt.index,
feed_me);
   if ( int(cubpt.mycol.b)+RASTER_MAP_B_ADD <= 255 )
    AddPoint(cubpt.mycol.r, cubpt.mycol.g, cubpt.mycol.b+RASTER_MAP_B_ADD, cubpt.index,
feed_me);
   if ( int(cubpt.mycol.b)-RASTER_MAP_B_ADD >= 0 )
    AddPoint(cubpt.mycol.r, cubpt.mycol.g, cubpt.mycol.b-RASTER_MAP_B_ADD, cubpt.index,
feed_me);
  }
 }
}
```

My simple benchmark shows it's almost 2 times faster.
It's probably a bit more memory hungry than your original code, but I don't think it's too greedy. (I didn't benchmark that, I didn't do any serious profiling under Win32 for long time, so I'm not sure what tools to use for that)

It does probably not give absolutely same result, but only marginaly different. (can you test it? I did some quick test to see if it looks functional, but didn't have any real-life app to see check)

Also it does confuse Assist a bit.
(try "feed_me[23]." ... and there we go, Assist has no idea it's Vector, instead it does thing it's sCubePoint)

I'm running on self compiled (MS VCC toolkit) upp-src-604-dev1.zip

And debugger is not really helpul with his inability to show feed[ii] data. (he keeps displaying feed[0] no matter what ii contains)

(At least it forced me to think harder why it does crash, and took me just 2min to figure out my mistake)