Scan Conversion Trails Colour

How to use the SPx Scan Converter under Windows to support the display of fade trails in different colours to live video

Summary

It is sometimes desirable to display radar video with trails in one colour and new data in a different colour.

This permits the recent history of target positions to be seen using the trails, whilst still permitting new data to be clearly identified.

This note outlines the procedure for creating this situation with the SPx Radar Scan Converter under Windows.

The process is explained for the use of the SPx C++ library, although the process is similar when using RDC.

The SPx scan converter can be used to create history trails, so that the historical position of targets can be easily observed as a trail of slowly fading video. This is shown in Figure 1, where the movement of two targets is clearly visible as a trail of decaying green video away from the direction of motion.

Using the SPx software, this situation is accommodated by using Sweep type fading, which means that the radar video is decayed on each sweep of the radar. The time constant of this type of fade is specified in terms of the number of sweeps of the radar over which a peak-value signal will decay from maximum brightness to zero. In the example of Figure 1, it is 30 sweeps.

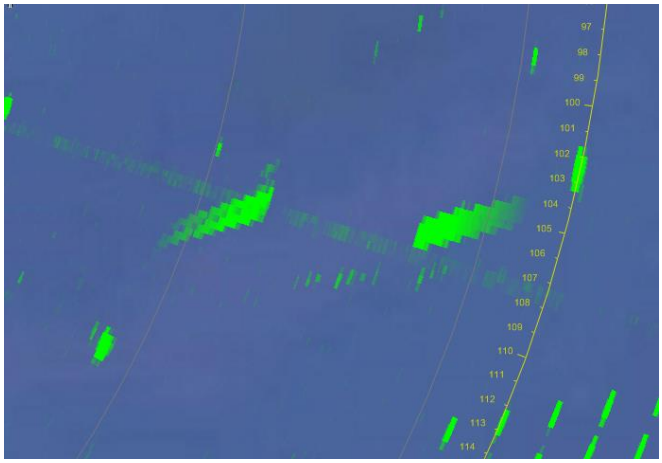
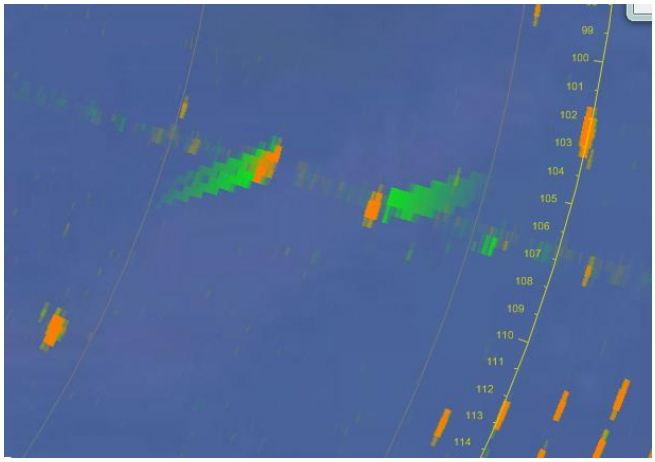


Figure 1 - Single scan converter configured with long fading to generate video trail history

One potential problem of the above display is that it is very difficult to distinguish new radar video from older video. For example, a target that initially is displayed at peak intensity becomes half that intensity after 15 scans. This 15-scan-old video would then be indistinguishable from new video (i.e. from the most recent scan) that is initially half brightness. It is often desirable to be able to look at the video picture and easily identify new video.

This is accomplished in SPx by creating two scan converter objects, which both output their images into the same Window to give a single composite picture. Using the scan converter configuration described above as a starting point, we need to create a second object, which processes the same input data, but which is configured in a slightly different way. The second scan converter is configured to be on top of the first (higher window priority), with no fading (in SPx this is called replacement mode) and the radar colour is chosen to be different from that of the first scan converter.



The combination of the "history trails" scan converter and the "new video" scan converter is shown in Figure 2. The video shown in orange is the output from the second scan converter and this represents the new video that is less than 1 scan old. The historical video is shown in green, as before.

Figure 2 - Dual scan converters configured to provide long term trails (green) and new video (orange)

In terms of SPx software modules, the dual scan converter is simply two `SPxScSourceLocal` scan converters working on the same input data and both outputting into the same Window. The general configuration of SPx objects for this situation is shown below.

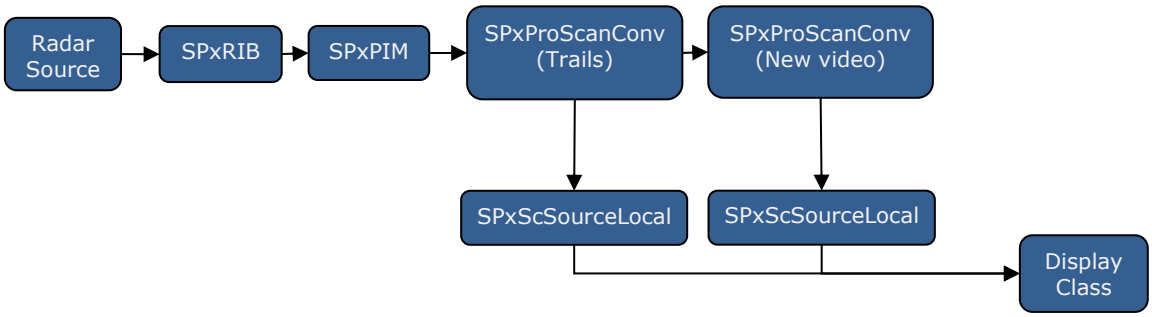


Figure 3 - Configuration of SPx objects to support scan conversion of new and old video in different colours.

Figure 3 shows the configuration of objects and the sequence of processing from the PIM data store into the display. As illustrated, there are two SPx processes, both `SPxProScanConv`. The first of these is the scan converter that handles the trails (configured with sweep fade). The second of these is the scan converter that handles the new video (configured with no fade). The Windows associated with these scan converters must have a stacking order that puts the New Video scan converter on top of the Trails scan converter. This is achieved by creating the Window for the New Video *after* the

Window for the trails. Both scan converters are configured to access video data from the same PIM.

Trail History Retained on View Changes

If the above configuration is implemented as described, it will create the display shown in Figure 4. However, if the view displayed in the window is changed by using a SetView() command, i.e. because the operator wants to pan/scroll or change the zoom factor, then the complete trail history (in green) gets lost. It will build up again in the new view, but this will take many scans.

The solution to this is to include support for trail history retention. This is an additional process that preserves the trail history when the view changes. It means that once trails have established, the view in the radar picture can be changed and the historical video (in green) remains.

The SPx objects to provide the complete capability, with the two scan converters and the retained trails is shown in the Figure 4. The changes to accommodate the trail history processing are shown in green boxes. The `SPxProHistory` process manages the history using the additional History PIM. When the first scan converter (the one handling the trails) needs to refresh the radar image at a new scale it uses the history PIM, instead of the normal input PIM.

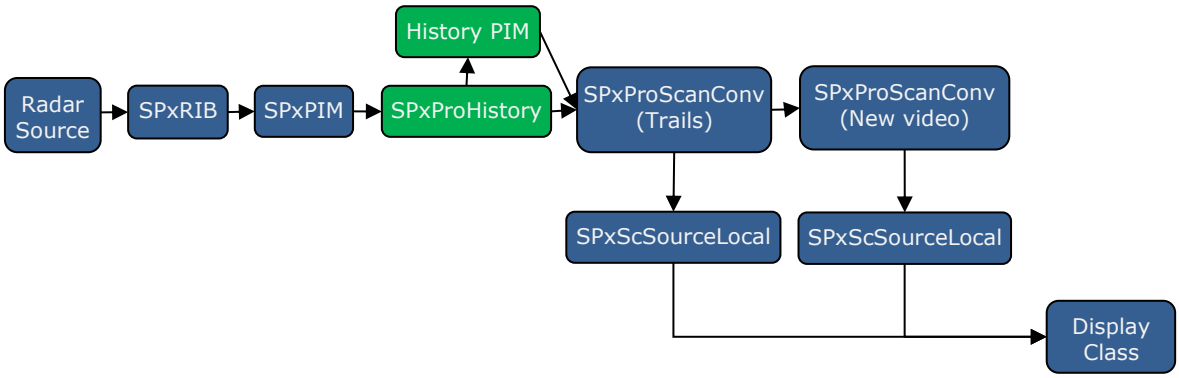


Figure 4 - Configuration of SPx objects to add trail history retention to the two colour scan conversion

Sample Code

The code fragment below is based on the standard example `SPxWin2ColourSC`, which is supplied as a worked example in the SPxExamples directory of the Windows Development package. The output from running the example is shown in Figure 5. The test pattern used by this example shows a number of targets moving out from the radar position.

```
BOOL CSPxWin2ColourSCDlg::OnInitDialog()
{
    …
    // SPx Initialisation
    SPxInit();

    // Create the Windows. Note that the trails Window is created before the New Radar window

    // Create the window for radar video 1(trails), giving it a maximum size of 512 x 512
    SPxScDestDisplayWin * spxDisplayWin1 = new SPxScDestDisplayWin();
    spxDisplayWin1->Create(0, 512, 512, GetDlgItem(IDC_RADAR_AREA));
    spxDisplayWin1->SetUpdateInterval(30);

    // Create the window for radar video 2 (new radar) giving it a maximum size of 512 x 512
    SPxScDestDisplayWin * spxDisplayWin2 = new SPxScDestDisplayWin();
    spxDisplayWin2->Create(0, 512, 512, GetDlgItem(IDC_RADAR_AREA));
    spxDisplayWin2->SetUpdateInterval(30);

    // Create the RIB. Size is 1 Mb, with memory allocation done by class.
    SPxRIB * spxRib = new SPxRIB(1024*1024, NULL);

    // Create the PIM to provide the polar storage.
    SPxPIM * spxPim = new SPxPIM(spxRib, 2048, 2048, SPX_PIM_RAN_PEAK,
                                 SPX_PIM_AZI_PEAK, SPX_PIM_OUTPUT(1));

    SPxPIM * historyPim = new SPxPIM(NULL, 2048, 2048);

    /* Create scan converter 1, telling it which display window to use and
    configuring it for sweep fade with 30 scans and green colour
    */
    spxSc1 = new SPxScSourceLocal(spxDisplayWin1);
    spxSc1->SetFade(SPX_RADAR_FADE_SWEEP, 30);
    spxSc1->SetRadarColour(0, 0, 255,0);

    /* Create the scan converter 2, telling it which display window to use and
    configuring it for no fade with orange colour video
    */
    spxSc2 = new SPxScSourceLocal(spxDisplayWin2);
    spxSc2->SetFade(SPX_RADAR_FADE_REPLACE,0);
    spxSc2->SetRadarColour(0, 255,200,0);

    // History process for trail retention.
    SPxRunProcess *historyProcess = new SPxRunProcess(SPxProHistory, NULL,  spxPim, historyPim);

    /* Create scan-conversion process 1 to link the pim to scan-converters. This scan converter
    handles the trails, so link to the history PIM and link the scan conversion object to the
    history process.
    */
    SPxRunProcess *spxScProcess1 = new SPxRunProcess(SPxProScanConv, historyProcess,
                                       spxPim, historyPim, spxSc1);
    spxSc1->SetRedrawHistoryProcess(historyProcess);

    /* Create the second scan conversion process, which handles new data. */
    SPxRunProcess *spxScProcess2 = new SPxRunProcess(SPxProScanConv, spxScProcess1,
                              spxPim, spxSc2);

    // Locate the radar window at the position of the Window defined by item IDC_RADAR_AREA
    SPxScFollowWin *winFollow = new SPxScFollowWin(spxSc1, GetDlgItem(IDC_RADAR_AREA));
    winFollow->AddSC(spxSc2);

    // Create a source of data and configure test picture 42, with a period of 2 seconds.
    SPxTestGenerator *spxTestGen = new SPxTestGenerator(spxRib, 2048, 2.0, (int)(2048/2.0), 42, 0);

    // Turn on the test generator.
    spxTestGen->Enable(1);
```

```
    /* Set the view in the window. Note that we need to set BOTH scan converters
    to be the same view, which is initially 100km.
    */
    spxSc1->SetView(0,0,100000,0);
    spxSc2->SetView(0,0,100000,0);

    return TRUE;  // return TRUE  unless you set the focus to a control
}
```

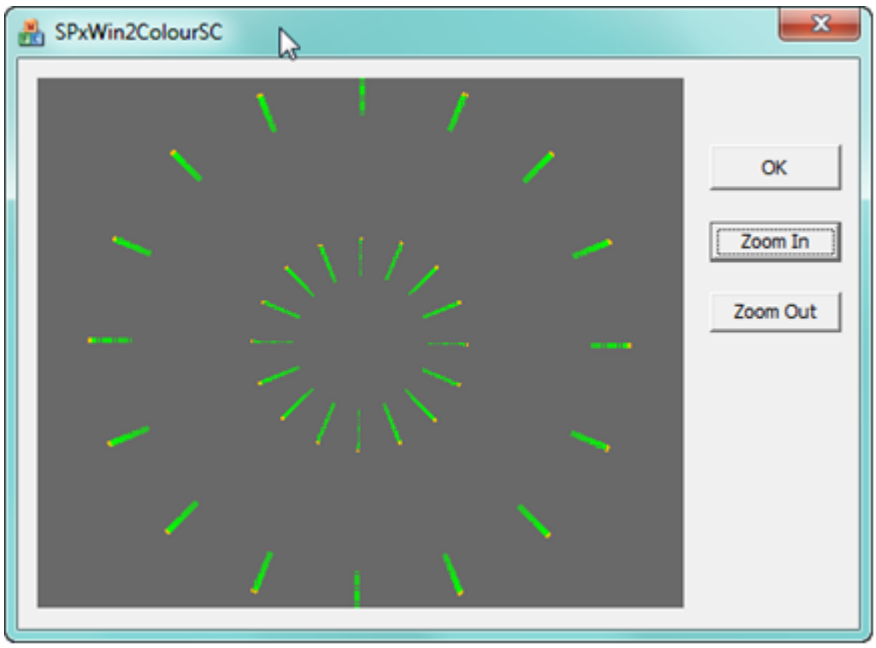


Figure 5 - Output from SPxWin2ColourSC example.
There are two scan conversions, which give trails in green and new video in orange.
The radar input source is test pattern 42 from the standard Test Generator.

When the **Zoom In** or **Zoom Out** buttons are pressed, the `SetView()` function is called on each of the two scan converter objects. This causes the view in the window to change to a new scale. Significantly, the trail history is *preserved* by the Trail History Retention process.

Note the following key lines from the code for the correct behavior of trail history retention:

1. The history PIM is created.

```
SPxPIM * historyPim = new SPxPIM(NULL, 2048, 2048)
```

2. The History process is created and the history PIM is provided as an argument, after the input PIM:

```
SPxRunProcess *historyProcess = new SPxRunProcess(SPxProHistory, NULL,
                                                  spxPim, historyPim)
```

3. When the scan converter for the trails is created, the history PIM is provided as an argument:

```
SPxRunProcess *spxScProcess1 = new SPxRunProcess(SPxProScanConv, historyProcess,
                                    spxPim, historyPim, spxSc1)
```

4. The trail scan conversion object is linked to the history process:

```
spxSc1->SetRedrawHistoryProcess(historyProcess)
```

< End of document >