RDC - True and Relative Trails with Moving Platform

How to use the SPx Radar Display Co-process (RDC) to support displays based on true or relative motion, with target history shown as true or relative trails in a different colour.

Summary

Cambridge Pixel's SPx Radar Display Co-process (RDC) software provides a simple means of inserting radar video into existing graphics applications. The display location may be referenced to the world (true motion) or to the platform (relative motion) based on navigational input to the RDC notifying updates to own ship position.

The display can be referenced as north-up (world referenced) or heading-up/course-up (referenced to platform heading or course).

A further enhancement to the radar display uses trails to show the historical position of targets as a trail of slowly fading video, typically presented in a different colour from live radar video for clarity.

These trails can be shown either as true trails (where motion relative to the world is shown) or as relative trails (where any motion relative to own ship, including fixed artefacts such as coastlines as well as static or moving targets, is shown). Trails may be ground stabilised (true trails are stationary relative to the ground) or sea stabilised (true trails are stationary relative to the sea).

This application note describes the use of moving platform trails within RDC client applications.

Navigation Data Input

The RDC must be configured to receive navigation data as NMEA-0183 sentences or AIS VDO messages from a serial line or network interface. Set the **NavData.Available** parameter in the RDC configuration file to "1" to enable moving platform support, and configure the other parameters to set the source for navigation data.

For example, for network input from multicast address "239.192.50.79" and port 5079:

```
NavData.Available = 1
NavData.NetAddr = "239.192.50.79"
NavData.Port = 5079
NavData.SerialPort = ""
NavData.SerialBaud = 0
NavData.Type = 0
```

The client RDC application may also need access to the navigation data. It can either receive the data directly from the same network source as the RDC, or use the RDC remote parameter interface to query the current platform position, course and heading using the **NavData.LatDegs**, **NavData.LongDegs**, **NavData.CourseDegs**, and **NavData.HeadingDegs** parameters.

Updating the View

When the view within the radar video window is changed, this change must be communicated to the RDC. The view may be set either based on metres from the platform position, or using Lat/Long. The view must be set on the **ViewCtrl*N*** object. In older versions of the RDC the view was set on the **CompWin*N*** object.

The old style of setting the view on the **CompWin*N*** object can be simply migrated to the **ViewCtrl*N*** object as follows:

```
// Old style of setting the view in metres from the platform.
m_rdc->SetRemoteParameter("CompWin0", "View", "0 0 10000 10000");

// New style of setting the view in metres from the platform.
m_rdc->SetRemoteParameter("ViewCtrl0", "CentreMetresDisplay", "0 0");
m_rdc->SetRemoteParameter("ViewCtrl0", "WidthMetres", "10000");
```

Ideally, the view should be set as a Lat/Long:

```
// New style of setting the view as Lat/Long.
m_rdc->SetRemoteParameter("ViewCtrl0", "CentreLatLong", "51.508 -0.128");
m_rdc->SetRemoteParameter("ViewCtrl0", "WidthDegs", "0.5");
```

Setting the Mode

The client application must inform the RDC of the display mode, motion mode and trails mode that is required.

The display motion is set using the **Server.DisplayMotionMode** parameter:

0 = Relative Motion
1 = True Motion

In relative motion mode the client application may need to retrieve the updating view centre position (as Lat/Long) to update other graphics being displayed in the radar video window.

The display reference is set using the **Server.DisplayRefMode** parameter:

0 = World-Referenced or North-Up
1 = Heading-Up
2 = Course-Up

The trails mode is set using the **Server.TrailsMode** parameter:

0 = Normal, trails the same as the display motion mode
1 = Relative trails in true display motion mode
2 = True trails in relative motion mode.

The stabilisation mode is set using the `Server.StabiliseMode` parameter:

0 = Ground stabilised trails
1 = Sea stabilised trails

Note that heading-up or course-up display reference modes with true trails is only supported with RDC External Compositing and requires the client to perform an additional rotation before display. The mechanism for performing this transform is detailed in the Heading-Up / Course-Up with True Trails section below.

Highlight new Video

It is often desirable to display recent radar video in a different colour to the fading radar trails. The RDC supports this mode by setting the **Win*M*.HighlightAvailable** parameter to "1" in the RDC configuration file. In this mode, each channel of radar video has two scan converters associated, which may be configured with different colours. One for live video, and one for trails video.

Heading-Up / Course-Up with True Trails

RDC supports Heading-Up and Course-Up radar video with True Trails by performing scan conversion North-Up and requiring the client application to rotate the received radar video before display. Heading-Up or Course-Up radar video with True Trails is only supported with the External Compositing mode of RDC, where radar video is written to a shared memory bitmap by the RDC.

As well as the usual display, motion and trails modes, the client application must opt-in to this behaviour by setting the **Server.RefMode** parameter to "1". The client application then uses the **DisplayWidth** and **DisplayHeight** parameters on the **Bitmap*M-N*** object to get the size of the radar video to copy from the shared memory bitmap, and uses the **DisplayMatrix** parameter to retrieve the required transformation matrix to apply before display. The code sample below demonstrates the use of the DisplayMatrix with Direct2D. Applying the matrix with other graphics libraries will be similar, but may require use of all the matrix elements, or swizzling of the matrix elements.

```
// Retrieve display matrix from RDC.
char displayMatrixStr[512];
m_rdc->GetRemoteParameter("Bitmap0-0", "DisplayMatrix",
                          displayMatrixStr, sizeof(displayMatrixStr));

// Parse the string into its 9 matrix components.
// Direct2D is for 2D drawing so only the first 6 elements are required.
D2D1_MATRIX_3X2_F transform = {};
float dummy1 = 0.0F;
float dummy2 = 0.0F;
float dummy3 = 0.0F;
std::sscanf(displayMatrixStr, "%f %f %f %f %f %f %f %f %f",
            &transform._11, &transform._21, &transform._31,
            &transform._12, &transform._22, &transform._32,
            &dummy1, &dummy2, &dummy3);
```

```
// Apply the transform.
m_d2dTarget->SetTransform(transform);

// The bitmap to render is larger than the display window in this mode.
float displayW = 0.0F;
float displayH = 0.0F;
m_rdc->GetRemoteParameter("Bitmap0-0", "DisplayW", &displayW);
m_rdc->GetRemoteParameter("Bitmap0-0", "DisplayH", &displayH);
rectD2D = D2D1::RectF(0, 0, displayW, display);

// Draw radar video bitmap to the window.
m_d2dTarget->DrawBitmap(bitmap, rectD2D, 1.0F,
                        D2D1_BITMAP_INTERPOLATION_MODE_NEAREST_NEIGHBOUR,
                        rectD2D);
```

A full demonstration with all modes supported is available for Direct2D and WPF in the **SPxD2DRdcMovingPlatform** and **SPxDNRdcMovingPlatformWpf** examples respectively.

< End of document >