

SplashIt

Introduction

SplashIt is a graphics tool, designed by a developer for developers and novices alike that can quickly integrate and generate a splash screen for your application. You can use the splash dll / splashlite dll modules to quickly create Gdi+ bitmaps and generate splash images in your applications. The Splashbin project folder can be used as a template for creating / integrating with your own splash application. What separates this tool from others is that it utilizes the Windows Imaging Component (WIC) integrated with a substantial amount of GDI+ v1.1 to load and render images thus saving you time. The source code is written in C/C++ and is designed to be run in Win Vista/7/8 environments or greater on the Windows platform.

Included Components:

- **SplashIt** – Project folder including source code for SplashIt GUI application
- **Splashdll** – Project folder including source code for Splash dll which is the main image processing library
- **Splashlitedll** – Project folder including source code for Splashlite dll which is a lite version of the splash dll
- **Splashbin** – Project folder including source code for the template splash execution program
- **SplashIt.chm** – SplashIt help file. (Useful for reference information).
- **Icon and Image files** used in the SplashIt application.

Included Sub Components (Dependencies):

SplashIt:

- EZTwain Classic library - EZTW32.dll
- Splash library – Splash.dll
- IDR_RCDATA1: Resource Splashbin application file - Splashbin.exe
- IDR_RCDATA2: Resource PECompact console program file - PEC2.EXE
- IDR_RCDATA3: Resource default loader for PECompact file - pec2ldr_default.dll
- IDR_RCDATA4: Resource PECompact Codec file - pec2codec_aplib.dll
- IDR_RCDATA5: Resource PECompact Codec file - pec2codec_lzma.dll
- IDR_RCDATA6: Resource PECompact lzma compression program file - lzma.exe

Splashdll (Dependencies):

- FFTW Fast Fourier Transform library - libfftw3-3.dll

Splashbin (Dependencies):

- IDR_RCDATA1: Resource Splashlite library - Splashlite.dll
- IDR_RCDATA2: Splashitinfo structure template binary file - test.bin

Build Instructions

SplashIt was built under Windows 7, Visual Studio 2013 on a 32 bit system. The source code can be copied into new project folders as is. For your convenience the project files *.vcxproj are included for VS2013 users, otherwise use the following parameters as a guideline:

SplashIt – Project Configuration Properties

General:

- Name = Release|Win32
- Configuration Type = Application (.exe)
- Use Of MFC = Use MFC in a Shared DLL
- Use of ATL = Not Using ATL
- Character Set = Use Unicode Character Set

C/C++ (Note: Include relevant files for your IDE / version. Files below pertain to VS2005 and may not be the same as yours):

- Additional Include Directories = "C:\Program Files\Microsoft SDKs\Windows\v6.1\Include"
- Preprocessor Definitions = WIN32;NDEBUG;_WINDOWS;
- Optimization = Your choice
- Enable C++ Exceptions = Yes (/EHsc)
- Runtime Library = Multi-threaded DLL (/MD)
- Use Precompiled Header = Not Using Precompiled Headers
- Calling Convention = __stdcall (/Gz)
- Compile As = Compile as C++ Code (/TP)

Linker ((Note: Include relevant files for your IDE / version. Files below pertain to VS2005 and may not be the same as yours):

- Additional Dependencies = WindowsCodecs.lib GdiPlus.lib
- Additional Library Directories = "C:\Program Files\Microsoft SDKs\Windows\v6.1\Lib"
- Sub System = Windows (/SUBSYSTEM:WINDOWS)
- Optimization = Your choice
- Entry Point = wWinMainCRTStartup
- Target Machine = MachineX86 (/MACHINE:X86)

Splashdll / Splashlitedll – Project Configuration Properties

General:

- Name = Release|Win32
- Configuration Type = Dynamic Library (.dll)
- Use Of MFC = Use Standard Windows Libraries
- Use of ATL = Not Using ATL
- Character Set = Use Unicode Character Set

C/C++ (Note: Include relevant files for your IDE / version. Files below pertain to VS2005 and may not be the same as yours):

- Additional Include Directories = "C:\Program Files\Microsoft SDKs\Windows\v6.1\Include"
- Preprocessor Definitions = WIN32;NDEBUG;_WINDOWS;_USRDLL;SPLASH_EXPORTS
- Optimization = Your choice
- Enable C++ Exceptions = Yes (/EHsc)
- Runtime Library = Multi-threaded DLL (/MD)
- Use Precompiled Header = Not Using Precompiled Headers
- Calling Convention = __cdecl (/Gd)
- Compile As = Compile as C++ Code (/TP)

Linker ((Note: Include relevant files for your IDE / version. Files below pertain to VS2005 and may not be the same as yours):

- Additional Dependencies = WindowsCodecs.lib GdiPlus.lib msimg32.lib
- Additional Library Directories = "C:\Program Files\Microsoft SDKs\Windows\v6.1\Lib"
- Sub System = Windows (/SUBSYSTEM:WINDOWS)
- Optimization = Your choice
- Entry Point =
- Target Machine = MachineX86 (/MACHINE:X86)
- Module Definition File = Splash.def / splashlite.def (Note: used to demangle exports)

Splashbin – Project Configuration Properties

General:

- Name = Release\Win32
- Configuration Type = Application (.exe)
- Use Of MFC = Use Standard Windows Libraries
- Use of ATL = Not Using ATL
- Character Set = Use Unicode Character Set

C/C++ ((Note: Include relevant files for your IDE / version. Files below pertain to VS2005 and may not be the same as yours):

- Additional Include Directories = "C:\Program Files\Microsoft SDKs\Windows\v6.1\Include"
- Preprocessor Definitions = WIN32;NDEBUG;_WINDOWS;
- Optimization = Your choice
- Enable C++ Exceptions = Yes (/EHsc)
- Runtime Library = Multi-threaded (/MT)
- Use Precompiled Header = Not Using Precompiled Headers
- Calling Convention = __stdcall (/Gz)
- Compile As = Compile as C++ Code (/TP)

Linker ((Note: Include relevant files for your IDE / version. Files below pertain to VS2005 and may not be the same as yours):

- Additional Dependencies = WindowsCodecs.lib GdiPlus.lib msimg32.lib
- Additional Library Directories = "C:\Program Files\Microsoft SDKs\Windows\v6.1\Lib"
- Sub System = Windows (/SUBSYSTEM:WINDOWS)
- Optimization = Your choice
- Entry Point =
- Target Machine = MachineX86 (/MACHINE:X86)

Interdependencies

- All projects share some common header files, however, there may be subtle differences. (i.e. splashitinfo.h will contain Exports for dll's and Imports for exe's). The respective *.def files for dll projects will demangle any exported functions. Please keep this in mind if developing your own applications from the provided source.
- The Splashbin application is dependent on the splashlite.dll and a binary representation of the splashitinfo structure test.bin (which may have been previously saved in the SplashIt application using a different name).
- The splashlite application is a stripped down version of the splashdll application for use in your own application. You can, of course, elect to use the splash.dll instead for more functionality.
- In your application you can elect to either link the splash / splashlite dll files statically or load them dynamically as is the case in the provided splashbin application.
- The SplashIt application uses a number of PECompact files to optionally compress an output application that consists of the splashbin.exe file as a stub program, with the original application and image file as appended sections in the output application. The output application, when executed, starts the splashbin.exe program which then extracts the splashlite.dll and test.bin file (from Resource data) to the temp folder, displays the splash screen image, then via a ShellExecute command, starts the original application.
- Know and understand the variables in the splashitinfo.h file. This is the main communication structure within the splashit programs. This file also contains pointers to information which must be defined (if used).
- The splash/splashlite dll applications DO NOT contain Windows Imaging Component functions to open / save files. This functionality (by design) is in the splash.cpp file within the SplashIt project. It is incumbent on you the developer to know your target audience and which image types are acceptable. Keep it simple.

Quick Start

- The splashbin project is a valuable template for building your own splash screen in your application. You may elect to include the splashlite.dll and an image file as Resources in your own application, or use them externally. Your choice.
- In your program's main startup routine, simply define any imports (functions) you will use from the splashlite.dll (if used dynamically) then perform the following steps:

1. Define and initialize your splashitinfo structure:

```
lpsplashitinfo = new splashitinfo();
memset(lpsplashitinfo,0,sizeof(splashitinfo));
```

Remember to set your splashitinfo variables before calling. Important:

```
lpsplashitinfo->isrunning = 1; // turn runsplash on
```
 2. Call InitApp to start GDI+ and COM.
 3. Call RunSplash to optionally extract any of your resource files (image, splashit bin file, etc.) necessary for splash screen generation and / or display your splash screen.
 4. Setup your msg loop to intercept the WM_QUIT msg for splash screen termination and destruction.
 5. Call ExitApp to stop GDI+ and COM.
 6. Run your application normally.
- Note: You can also use the splashlite.dll to optionally load images into your program, etc.

Why use SplashIt?

- Convenient and powerful tool for loading, manipulating and displaying splash / non splash images of different image types.
- Fewer lines of code so that you can spend less time developing your own splash screen routines and concentrate on your application.
- Develop new features and functionality to improve upon the existing framework that can benefit the developer community.
- Learn how to process image bitmaps in GDI / GDI+ / Windows Imaging Component (WIC), Fast Fourier Transform (FFTW) for use in your application.
- Generate a splash screen for your custom application, quickly and easily with a few mouse clicks
- New, Open, Edit, Cut, Copy, Paste, Paste as new image, Draw, Crop, Resize, Rotate, Shear, Zoom, Print, Scan, Select, Save image files, and more...
- Splash Dll support
- Standalone SplashIt Exe support
- Splashbin VS2005 project application for developing your own splash screen application
- Easily integrates a splash screen into an application using a limited amount of code
- Alphablending (transparency) 0 to 100% including fadein / fadeout
- Over 40 different Linear / nonlinear filters that utilize the GDI+ matrix (support for 3x3 and 5x5 matrices) and special effects
- Custom textures / effects
- Scan Image Support via EZTwain Classic provides very easy access to basic TWAIN services to import images into the application
- Hotkeys, toolbar menu and menu support
- Restoration module including (Fast Fourier Transform) High / Low Pass filters, support for kernel (point spread function) deblurring including deblurring filters (Wiener and Total Variation with regularization), state-of-the-art color balancing filters, image denoising and image inpainting capabilities.

- Color options (contrast, brightness, saturation, hue, tint, balance, gamma, etc.)
- Over a dozen different text format options with Font, style, full color attributes, including support for selection, relative text pixel position, horizontal, vertical, shear and angle text orientations
- Designate any one of the more than 1,000,000 colors available as a transparency (Region) color
- Animated Gif support for gif spec "Gif89a" up to 128 frames
- Animation support for 12 custom slide effects
- Three different selection methods to choose from: rectangle, ellipse and freehand including zoom selection
- Preview mode for filters, effects, color, text, redeye, restoration and texture
- 1 level of undo / redo
- Unicode compliant, support files up to 1000 characters in length
- Full printing support includes page and print setup, preview and print dialogs
- Multiple frame support for .tif, .png, and .gif files which support more than one frame per image. You can view, alter and independently save any frame to a file
- Drag & Drop support
- Window's Explorer context menu shell open command support

Why use GDI+ and WIC?

- GDI+ is a powerful tool for working with digital images and contains many easy to use built-in functions with floating point precision. It simplifies and improves upon many features that would be much more difficult to do in straight GDI.
- WIC is open source architecture for designing, developing and using your own encoders / decoders for saving / loading images respectfully, with many powerful functions for extracting image metadata. You may install as many windows image codecs (WIC) for encoding / decoding images as you require. This framework can then make immediate use of these codecs.

Solutions

Hopefully, many questions posted on forums and such can be answered by viewing and using the provided source code in this framework. Questions such as:

1. How do I implement the Windows Imaging Component (WIC) in my application?
2. How do I implement GDI+ in my application?
3. How do I manipulate a bitmap in GDI+?
4. How do I use Object Linking and Embedding (OLE)?
5. How do I manage a modal dialog that communicates with MFC SDI?
6. How do I add text to a bitmap?
7. How do I use the standard print / save dialogs in my application?
8. How do I modify a standard dialog?
9. How do I save a WIC bitmap?
10. How do I use a Matrix in GDI+?
11. How do I use Paths in GDI+?
12. How does alpha blending work in GDI+?
13. How can I add drag & drop support to an application?

14. How can I utilize a toolbar in a modal dialog?
15. How can I Resize, Scale, Rotate, Translate and Shear a bitmap in GDI+?
16. How can I write my own filter to manipulate a bitmap in GDI+?
17. How can I implement Fast Fourier Transform in my application?
18. What is meant by spatially invariant kernel and the point spread function (PSF)?
19. How do Linear / Non Linear filters work with GDI+?
20. How do color matrices work with GDI+?
21. How can I use GDI+ brushes in my application?
22. How can I use gradients with GDI+ in my application?
23. How can I interface GDI and GDI+ in my application?
24. How can I implement gif animated files in my application?
25. How can I use shortcuts in my application?
26. How can I modify / reference a PE file structure in my application?
27. How to Spawn Console Processes with Redirected Standard Handles.
28. How can I update a Resource in a PE file?
29. How can I utilize a Progress bar in my application?
30. How can I implement Structured exception handling in my application?
31. How can I utilize Slider controls in my application?
32. How can I utilize Tab controls in my application?
33. How can I print a GDI+ bitmap?
34. How can I reference a Class to preview my bitmap images in different dialogs?
35. How can I access / update Icon files in my application?
36. What is image restoration and how does it work?
37. What does the concept of image resolution mean in GDI+?
38. How can I access the individual color channels in GDI+?
39. How can I scan an image in VC++?
40. How can I draw on a GDI+ bitmap?

... and possibly many more can be answered, or at least partially answered.

Todo (Wish) list

- Incorporate newer versions of GDI+ as they become available.
- Incorporate newer versions of WIC as they become available.
- Incorporate Direct2D as an imaging tool.
- Add new features and functionality to the SplashIt framework.
- Create separate classes of information to be used in external programs in lieu of using the splash / splashlite dll files.
- Add hardware acceleration to digital image processing.
- Consider a plug-in interface.