Una manera rápida de trabajar con datos de pixeles de bitmaps en Windows sin usar los métodos lentos GetPixel y SetPixel

Jan Kurbanaliev, 5 Jun 2010

When GetPixel() and SetPixel() are very slow, try this solution.

- **Artículo original**

Introduction

This article shows a fast way to view and change pixel color data (Windows bitmaps) without using the `GetPixel` and `SetPixel` methods.

Background

I spent a couple of months in trouble not knowing a fast way to edit images with C++ (I was using the Windows API functions `GetPixel` and `SetPixel`). I lost a couple of clients. I then spent a couple of weeks browsing through forums trying to find answers. But eventually, I got through this by myself and learned this easy pointer arithmetic method.

Using the Code

We use a pointer to bitmap bits:
`bitmapzor.GetBits();`.

Then, we use a pointer offset to get to another line in the bitmap pixel colors array:
`bitmapzor.GetPitch();`.

Points of Interest

A normal bitmap is a WxH pixel table with a header. Each value in the pixel table is a 24 bit integer or 3x`CHAR`s. To access these values, we need to know the starting point of the pointer and the offset. Normal bitmaps have a negative offset; e.g., if the pointer to some line of the color table is a `byteptr`, then the pointer to the next line is `byteptr-3*W`. But sometimes, bitmaps are upside-down oriented and the pointer to the next line is `byteptr+3*W`. Anyway, the offset negative or positive is always given by the `GetPitch()` method.

License

The complete code with comments is listed here:

```
// FastImageManipulations.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"

int _tmain(int argc, _TCHAR* argv[]) {

    // Okay Now you need a bitmap to work with. Find any bmp file on your PC and
    // Put it to C:\\1.bmp location. You can enter any name, just alter the file
    // path below:
    CImage bitmapzor;

    bitmapzor.Load(_T("C:\\1.bmp"));

    // I use ATL CImage class because it is easy to use. Low number of code lines
    // You can use HBITMAP, HGDIOBJ of GDIplus and other methods,
    // They work the same way and at the same speed, but take tons of code lines
    // For instance you can use HBITMAP bitmapzor = (HBITMAP)ImageLoad(...)
    // And then you are to use GDI and API functions

    // Let us see the difference between standard API and pointer way:
    printf ("Now we use ATL (Api) GetPixel and SetPixel functions to "
             "enforce (for example) all green pixels by 30%\n please "
             "press any key to start and start counting minutes :)");
    getchar();
    COLORREF PixColor=0; // This is a color data

    int R=0,G=0,B=0; // These are the channel values

    // First we shall for instance allter all pixel colors to green using standard way
    for (int i=0; i<bitmapzor.GetWidth(); i++) //along line
        for (int j=0; j<bitmapzor.GetHeight(); j++) //new line
            {
            // This evil slow function extracts 24bit pixel color as BGR value
            PixColor = bitmapzor.GetPixel(i,j);

            R=GetRValue(PixColor); // This macro extracts red channel value
            G=GetGValue(PixColor); // This macro extracts green channel value
            B=GetBValue(PixColor); // This extracts blue pixel color
            G=(int)((float)G*1.3); // There we enforce green color by 30%.
            if (G>255) G=255; // There we ensure G is within 0-255 range
            // We can assemble the BGR 24 bit number that way
            // PixColor = B*0x10000+G*0x100+R;
            PixColor = RGB(R,G,B);//Or we can use this function

            // Now we save a new "green" pixel value back to bitmap
            bitmapzor.SetPixel(i,j,PixColor);
         }

    // Now we need to save this result to a HD:
    LPCTSTR filename=_T("C:\\GetPixel.bmp");
    // you can change the location of this file
    bitmapzor.Save(filename);

    printf ("Done, file is saved to: %s\n Please press "
             "any key for another method demo", filename);
    getchar();
```

```
    // Okay now another way with the pointer arithmetics:
    printf ("Pointer arithmetics demo (without GetPixel and SetPixel functions): \n");
    bitmapzor.Destroy(); // Unload the changed bitmap from memory
    bitmapzor.Load("C:\\1.bmp");

    // this is a pointer to the exact bitmap pixel color array location
    BYTE* byteptr = (BYTE*)bitmapzor.GetBits();


    // You can use other functions for HBITMAP and HGDIOBJ methods to find out the
    // container for bitmap color table.
    // For instance when you use GDI and HBITMAPS, you should use
    // BYTE* byteptr = (BYTE*)(HBITMAPINFOHEADER + HBITMAPINFOHEADER->biSize);
    // simple pointer arithmetics

    int pitch = bitmapzor.GetPitch(); // This is a pointer offset to get new line of the bitmap

    for (int i=0; i<bitmapzor.GetWidth();i++)
         for (int j=0; j<bitmapzor.GetHeight();j++) {
              // pointer arithmetics to find (i,j) pixel colors:
               R = *(byteptr+pitch*j+3*i);
               G = *(byteptr+pitch*j+3*i+1);
               B = *(byteptr+pitch*j+3*i+2);

              // allter pixel G color:
               G = (int)((float)G*1.3);
               if (G>255) G=255;

              // write down the new G-Color
                *(byteptr+pitch*j+3*i+1) = G;
          }

    // Save the bitmap:
    LPCTSTR filename2 = _T("C:\\ptrarithm.bmp"); //you can use any other file name
    bitmapzor.Save(filename2);

    printf ("Done, file is saved to: %s\n Please press any key to exit program", filename2);
    getchar();

    bitmapzor.Destroy(); //destroy the bitmap
    return 0;
}
```