e2eSoft ® Slideshow SDK

This documentation describes the e2eSoft® Slideshow Software Development Kit (SDK).

The e2eSoft Slideshow SDK enables developers to combine two images with real time transition and apply real time effect on a single image in codes. It provides hundreds of transitions and effects. And it also provides some basic image features including loading images, saving images, resample and rotate etc.

The SDK are optimized with MMX, SSE and SSE2, some transitions are processing with GPU. So the SDK has high performance. The effects and transitions are based on plug-ins mode, so it can be extended in the future.

The SDK is easy to use in your codes. It deals with 32bits bitmap data. It provides COM interfaces that can be used in almost all develop languages for example: Visual C++, C++ Builder, Visual Basic, C# and so on.

The key features of the e2eSoft Slideshow SDK are:

- Load images from file, stream and buffer. The SDK supports loading JPEG, BMP and PNG image data. The SDK supports saving images into BMP file only. And you can pass a buffer to the image class, so the processing will be applied on the buffer.
- Image help functions. The SDK provides some useful functions for image including: resample (bilinear interpolation and cubic convolution), rotate, fill with color etc.
- About three hundreds of transitions. There are many kinds of transitions like: page roll, 3D flip, 3D explosion, wipe, shift, zoom in, zoom out, fade in, fade out and so on. All these have high performance even with big images.
- Provides real time image effects including: sharpen, blur, low pass, high pass, solarize, gamma correction, white balance etc.

To download the e2eSoft Slideshow SDK, see the below link at the e2eSoft web site:
http://www.e2esoft.cn/slideshow/sdk.asp

This document describes how you can use the SDK in your codes to deal with images or videos. It is divided into the following sections.

- **Being Ready for Using SDK**
  This section provides information of files and folders that the SDK offers.
- **Image Features**
  This section provides overview and background information of the image interface.
- **Transition Features**
  This section provides detailed instructions for performing transitions, such as creating, applying transition, and so on.
- **Programming Reference**

This section provides detailed information of each interface and methods.

1. Being Ready for Using SDK

The SDK is provided as a zip package. Unzip the package and there are following folders:

Folder	Contents
root	SDK DLL, COM and executable demos for the demonstration programs
.\transtion	Transition plug-ins
.\effect	Effect plug-ins
.\pics	Sample pictures for demo
.\inc	Header files for C++ library and header files for COM
.\lib	SDK library file. COM doesn't need this
.\doc	This document
.\samples	Source projects of C++ SDK demo, C++ COM demo, C# demo

The SDK provides two methods to use: DLL mode and COM mode, in fact the COM mode SDK encapsulates the DLL mode.

The DLL mode offers dll (SImage.dll) and library (SImage.lib) that export functions and class. It's easy to use DLL mode for C++ developers and don't need to know any COM knowledge. But it's hard to use this mode in C# and Visual Basic etc.

The COM mode can be used in all develop languages including C++, Visual Basic and C#, even with Delphi.

1.1 Using COM Mode

There are four files are relative with the SDK COM: SlidePL.dll, _SlidePL.h, _SlidePL_i.c and _SlidePL.tlb.

- **SlidePL.dll**
  It's the COM DLL of the SDK. For C# and Visual Basic developers, import this into the project and all is done. Before using the SDK or running the demo, please register this dll in system like: "regsvr32 slidepl.dll".
  It's in "bin" folder.
- **_SlidePL.h**
  It defines the interfaces that provided by the SDK. C# and Visual Basic developers don't need this. C++ users should include this in somewhere.
  It's in "inc" folder.
- **_SlidePL_i.c**
  It defines the CLSID of interfaces that provided by the SDK. C# and Visual Basic developers don't need this. C++ users should include this where create the

interfaces.
It's in "inc" folder.

- **_SlidePL.tlb**
  It's the type lib of SDK. Visual Basic developers may import this file for using the SDK.
  It's in "inc" folder.

Before using the transitions or effects, must tell the SDK what folders the transition plug-ins and effect plug-ins are in. Do this by "**ISlideFactory::SetTransitionDir**" and "**ISlideFactory::SetEffectDir**". Then the SDK will load plug-ins in these folders. The transition plug-ins folder is ".\bin\transition", and the effect plug-ins folder is ".\bin\effect'.

1.2 Using DLL and Library Mode

DLL and Library Mode is for C++ developers, it's hard to use this mode in Visual Basic and C#, but it's the best way to use the SDK in Visual C++ application. In this mode, don't need to create COM interface, just class and functions.

In this mode, include the header "SImageConf.h" in "inc" folder, add the library file "simage.lib" in "lib" folder.

Before using the transitions or effects, must tell the SDK what folders the transition plug-ins and effect plug-ins are in. Do this by "SInitTransitionLibrary" and "SInitEffectLibrary". Then the SDK will load plug-ins in these folders. The transition plug-ins folder is ".\bin\transition", and the effect plug-ins folder is ".\bin\effect'.

1.3 About Source Projects of Demo

There are three demos in the "samples" folder, they are all source project that tell how to use the SDK in C++ and C#.

C# Demo
WpfDemo is a C# demo which uses the SDK in C#.

C++ COM Demo
DemoSD is a C++ demo which uses the SDK COM mode.

C++ DLL and Library Demo
Demo is a C++ demo which uses the SDK DLL and Library mode.

2. Image Features

All transitions and effects of SDK deal with the image interface. This section demonstrates how to create an image interface and detail of the image interface.

The image interface is **ISlideImage**. The ISlideImage holds image data with ARGB format (A8G8B8B8). When you load an image from file, stream or buffer, ISlideImage will convert the data into ARGB format automatically.

2.1 Create an ISlideImage instance

Following codes tell how create an ISlideImage instance:

Visual C++

```
#include "_SlidePL.h"
#include "_SlidePL_i.c"
ISlideImage* pImage = 0;
CoCreateInstance(CLSID_CSlideImage, NULL, CLSCTX_INPROC_SERVER
   , IID_ISlideImage, (void**)&pImage);
```

C#

```
private SlidePL.CSlideImage m_SDImageA;
m_SDImageA = new SlidePL.CSlideImage();
```

2.2 Load image or Attach Image with a buffer

Following codes tell how to load a file into the image instance:

Visual C++

```
pImage->LoadFile(L"C:\\pics\\a.jpeg");
```

C#

```
m_SDImageA.LoadFile(ImageFileName);
```

Sometimes you don't need to load image from file, just want to set the dimension of image with **ISlideImage::SetSize(int Width, int Height)**.

Sometimes you don't need the image interface maintain its own image buffer, you want the image processing on the buffer or array you offered. At this time, you should use **ISlideImage::Attach** method.

2.3 Get information of the image

Any time you can get information of an image instance with **GetWidth**, **GetHeight**, **IsReady** etc.

2.4 Get image content into a buffer or BYTE array

After transitions or effects applied, you may need to get the image content into a buffer or array. The **GetBits** do this for you. Following codes tell how to use this method:

Visual C++

```
m_Bits = new BYTE[800 * 4 * 600];
pImage->GetBits(m_Bits, 800 * 4);
```

In up codes the GetBits method will fill the m_Bits buffer with image content.

C#

```
private byte[] m_Bits;
int height = m_SDImageA.GetHeight();
int width = m_SDImageA.GetWidth();
int pitch = width * 4;
int total = pitch * height;
m_Bits = new byte[total];
m_SDImageA.GetBits(ref m_Bits[0], pitch);
```

In up C# codes, we allocate a byte array named m_Bits, and call GetBits to fill the array with image content.

3. Transition Features

The transitions are offered with plug-ins which's postfix is "**trn**", you can find two transition plug-ins in the "transition" folder: e2e3DTransition.trn and e2eTransition.trn. You should tell the SDK the transition folder first.

Before create transitions for using, you should create an **ISlideFactory** instance. This interface is used to load transition plug-ins and effect plug-ins, get how many kinds of transitions are provided, and get detail information of a kind of transition or a kind of effect.

The **ISlideTransition** presents a transition, this interface can't be created directly, it should be created by **ISlideFactory:: CreateTransition** with the transition ID. The transition ID presents a kind of transition, it begins from 0 and the max can be get by **ISlideFactory:: GetTransitionCount** minus **1**.

3.1 Create a Transition

Don't create ISlideFactory every time when you need use it, it's better to create it once and save it as a member of your class.

Following codes tell how to create global factory and create a transition instance:

Visual C++

```
class CDemoSDDlg : public CDialog
{
private:
    ISlideFactory*      m_SDFactory;
};

BOOL CDemoSDDlg::OnInitDialog()
{
    // Initialize the COM library
    ::CoInitialize(NULL);

    // Create the factory
    CoCreateInstance(CLSID_CSlideFactory, NULL, CLSCTX_INPROC_SERVER
       , IID_ISlideFactory, (void**)&m_SDFactory);

    // Set plug-in dir
    WCHAR Dir[256];
    GetModuleFileNameW(NULL, Dir, 256);
    PathRemoveFileSpec(Dir);
    wcscat(Dir, L"\\transition");
    m_SDFactory->SetTransitionDir((BSTR)Dir);
```

```
    // Add all transition's name into the listbox
    LONG lTransitionNum = 0;
    m_SDFactory->GetTransitionCount(&lTransitionNum);
    for (LONG i = 0; i < lTransitionNum; i++) {
        CComBSTR bstrName;
        m_SDFactory->GetTransitionName(i, &bstrName);
        m_CtrlTransitions.AddString((BSTR)bstrName);
    }
    return TRUE;
}
```

In update codes we create factory first, then we load the plug-ins with SetTransitionDir method. After that we get the transition type number and add them into a List Box.

When the some item in list box has selected, we create a transition with the transition ID like following codes:

```
void CDemoSDDlg::OnLbnSelchangeListEffects()
{
    int iCurSel = m_CtrlTransitions.GetCurSel();
    if (iCurSel >= 0) {
        ISlideTransition* pTransition = 0;
        m_SDFactory->CreateTransition(iCurSel, 0, &pTransition);
        if (pTransition) {
            if (m_Transition) m_Transition->Release(), m_Transition = 0;
            m_Transition = pTransition;
        }
    }
}
```

C#

```
private void LoadTransition()
{
    // get current working dir
    string strDir = System.Environment.CurrentDirectory;
    // the transition Plug-ins are put here
    strDir += "\\transition";
    // create the SDK factory component
    m_SDFactory = new SlidePL.CSlideFactory();
    // tell the factory where is the transition plug-ins folder
    m_SDFactory.SetTransitionDir(strDir);
```

```
    // get transitions and add them into listbox
    int iTransitionCount = m_SDFactory.GetTransitionCount();
    for (int i = 0; i < iTransitionCount; i++)
    {
        // get transition name by ID
        System.String strName = m_SDFactory.GetTransitionName(i);

        // new an item instance
        System.Windows.Controls.ListBoxItem item = new
System.Windows.Controls.ListBoxItem();
        tem.DataContext = i; // saving ID
        item.Content = strName; // saving Name
        // add to listbox
        Transitions.Items.Add(item);
    }
}
```

In up codes we new a factory instance and save it as a member of main window. Then we get the transition kind number, for each kind of transition get its name and add the name string into a list box.

```
private void Transitions_SelectionChanged(object sender,
SelectionChangedEventArgs e)
{
    ListBoxItem lbi = ((sender as ListBox).SelectedItem as ListBoxItem);
    int iTransitionID = (int)(lbi.DataContext);
    // create transition by ID
    m_SDTransition = m_SDFactory.CreateTransition(iTransitionID, 0);
}
```

When an item of list box is selected, we create a transition by the selected item's DataContext. The item's Datacontext saving the transition's ID.

3.2 Do Transition

After get a transition instance, you can do transition with this instance between two images with the method **ISlideTransition::DoEffect**:

DoEffect(ISlideImage* imgC, ISlideImage* imgA, ISlideImage* imgB, LONG percent, LONGLONG param)

The DoEffect method has five parameters.

- The imgC is saving processed image.
- The imgA is the source image A.
- The imgB is the source image B.

- The percent is between 0 and 100 which presents the transition position you want. If the percent is 0, in general the image A is copied to image C. If the percent is 100, in general the image B is copied to image C.
- The last parameter “param” is not used, it should be passed as “0”.

The image C, image A and image B should have same width and height.

4. Programming Reference

Here we list all interfaces of the SDK and all method of each interface.

4.1 ISlideImage

ISlideImage maintain images and provides some help functions to deal with images. It's the base of SDK.

Sometimes you want to process a sub region of an image not entire image, at this time, the **Attach** method is useful for this. Please refer to the Attach method.

- **IsReady([out, retval] LONG* lReady)**

  Function:

  Judge if the image is ready for using.

  Parameter:

  lReady

  [out, retval] receive if the image is ready.

  1: ready, 0: not ready.

  Return:

  Returns 1 for image is ready, returns 0 for image is not ready.

- **GetWidth([out, retval] LONG* lWidth)**

  Function:

  Get image width.

  Parameter:

  lWidth

  [out, retval] receive the image's width.

  Return:

  Returns the width of image.

- **GetHeight([out, retval] LONG* lHeight)**

  Function:

  Get image height.

  Parameter:

  lHeight

  [out, retval] receive the image's height.

  Return:

  Returns the height of image.

- **GetPitch([out, retval] LONG* lPitch)**

  Function:

  Get the number of bytes in one scan line.

Parameter:

lPitch

[out, retval] receive the image's pitch.

Return:

Returns the pitch of image.

- **SetSize(LONG lWidth, LONG lHeight)**

Function:

Set the dimension of image.

Parameter:

lWidth

[in] Specifies the image width, in pixels.

lHeight

[in] Specifies the image height, in pixels.

Return:

None

- **GetBits(BYTE* Bits, LONG lPitch)**

Function:

Retrieves the bits of the image and copies them into a buffer or an byte array with ARGB format.

Parameter:

Bits

[out] Pointer to a buffer or a ref array to receive the image data.

lPitch

[in] Specifies the number bytes of one scan lines.

Return:

None.

Note:

For Visual C++ developer, pass a pointer of a buffer to the Bits. For C# pass the ref of an array to Bits, like below:

```
int pitch = width * 4;
int total = pitch * height;
m_Bits = new byte[total];

// copy image A's content to displaying image
m_SDImageA.GetBits(ref m_Bits[0], pitch);
```

- **SaveBitmap(BSTR FileName)**

Function:

Save the image as a file.

Parameter:

FileName

[in] Specifies the full path file name.

Return:

None

- **LoadBits(BYTE* lBits, LONGLONG lCount)**

  Function:

  Load image from buffer or a byte array.

  Parameter:

  lBits

  [in] Specifies pointer of a buffer or ref of a byte array.

  lCount

  [in] Specifies the bytes number of the buffer.

  Return:

  If succeeded returns S_OK else returns E_FAIL.

- **LoadFile(BSTR Filename)**

  Function:

  Load image from file.

  Parameter:

  Filename

  [in] Specifies the full path file name.

  Return:

  If succeeded returns S_OK else returns E_FAIL.

- **LoadStream(IStream* pStream)**

  Function:

  Load image from an instance of IStream.

  Parameter:

  pStream

  [in] An instance of IStream.

  Return:

  If succeeded returns S_OK else returns E_FAIL.

- **Attach(LONG lWidth, LONG lHeight, LONG lPitch, BYTE* lBits)**

  Function:

  This method attaches a buffer or a byte array to the image object. So all image processing will be applied directly to this buffer. And the image object will not allocate buffer inside.

  Parameter:

  lWidth

  [in] Specifies the image width, in pixels.

  lHeight

  [in] Specifies the image height, in pixels.

  lPitch

[in] Specifies number of bytes in one scan line.

lBits

[in] Pointer to a buffer or ref of a byte array.

Return:

Returns S_OK.

Note:

This function is very useful when you want the SDK to process your image buffer directly. Following is a C# sample:

```
// allocate a byte array to store image C's bits
int pitch = width * 4;
int total = pitch * height;
m_Bits = new byte[total];

// pass the array to Image C, the ImageC will hold this bits
m_SDImageC.Attach(width, height, pitch, ref m_Bits[0]);
```

- **Fill(LONG lAlpha, LONG lRed, LONG lGreen, LONG lBlue)**

  Function:

  Fill the image with specified color.

  Parameter:

  lAlpha

  [in] Specifies the alpha value, between 0~255.

  lRed

  [in] Specifies the red value, between 0~255.

  lGreen

  [in] Specifies the green value, between 0~255.

  lBlue

  [in] Specifies the blue value, between 0~255.

  Return:

  Returns S_OK.

- **CopyFrom(ISlideImage* pImageSouce)**

  Function:

  Copy another image’s content to this image.

  Parameter:

  pImageA

  [in] Specifies a source image.

  Return:

  Returns S_OK

  Note:

  If the dimension of the source image is different with current image, the current image’s dimension will be changed to the same with the source image.

- **ResizeTo(ISlideImage* pDestination, LONG lInterpolation)**

  Function:

  Resample current image into the destination image with the destination image's width and height.

  Parameter:

  pDestination

  [in] Pointer to an image instance which is the destination image.

  lInterpolation

  [in] Specifies the resample flag. It could be 0 or 1.

  0: bilinear interpolation

  1: cubic convolution

  This flag would be extended in future.

  Return:

  Returns S_OK

4.2 ISlideFactory

ISlideFactory interface is used to load transition and effect plug-ins, get plug-ins information, get how many transition and effect supported, create transition and create effect etc. You can't new an instance of transition or effect directly, you should create them with the ISlideFactory.

Don't create ISlideFactory each time when you want to create transition. It's better to create a global ISlideFactory once in your application.

Before you use ISlideFactory's other methods relative with transitions, please call ISlideFactory::SetTransitionDir to init transition plug-ins. And then call ISlideFactory::GetTransitionCount to get how many kinds of transition the SDK providing.

- **SetTransitionDir(BSTR Dir)**

  Function:

  Init transition component and load all transition plug-ins in specifies folder.

  Parameter:

  Dir

  [in] Specifies the transition plug-ins folder.

  Return:

  Returns the transition numbers.

- **GetTransitionCount([out,retval] LONG* lCount)**

  Function:

  Get how many kinds of transition provided.

  Parameter:

  lCount

  [out] Get the transition numbers.

  Return:

  Returns the transition numbers.

- **GetTransitionName(LONG lID, [out,retval] BSTR* Name)**

  Function:

  Get the name of a kind of transition with transition ID.

  Parameter:

  lID

  [in] Specifies transition ID.

  Name

  [out] Transition name.

  Return:

Returns S_OK.

Note:

All transition names follow the same conventions: use backslashes (\) to group the transition. For example a transition named: wipe\clock, and another transition named: wipe\dropwater, another transition named: 3d\flip. All these like window's path, so you can organize them as group.

- **GetTransitionDescription(LONG lID, [out,retval] BSTR* Description)**

Function:

Get the detail of transition.

Parameter:

lID

[in] Specifies transition ID.

Name

[out] Transition defail.

Return:

Returns S_OK.

- **GetTransitionPlugInFileName(LONG lID, [out,retval] BSTR* Filename)**

Function:

Get which plug-in export this transition.

Parameter:

lID

[in] Specifies transition ID.

Name

[out] Transition plug-in file name.

Return:

Returns S_OK.

- **CreateTransition(LONG lID, LONGLONG lParam, [out,retval] ISlideTransition** pTransition)**

Function:

Create a transition instance by ID.

Parameter:

lID

[in] Specifies transition ID, from 0 ~ (GetTransitionCount() - 1).

lParam

[out] Reserved. Set this parameter to 0.

pTransition

[out, retval] Pointer to a transition interface, representing the created transition.

4.3 ISlideTransition

The ISlideTransition applies transition on image A and image B, and stores the result to image C. Image A, B and C should have same width and height.

Don't create an ISlideTransition instance directly, create it from ISlideFactory.

- `DoEffect(ISlideImage* C, ISlideImage* A, ISlideImage* B, LONG percent, LONGLONG param)`

  ```
  Function:
      Do transition.
  Parameter:
      C
          [in] Destination image handle.
      A
          [in] Source image A.
      B
          [in] Source image B.
      percent
          [in] Specifies the transition pos, between 0 ~ 100.
      param
          [in] Reserved. Set this parameter to 0.
  Return:
      Returns S_OK if succeeded, else returns E_FAIL.
  ```