Step by Step guide to the Surround Video API

Pre-Release Version

Build 160

1Overview

This document attempts to show a simple approach to using the Surround Video Toolkit. It might be useful to read this before delving into the sample code. All the examples in this document were taken from the SVVIEWER sample included in the toolkit, but here you don't have to wade through all of the non-surround video code to see what is going on.

2What is the Surround Video Toolkit ?

The Surround Video Toolkit consists of

- Surround Video Prep tool (SVEDIT.EXE)
- Surround Video Component Object (SURROUND.DLL)
- Surround Video Library (SURROUND.LIB)
- Surround Video API Header file (SURROUND.H)
- Sample code (SVVIEWER and HOTSPOT)
- Sample Images
- Surround Video ActiveX™ control (SVIDEO.OCX)
- Sample HTML files and supporting images
    - Surround Video Image Link Editor (LINKEDIT.EXE)

2.1Surround Video Prep Tool (SVEDIT.EXE)

The Prep tool is used to create Surround Video Images (*.svi files) from images taken with a 360° camera. Please see the documentation provided with the prep tool for more information.

2.2Surround Video Component Object (SURROUND.DLL)

The Surround Video Component Object is the heart of the Surround Video API. It is a single OLE object with several interfaces. Your application code uses these interfaces to interact with the Surround Video Image.

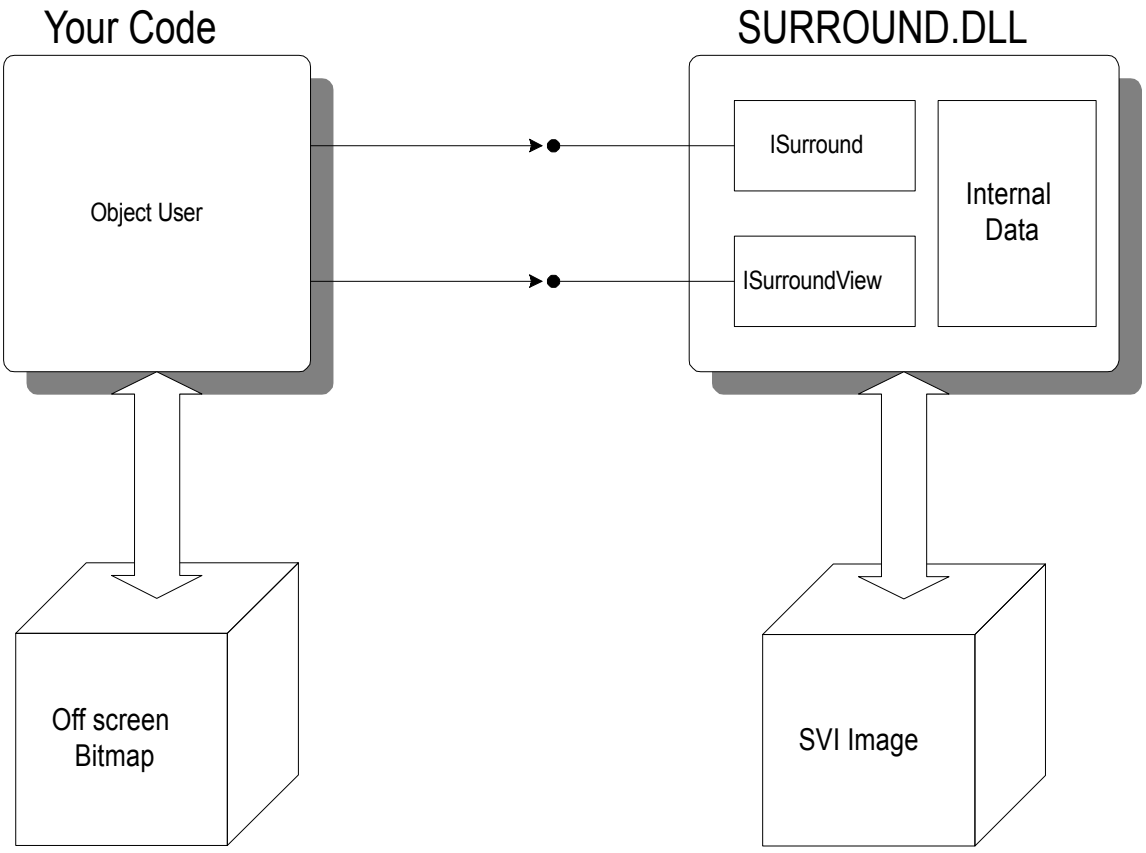

Figure 1

3Create an ISurround Object and Initialize it

Before you can do anything with a Surround Video Image, you must load the Surround Video Component Object (surround.dll) and instantiate an ISurround Object.

3.1Initialize the OLE libraries

The first step is to initialize the OLE libraries. This is usually done in ***CWinApp:: InitInstance()***.

```
// Initialize OLE libraries
if (!AfxOleInit())
{
    AfxMessageBox(IDP_OLE_INIT_FAILED);
    return FALSE;
}
```

3.2Instantiate an ISurround object

Next we instantiate an ISurround object using one of three C functions contained in the file SURROUND.LIB. Please look at the ***SVViewer*** sample application for an example of how to use the following functions.

3.2.1PanoramicSurroundFromDib()

This function takes a Dib and uses it as the source of the image bits for the newly created ISurround.

```
HRESULT PanoramicSurroundFromDIB(
    LPBITMAPINFOHEADER lpbmi,          // Pointer to Dib header
    LPRGBQUAD lpColors,                // Pointer to Dib color table
    LPVOID lpBits,                     // Pointer to Dib bits
    int iHorizon,                      // Horizon of image
    ARCSECONDS extent,                 // horizontal extent of the image
    ISurround** ppISurround);          // Pointer to returned ISurround interface
```

The Horizon of the image (***iHorizon***) controls the internal image correction for cylindrical images and is normally the center of the image (1/2 the image height) but can be different due to cropping of the origional image.

3.2.2PanoramicSurroundFromFile()

This function creates an ISurround based on a file. This file is assumed to have been created by SVEdit or any other application that adheres to the Storage/Stream naming convention used by SVEdit .

```
HRESULT PanoramicSurroundFromFile(
    LPCTSTR lpszFilename,              // Filename of the SVI file
    UINT iDepthRequested,              // Bit depth requested
    ISurround** ppISurround);          // Pointer to returned ISurround interface
```

Since certain Surround Video Images can support both 8 and 24 bit ISurrounds, the ***iDepthRequested*** is used to determine what mode the SVI file or stream is opened with. If the requested bit depth is not available, an HRESULT of **SV_E_INCOMPATIBLE_SURFACE** is returned.

3.2.3PanoramicSurroundFromStream()

This function is essentially the same as ***PanoramicSurroundFromFile()*** except it uses stream to get it's data.


```
HRESULT PanoramicSurroundFromStream(
    IStream __RPC_FAR *pStream,        // pointer to an existing IStream
    UINT iDepthRequested,              // Bit depth requested
    ISurround** ppISurround);          // Pointer to returned ISurround interface
```

3.2.4PanoramicSurroundFromPartialStream()

This function is essentially the same as ***PanoramicSurroundFromStream()*** except it takes an offset and number of valid bytes in the stream. It is used in conjunction with the URL Moniker functions used to progressively download an image from a World Wide Web page. The stream length is updated by calls to ISurround::UpdateStreamLength() from the ***IBindStatusCallback::OnDataAvailable()*** method as data becomes available.

```
HRESULT PanoramicSurroundFromStream(
    IStream __RPC_FAR *pStream,        // pointer to an existing IStream
    UINT iDepthRequested,              // Bit depth requested
    ISurround** ppISurround,           // Pointer to returned ISurround interface
    DWORD dwValidBytes,                // Number of valid bytes in the stream
    DWORD dwOrigin);                   // Offset of SVI data in stream
```

3.2.4.1ISurround::UpdateStreamLength()

```
HRESULT ISurround::UpdateStreamLength(
    DWORD dwValidBytes,                // Number of valid bytes in the stream
    BOOL FAR* pbUpdate);               // *pbUpdate is set to TRUE if new data
                                       // has been read and needs to be painted
                                       // to the screen (if NULL, nothing
                                       // is returned)
```

We now have a valid ISurround Object which has been initialized. The next step is to get a view on it so we can look around.

4Create an ISurroundView Object and Display it

4.1Instantiate an ISurroundView Object

In the *svviewer* sample app, we check to see what the maximum size view we can create. This is done using the ***ISurround::GetMaxViewSize()*** function.

```
SIZE maxViewSize;
float fZoom = 1.0f;                         // Zoom factor of the view (1:1)
int iViewQuality = 100;                     // Quality setting of view (0 to 100)
DWORD dwFlags = SV_TOTAL_CORRECTION;        // Vertical and Horizontal correction

// Get the maximum size view we can create with this zoom factor
m_pISurround->GetMaxViewSize( fZoom, &maxViewSize );
```

Now we create the view.

```
HRESULT hr;

hr = pISurround->GetView( &maxViewSize, fZoom, iViewQuality,
                          dwFlags, &pISurroundView );
if( FAILED(hr) )
    ....error....
```

Now we determine what range of locations are valid for this view. The ***ISurroundView::Draw()*** function takes a location relative to the center of the viewport when rendering to your off screen Bitmap so we would like to know what locations are valid. We also need to know what the latitude at the center of the image.

NOTE

it can not be assumed that the center of the image is exactly at the equator (0°) because the image is usually cropped after scanning. This value is set using the Surround Video Prep tool (*svEdit*) and is derived from the image's horizon.

```
SPHERE_RECT viewExtents;
SPHERE_POINT location;
HRESULT hr;

result = pISurroundView->GetViewRange( &viewExtents, &location.latitude );
```

4.2Render the Image

At this point we have an ISurroundView object and we know the range of possible locations. All that is left to do is create an off screen Bitmap that is compatible with the view (it has the same size and color depth) and use ***ISurroundView::Draw()*** to render the portion of the image you wish to see. You will also need to create a palette based on the colors in the image.

```
SPHERE_POINT location;
BITMAPINFOHEADER* pDib;
VOID* pBits;
RECT viewRect;
int iDrawQuality = 100;    // Draw with maximum quality

hr = pISurroundView->Draw( &location, &pDib, pBits, &viewRect, iDrawQuality );
```

Notice the *iDrawQuality* parameter. It is sets the draw quality of the image and ranges from 0(lowest) to 100(highest). It can be used to perform faster rendering while panning.

5Summary

Please look at the SVViewer sample code. It shows how to open a file (svi or bmp/dib) and create an ISurround using the appropriate functions. Please see the online help file for a complete description of the Surround Video API.