ImageStone - A Powerful C++ Class Library for Image Manipulation

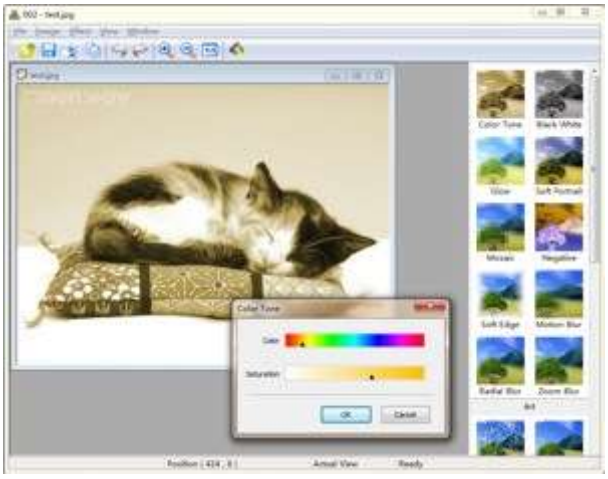
crazybit

6 Dec 2011 Zlib

Rate me: 4.80/5 (249 votes)

An article on a library for image manipulation

- Download full source with examples - 434.42 KB
- Download CHM document - 367.54 KB
- Download executable example - 397.4 KB



Introduction

`ImageStone` is a powerful C++ class library for image manipulation. Its features include load, save, display, transformation, and nearly 100 special image effects. It can be used cross platform (includes Windows, Linux, Mac), and especially under Windows, it can be used as a DIB wrapper class.

How to Use

Similar to STL and Boost library, `ImageStone` is a header-only C++ library, it consists entirely of header files, so you can begin to use it just add `#include "ImageStone.h"` at the beginning of your source code, But in order to take full advantage of operating system features, there are some differences between various platforms.

On Windows

Only need to add `#include "ImageStone.h"` in your project, normally to be added at the end of *StdAfx.h* file.

`ImageStone` uses raw Win32 GDI+ API to implement image `load`/`save` function, so it has high efficiency and small final file size.

On Non-Windows (Unix, Linux, Mac...)

`ImageStone` uses `FreeImage` lib to implement image load/save function.

1. Get `FreeImage` from http://sourceforge.net/projects/freeimage/, then compile it.
2. Include `FreeImage` header before include `ImageStone`.

```C++
#include "FreeImage.h"
#include "ImageStone.h"
```

3. Now you can use `ImageStone`.

Advanced Image Effect

In order to reduce compile time, a lot of rarely-used image effects (emboss, twist, ripple...) are not to be included by default, you can use them by the following way:

```C++
#define IMAGESTONE_USE_EXT_EFFECT
#include "ImageStone.h"
```

... Load Image from File, Memory or Resource, Save Image to File

```C++
FCObjImage   img ;

// from file
img.Load (L"test.jpg") ;

// from memory
img.Load (pMemory, nSize, IMG_JPG) ;

// from resource under windows
img.LoadBitmap (ID_BITMAP) ;
img.LoadResource (ID_JPG_FILE, _T("JPG"), IMG_JPG) ;

// save as jpg with quality 90 ( max is 100 )
img.Save (L"test.jpg", 90) ;
```

In order to better support for multiple languages, `ImageStone` only supports wide `char` filename, so you need to call a conversion function for ANSI filename.

```C++
std::wstring A_to_W (const char* p)
{
    std::wstring   ws ;
    if (p)
```

```
    {
        setlocale (LC_CTYPE, "") ;

        std::vector<wchar_t>  buf (strlen(p)+1, 0) ;
        mbstowcs (&buf[0], p, buf.size()-1) ;
        ws = &buf[0] ;
    }
    return ws ;
}

FCObjImage   img ;
img.Load (A_to_W("test.jpg").c_str()) ;

// under windows, you can use _bstr_t to implement conversion
img.Load (_bstr_t("test.jpg")) ;
```

... Load multi-page GIF

C++

```
std::vector<FCObjImage*>   frame_list ;
FCImageProperty   prop ;
FCObjImage::Load (L"test.gif", frame_list, &prop) ;

// now frame_list store all frames of Gif image
// member m_frame_delay of prop store delay time between two frame
prop.m_frame_delay ;

// caller must delete all frames when them no longer be used
for (size_t i=0 ; i < frame_list.size() ; i++)
{
    delete frame_list[i] ;
}
```

... Apply Image Effect

C++

```
FCObjImage   img ;
img.Load (L"test.jpg") ;

img.Stretch (200, 200) ;
img.Stretch_Smooth (500, 500) ;

FCEffectBlur_Gauss   c1 (16, true) ;
img.ApplyEffect (c1) ;

FCEffectSplash   c2 (20) ;
img.ApplyEffect (c2) ;

FCEffectMosaic   c3 (20) ;
img.ApplyEffect (c3) ;

FCEffectBrightnessContrast   c4 (30, 0) ;
img.ApplyEffect (c4) ;
```

You can monitor the progress of image processing by setting an observer:

C++

```
// this monitor will stop image processing when user press ESC key
class CMyProgressMonitor : public FCProgressObserver
{
```

```
public:
    virtual bool OnProgressUpdate (int nFinishPercentage)
    {
        MSG   msg ;
        if (PeekMessage (&msg, NULL, WM_KEYDOWN, WM_KEYDOWN, PM_REMOVE) &&
            (msg.wParam == VK_ESCAPE))
        {
            return false ;
        }
        return true ;
    }
};

// apply effect with progress monitor
FCObjImage   img ;
img.Load (L"test.jpg") ;

CMyProgressMonitor   aProgress ;

FCEffectMosaic   c1 (20) ;
img.ApplyEffect (c1, &aProgress) ;
```

... Custom Image Effect

C++

```
// create our effect : split RGB channel of image
class CEffectSplitChannel : public FCImageEffect
{
public:
    FCObjImage   m_red, m_green, m_blue ;

private:
    virtual bool IsSupport (const FCObjImage& img)
    {
        return img.IsValidImage() && (img.ColorBits() >= 24) ;
    }

    virtual void OnBeforeProcess (FCObjImage& img)
    {
        int   w = img.Width() ;
        int   h = img.Height() ;
        m_red.Create (w, h, 24) ;
        m_green.Create (w, h, 24) ;
        m_blue.Create (w, h, 24) ;
    }

    virtual void ProcessPixel (FCObjImage& img, int x, int y, BYTE* pPixel)
    {
        PCL_R(m_red.GetBits(x,y)) = PCL_R(pPixel) ;
        PCL_G(m_green.GetBits(x,y)) = PCL_G(pPixel) ;
        PCL_B(m_blue.GetBits(x,y)) = PCL_B(pPixel) ;
    }
};

// test, save RGB channel to image
void Test()
{
    FCObjImage   img ;
    img.Load (L"test.jpg") ;

    CEffectSplitChannel   cmd ;
    img.ApplyEffect (cmd) ;

    cmd.m_red.Save (L"red.jpg") ;
    cmd.m_green.Save (L"green.jpg") ;
```

```
    cmd.m_blue.Save (L"blue.jpg") ;
}
```

... Read and Modify EXIF (Win Only)

C++

```
FCObjImage   img ;
FCImageProperty   prop ;

img.Load (L"d:\\test.jpg", &prop) ;

// modify manufacturer of the equipment
prop.m_EquipMake = "PhoXo" ;

// modify the date of photograph
prop.m_ExifDTOrig = "2011:11:26 09:26:00" ;

std::vector<std::string>   & ls = prop.m_other_gdiplus_prop ;
for (size_t i=0 ; i < ls.size() ; i++)
{
    Gdiplus::PropertyItem   it = FCImageCodec_Gdiplus::ReferencePropertyBuffer (ls[i]) ;

    // modify ISO speed to 400
    if (it.id == PropertyTagExifISOSpeed)
    {
        short   nISO = 400 ;
        ls[i] = FCImageCodec_Gdiplus::CreatePropertyBuffer(it.id, it.type, 2, &nISO) ;
    }
}

// save image file with modified EXIF
img.Save (L"d:\\test.jpg", prop) ;
```

... Draw Image (Win Only)

C++

```
FCObjImage   img ;
img.Load (L"c:\\test.png") ;

// for 32bpp image, method Draw use Win32 API AlphaBlend to draw,
// so we need pre-multiply alpha before draw
// for <= 24bpp image, method Draw use Win32 API BitBlt to draw
if (img.ColorBits() == 32)
{
    img.ApplyEffect (FCEffectPremultipleAlpha()) ;
}

// Draw whole image (no stretch) on point(0,0) of hdc
img.Draw (hdc, 0, 0) ;

// Draw whole image on specified region of hdc.
RECT   rcOnDC = {100, 100, 200, 200} ;
img.Draw (hdc, rcOnDC) ;

// Draw region of image on specified region of hdc.
RECT   rcOnImage = {20, 20, 50, 50} ;
img.Draw (hdc, rcOnDC, &rcOnImage) ;

// Draw whole image in window.
SIZE   img_size = {img.Width(), img.Height()} ;
RECT   rcWindow = {0, 0, 500, 500} ;
```

```
RECT   rc = FCObjImage::CalcFitWindowSize(img_size, rcWindow) ;
img.Draw (hdc, rc) ;
```

Although BitBlt is faster than AlphaBlend, it will destroy the alpha channel of destination. Some features such as: layered window, DWM need alpha channel, so these cases we have to convert image to 32bpp, using AlphaBlend to draw.

... Convert between HBITMAP and Gdiplus::Bitmap (Win Only)

C++

```
FCObjImage   img ;

// FCObjImage can be converted to HBITMAP automatically, and can be selected into HDC
SelectObject (hdc, img) ;

// create image based on HBITMAP
img.FromHBITMAP (hBitmap) ;

// create Gdiplus::Bitmap, caller must delete returned bitmap
Gdiplus::Bitmap   * pBmp = img.CreateBitmap() ;
if (pBmp)
    delete pBmp ;

// create image based on Gdiplus::Bitmap
Gdiplus::Bitmap   gpBmp (L"c:\\test.jpg") ;
img.FromBitmap (gpBmp) ;
```

... Add Text on Image (Win Only)

C++

```
void DrawText (HDC dc)
{
    // GDI draw text
    SetTextColor (dc, RGB(0,0,255)) ;
    TextOut (dc, 0, 0, _T("PhoXo"), 5) ;

    // GDI+ draw text
    Gdiplus::Graphics   g(dc) ;
    g.SetSmoothingMode (Gdiplus::SmoothingModeAntiAlias) ;
    g.SetInterpolationMode (Gdiplus::InterpolationModeHighQualityBicubic) ;

    Gdiplus::FontFamily   ffami (L"Arial") ;
    Gdiplus::StringFormat fmt ;

    Gdiplus::GraphicsPath   str_path ;
    str_path.AddString (L"PhoXo", -1, &ffami,
    Gdiplus::FontStyleBold, 48, Gdiplus::Point(20,20), &fmt) ;

    Gdiplus::Pen   gp (Gdiplus::Color(0,0,160), 8) ;
    gp.SetLineJoin (Gdiplus::LineJoinRound) ;

    Gdiplus::Rect    rc (20, 20, 30, 60) ;
    Gdiplus::Color   cStart (255,255,255) ;
    Gdiplus::Color   cEnd (0,128,255) ;
    Gdiplus::LinearGradientBrush  gb (rc, cStart, cEnd,
    Gdiplus::LinearGradientModeVertical) ;

    g.DrawPath (&gp, &str_path) ;
```

```
    g.FillPath (&gb, &str_path) ;
}

void Test()
{
    FCObjImage   img ;
    img.Create (300, 200, 24) ;

    // fill back grid
    img.ApplyEffect (FCEffectFillGrid(FCColor(192,192,192), FCColor(255,255,255), 16)) ;

    DrawText (FCImageDrawDC(img)) ;

    img.Save (L"d:\\test.png") ;
}
```

... Using in DLL (Win Only)

In most cases, there is no difference on using between in DLL and in client:

C++

```
extern "C" __declspec(dllexport) HBITMAP LoadFileToDDB (LPCWSTR sFilename)
{
    FCObjImage   img ;
    img.Load (sFilename) ;

    HDC      screen_dc = GetDC(NULL) ;
    HBITMAP  hBmp = CreateCompatibleBitmap(screen_dc, img.Width(), img.Height()) ;
    ReleaseDC (NULL, screen_dc) ;

    img.Draw (FCImageDrawDC(hBmp), 0, 0) ;
    return hBmp ;
}
```

But, if you want to call `Load` or `Save` method of `FCObjImage` in `DllMain` or constructor of global object in DLL, you need use `FCAutoInitGDIPlus` to init GDI+ module in your clients before load DLL, and disable auto GDI+ init of `ImageStone` in DLL.

C++

```
// call Load in global object.
class CGlobalObj
{
public:
    CGlobalObj()
    {
        FCImageCodec_Gdiplus::AUTO_INIT_GDIPLUS() = FALSE ;

        FCObjImage   img ;
        img.Load (L"c:\\test.jpg") ;
    }
};

CGlobalObj   g_obj ;

// call Load in DllMain.
BOOL APIENTRY DllMain (HMODULE hModule, DWORD, LPVOID)
{
    FCImageCodec_Gdiplus::AUTO_INIT_GDIPLUS() = FALSE ;

    FCObjImage   img ;
    img.Load (L"c:\\test.jpg") ;

    return TRUE ;
}
```

... Miscellaneous Function (Win Only)

C++

```
FCObjImage   img ;

// capture current screen
HDC   screen_dc = GetDC(NULL) ;
RECT  rc = {0, 0, GetSystemMetrics(SM_CXSCREEN), GetSystemMetrics(SM_CYSCREEN)} ;
img.CaptureDC (screen_dc, rc) ;
ReleaseDC (NULL, screen_dc) ;

// copy this image to clipboard
img.CopyToClipboard() ;

// get image in clipboard
img.GetClipboard() ;

// convert image to HRGN
img.ApplyEffect (FCEffectThreshold(128)) ;
::SetWindowRgn (hWnd, img.CreateHRGN(), TRUE) ;
```

History

- **2011 - 11 - 27, V7.0**
  + Code Refactoring, more features, more efficient, more easier to use
- **2007 - 03 - 11, V4.0**
  + Add `FCPixelFillGradientFrame`
  + Add `FCPixelLensFlare` / `FCPixelTileReflection` / `FCPixelSoftGlow` effect
  * Modify example
  * Improve `FCObjImage::AlphaBlend`
  * Modify `FCPixelBlinds`
  * Modify brightness/contrast/hue/saturation/emboss
  * Rewrite gauss blur processor
  - Remove `FCPixelDeinterlace`
  - Remove `FCPixelAddRandomNoise`
  - Remove `FCPixelFill3DSolidFrame`
  - Remove `FCImageHandleFactory_IJL15` and `FCImageHandle_IPicture`
- **2006 - 10 - 25, V3.0**
  * Improve `FCImageHandle_Gdiplus` class to load multi-frame gif/tiff image and load jpeg's EXIF information
  * Improve `FCImageHandle_FreeImage` class to save gif with transparency color
  * Change `FCPixelHueSaturation`'s hue arithmetic
  * Change `FCPixelColorTone`'s arithmetic, more look like old photo
  * Change inner `FCImageHandleBase` interface, it's never mind for user
  * Substitute `std::fstream` by ANSI C file function, because of a bug in VC2005
  + Add `FCImageProperty` to store image's property, function `FCObjImage::Load` and `FCObjImage::Save` support it
  + Add example 010: Load jpeg's EXIF information via GDI+
  - Remove `FCObjImage::GetNextFrameDelay` and `FCObjImage::SetNextFrameDelay`, you can get them from `FCImageProperty`
- **2006 - 09 - 07, V2.0**
  + More mature
- **2006 - 03 - 11, V1.0**
  + Initial version

License

Share

About the Author

crazybit
Team Leader PhoXo
China

graduate from University of Science and Technology of China at 2002.

Now I work at www.phoxo.com.

Comments and Discussions

You must Sign In to use this message board.

First Prev Next

Next step suggestion Pin
Michael Chourdakis **30-Jun-20 4:02**

Good. Pin
Michael Chourdakis **29-Jun-20 5:51**

Can't compile it! (Use full free Visual Studio 2017) Pin
karaulov **2-Oct-17 7:26**

Re: Can't compile it! (Use full free Visual Studio 2017) Pin
Doug Rogers 24-Feb-20 5:11

My Vote of 5 and a question Pin
Eli1995 **9-Sep-14 6:18**

Nice one Pin
Moeen Naqvi **26-May-14 0:50**

Simply great Pin
v_krishan **30-Jul-13 9:01**

My vote of 4 Pin
Mable John **19-Mar-13 0:30**

How can I increase speed of redraw image, when I using scrolling? Pin
hdnn **6-Dec-12 3:27**

Can't draw text! Pin
StefanoA **1-Aug-12 4:05**

Re: Can't draw text! Pin
crazybit 2-Aug-12 0:37

Re: Can't draw text! Pin
StefanoA 2-Aug-12 0:54

Re: Can't draw text! Pin
crazybit 2-Aug-12 1:13

Re: Can't draw text! Pin
StefanoA 2-Aug-12 2:27

Re: Can't draw text! Pin
crazybit 2-Aug-12 22:46

Problem Pin
adri9102 **22-Jun-12 4:14**

Need help to understand the logic. Pin
mbatra31 **23-May-12 4:51**

Re: Need help to understand the logic. Pin
crazybit 23-May-12 16:17

Good job! Pin
Aamer Alduais **11-May-12 21:30**

My vote of 5 Pin
Aamer Alduais **11-May-12 21:29**

How can i add imagestone effects to my application with out using the load funtion of FCObjImage class. Pin
sunny chouhan **9-May-12 5:29**

Re: How can i add imagestone effects to my application with out using the load funtion of FCObjImage class. Pin
crazybit 13-May-12 22:38

Print functionality Pin
jasondemont **19-Mar-12 12:14**

Region or ROI (region of interest) Pin
boisselazon **19-Mar-12 4:15**

Relation between GDI+ Image Class and FCObjImage Pin
boisselazon **15-Mar-12 5:08**

General News Suggestion Question Bug Answer Joke Praise Rant Admin

Use Ctrl+Left/Right to switch messages, Ctrl+Up/Down to switch threads, Ctrl+Shift+Left/Right to switch pages.