# See Reports in Your Mind's Eye

*Doug Hennig*

**Interested in more flexibility in your reports? Want to output to PDF files? How about graphs? The Mind's Eye Reporting Engine is an incredible new tool that'll make you forget the VFP Report Preview window. In this month's article, Doug explains why.**

In her book "The Visual FoxPro Report Writer" (Hentzenwerke Publishing, www.hentzenwerke.com), Cathy Pountney describes many of the limitations of the VFP Report Designer and Report Preview window. Cathy, the queen of VFP reporting, provides workarounds for some of these limitations, including how to fake multiple detail bands in a report, make objects line up properly when they're below stretchable objects, and so forth. However, many of these workarounds are complex, and often still don't give you exactly what you want.

Until recently, the only solutions you had for these issues were to use the workarounds or use a different report writer, such as Crystal Reports. However, now there's a third alternative: the Mind's Eye Report Engine (MERE). MERE is a new tool written by Rich Simpson, president of Mind's Eye, Inc., that provides an ActiveX control and a VFP class library. The ActiveX control, which Rich wrote in Delphi, is a report rendering control that provides properties, events, and methods (PEMs) for outputting text and graphics to a preview window, a printer, and even a PDF file. The VFP class library provides several classes, including one called MindsEyeReportEngine that knows how to output a VFP FRX report to the ActiveX control.

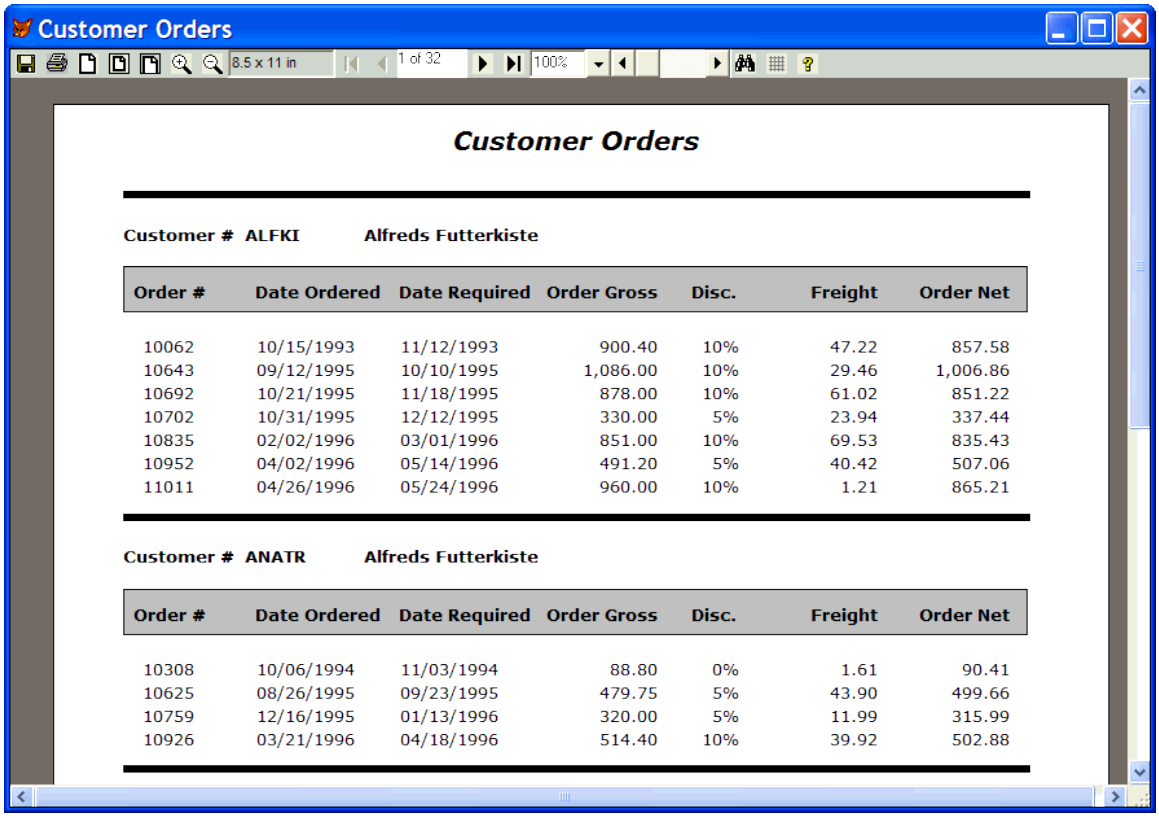
## Running an FRX report

Previewing an FRX in the MERE takes just a few lines of code. First, you instantiate the MindsEyeReportEngine class in MindsEyeReportEngine.VCX. This VFP class is responsible for rendering an FRX report in the MERE. You set any properties of the class you wish and then call the ReportFormPreview method to preview the FRX. Here's an example (Orders.PRG in the Subscriber Downloads for this month):

```
#if version(5) >= 700
local loRE as MindsEyeReportEngine of MindsEyeReportEngine
#endif
set classlib to MindsEyeReportEngine
loRE = createobject('MindsEyeReportEngine')
loRE.lDisplayExecutionTime     = .F.
loRE.lShowReportPreviewToolbar = .F.
loRE.cReportPreviewFormCaption = 'Customer Orders'
loRE.ReportFormPreview('orders.frx', 2)
```

The #IF statement brackets the LOCAL statement, which provides IntelliSense for the MindsEyeReportEngine class, so this code can run in VFP 6 if necessary. I set the lDisplayExecutionTime property to .F. because I don't want the class to show how long it took to render the report, the lShowReportPreviewToolbar property to .F. because I want the MERE toolbar used rather than a VFP toolbar, and the cReportPreviewFormCaption property to the caption to use for the MERE window. The call to ReportFormPreview passes the name of the FRX to render and the window style (0 for a window in _SCREEN, 2 for a top-level form). You can also optionally pass the left, top, width, and height values for the window. Running this code produces the result shown in Figure 1.

*Figure 1. The Mind's Eye Report Control has a lot more features than the VFP Report Preview window.*




As you can see, the MERE has a lot more features that the VFP Report Preview window, plus it's a lot easier to control since it's an object. Let's look at some of the features of the MERE.

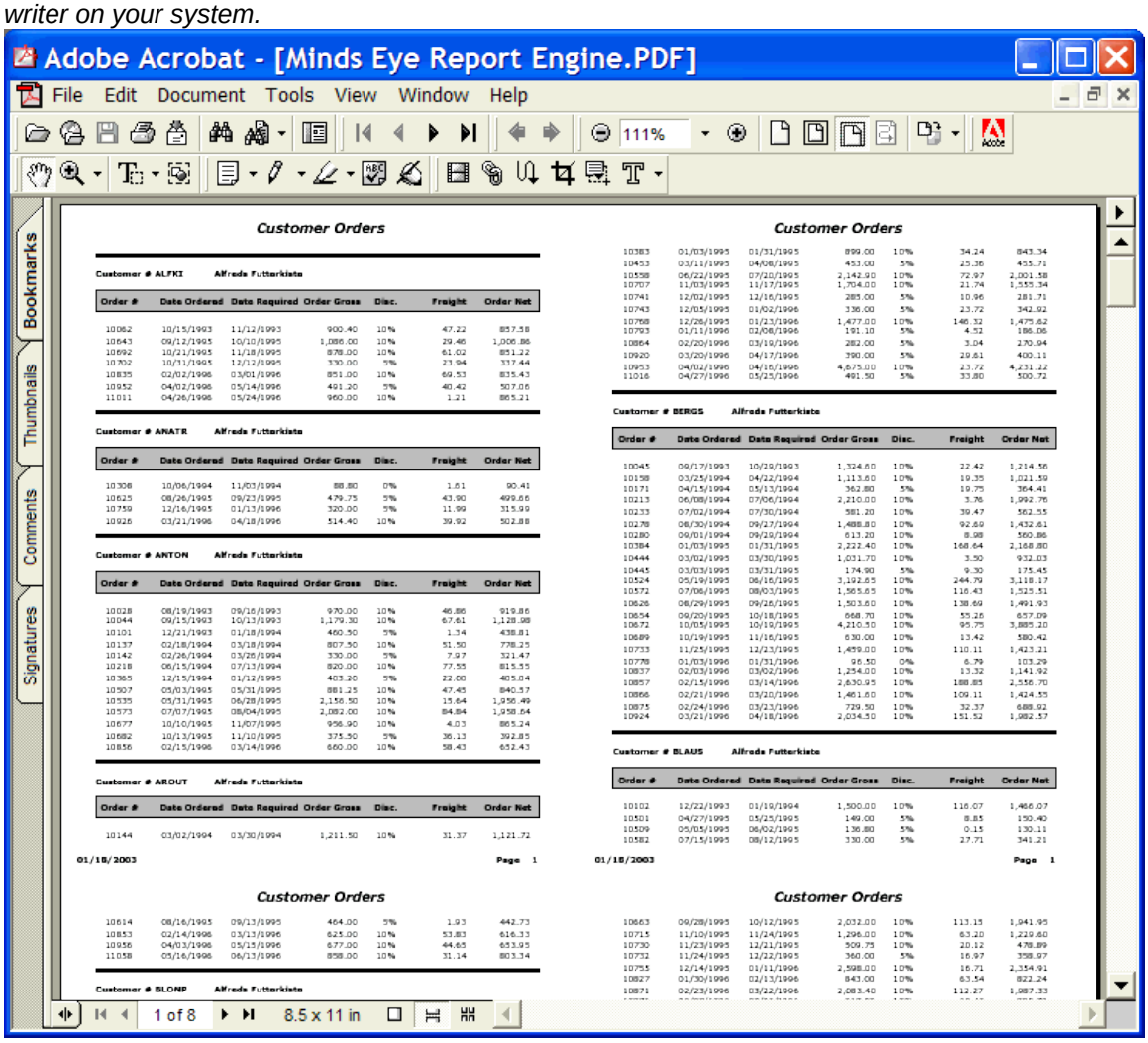
The first thing you'll notice is the various ways you can control how the report is displayed in the window. The third button in the toolbar displays the report at its actual size, as shown in Figure 1. The fourth button reduces the report so the entire page fits in the window. The fifth button adjusts the report so it's about as wide as the window. The next two buttons zoom out or zoom in at 5% increments. The zoom scale combo box allows you to choose from a number of pre-defined zoom scales or type in a specific zoom value. Believe it or not, you can scale a report up to 1,000,000%, a size at which each pixel of each letter takes an entire page. (OK, that's not really practical; I think Rich was just showing off when he decided to allow values that high!)

There are several controls for page navigation. The traditional navigation buttons move you to the first, previous, next, and last pages of the report, and a status control shows the current page out of the total number of pages. There's also a mini-scroll bar to the right of the zoom scale combo box that moves in larger increments than single page at a time (depending on the number of pages in the report). You'll notice a couple of things about these controls when you use them. First, a nice feature is that you can hold down the mouse button to continuously scroll through the pages rather than having to click over and over. Second, the navigation controls are incredibly fast. Clicking on the last page button, for example, instantly displays the last page. In the VFP Report Preview window, there's a noticeable pause before the last page is displayed, and in complex reports with calls to user-defined functions or many pages, it can take a long time to even move from one page to the next (moving backwards through pages seems to be considerably slower than moving forwards). One of the reasons the MERE is so fast is that the report is completely rendered in the engine, as opposed to VFP's window, which only renders one page at a time. The downside of this approach is that it takes longer for the report to initially appear in MERE as the control is fully loaded with report pages, but as of this writing, Rich is working on a way to display the first page as soon as possible and then continue to render pages in the background.

There are several other controls in the MERE toolbar. The first button is sure to get your attention: save to PDF. Click on this button to display a dialog of PDF file options, including the file name, title, and author; when you choose OK, MERE creates a PDF file from your report. Here's the best part: you don't

need to install a PDF writer such as Adobe Acrobat or Amyuni on your system. This feature alone makes MERE well worth the price! It also supports 1, 2, 4, or 8 report pages per PDF page. This is handy if you need a very compact report. Figure 2 shows the PDF file generated from Orders.FRX with 4 pages per sheet selected.

*Figure 2. The Save To PDF button creates a PDF file from the report without requiring a PDF writer on your system.*




The print button prints the report, first displaying a dialog in which you can select the printer and options such as number of copies and pages per sheet. The find button is very cool—it highlights in yellow instances of text you specify in a search dialog. The grid button shows or hides a background grid on the report. The help button currently does nothing, but Rich is adding properties in which you can specify the name of a help file and the context ID to use for the MERE.

## More Control

You have a lot of control over how the MERE displays a report. There are dozens of properties controlling things like the background color of the window and which buttons are visible in the toolbar. You can even add a watermark or a calendar overlay to a report. Table 1 shows a few of the properties; the MERE help file lists all of them.

*Table 1. Some of the MERE properties that control how a report is displayed.*

| Property | Description |
|---|---|
| DisplayGridButton | .T. to display the grid button in the toolbar. |
| DisplayHelpButton | .T. to display the help button in the toolbar. |
| PDFEncrypt | .T. to encrypt the PDF file when it's created. |
| PDFPassword | The password to use for an encrypted PDF file. |
| WatermarkText | The text for a watermark on a page (each page can have a different watermark). |
| WatermarkFontName | The font to use for a watermark. |
| CalendarShow | .T. to display an overlay calendar. |
| CalendarHeight | The height of the calendar (there are also CalendarWidth, CalendarTop, and CalendarLeft properties). |

As of this writing, most of these properties aren't exposed in the MindsEyeReportEngine VFP class, so you have to address the oCanvas member of the class, which contains a reference to the MERE control, directly. (Rich is planning on adding properties to the class which will simply pass them through to the MERE control). One complication is that the MERE isn't instantiated until you call the ReportFormPreview method, so you can't set MERE properties before running a report. Fortunately, since this is a VFP class, we can easily subclass it to provide the behavior we want.

I created a subclass of MindsEyeReportEngine called SFMEReportEngine (in SFMERE.VCX). I added a custom method called SetupEngine with the following code:

```
lparameters tnLeft, ;
  tnTop, ;
  tnWidth, ;
  tnHeight
local lnLeft, ;
  lnTop, ;
  lnWidth, ;
  lnHeight
if vartype(tnLeft) = 'N'
  lnLeft = tnLeft
else
  lnLeft = 0
endif vartype(tnLeft) = 'N'
if vartype(tnTop) = 'N'
  lnTop = tnTop
else
  lnTop = 0
endif vartype(tnTop) = 'N'
if vartype(tnWidth) = 'N'
  lnWidth = tnWidth
else
 lnWidth  = _screen.Width - 7
endif vartype(tnWidth) = 'N'
if vartype(tnHeight) = 'N'
  lnHeight = tnHeight
else
  lnHeight = _screen.Height - 7
endif vartype(tnHeight) = 'N'
if vartype(This.oReportPreviewForm) <> 'O'
  This.oReportPreviewForm = ;
    newobject('SFMEReportForm', 'SFMERE.VCX', '', This)
  with This.oReportPreviewForm
    if not empty(This.cReportPreviewFormCaption)
      .Caption = This.cReportPreviewFormCaption
    endif not empty(This.cReportPreviewFormCaption)
    .Left   = lnLeft
    .Top    = lnTop
    .Width  = lnWidth
    .Height = lnHeight
    .Resize()
  endwith
endif vartype(This.oReportPreviewForm) <> 'O'
This.oCanvas = This.oReportPreviewForm.oReportEngine
```

This code instantiates another class in SFMERE.VCX called SFMEReportForm that's a simplified version of the MindsEyeReportPreviewForm class MindsEyeReportEngine uses to host the MERE. It then

sets the size and position of the form to either the passed values or default values if none are passed, and sets the oCanvas property to the MERE object on the form.

I also changed the values of a few properties. lCloseTablesAfterReport is .T. so any tables opened by the report are closed after the report is run. lDisplayExecutionTime and lShowReportPreviewToolbar are .F. so I don't have to set these in code every time I use the class.

Using SFMEReportEngine, you can set properties of the MERE before calling the ReportFormPreview method. Here's an example, taken from Orders2.PRG, which removes the grid button, help button, and page size control from the toolbar and asks you for the background color of the MERE before displaying the report:

```
#if version(5) >= 700
local loRE as SFMEReportEngine of SFMERE
#endif
set classlib to SFMERE
loRE = createobject('SFMEReportEngine')
loRE.cReportPreviewFormCaption = 'Customer Orders'
loRE.SetupEngine()
loRE.oCanvas.DisplayGridButton = .F.
loRE.oCanvas.DisplayHelpButton = .F.
loRE.oCanvas.DisplayPageSize   = .F.
loRE.oCanvas.nBackgroundColor  = getcolor()
loRE.ReportFormPreview('orders.frx', 2)
```

There are a few other methods you may want to call besides ReportFormPreview. ReportToPrinterPrompt prints the specified report (although there's not much advantage to doing that over just using REPORT FORM TO PRINT). SaveToPDF does what its name implies: creating a PDF file from the report without having to display it first and click on a button. The following code (OrdersPDF.PRG) creates Orders.PDF without displaying any UI:

```
#if version(5) >= 700
local loRE as SFMEReportEngine of SFMERE
#endif
set classlib to SFMERE
loRE = createobject('SFMEReportEngine')
loRE.SetupEngine()
loRE.oCanvas.PDFDisplayConfirmationAfterCreate = .F.
loRE.ReportToPDF('orders.frx', 'orders.pdf', .T.)
```

AppendReport is an interesting method: it adds another report to the MERE. This allows you to combine two or more reports in the same preview window, printed report, or PDF file. The reports can even use different orientations; one could use portrait while the other uses landscape. Here's some code, taken from TwoReports.PRG, which displays both Orders.FRX and Customers.FRX in the same preview window.

```
#if version(5) >= 700
local loRE as SFMEReportEngine of SFMERE
#endif
set classlib to SFMERE
loRE = createobject('SFMEReportEngine')
loRE.cReportPreviewFormCaption = 'Customer Orders'
loRE.SetupEngine()
loRE.ReportFormPreview('orders.frx', 2)
loRE.AppendReport('customers.frx')
```

### Using the Rendering Engine

Although the most common use for the MERE is to print FRX reports, because it's actually a generic rendering engine, you can do a lot more with it if you're willing to write some code. For example, the AddLine, AddPicture, AddShape, and AddText methods add line, picture, shape, and text objects to a page, respectively. Calling these and other methods allow you to build your own output without using an FRX. Of course, you're responsible for positioning the objects correctly, so it can be a fair bit of work, but in exchange, you have complete control over how the report looks. Some of the things you can do with MERE used this way are:

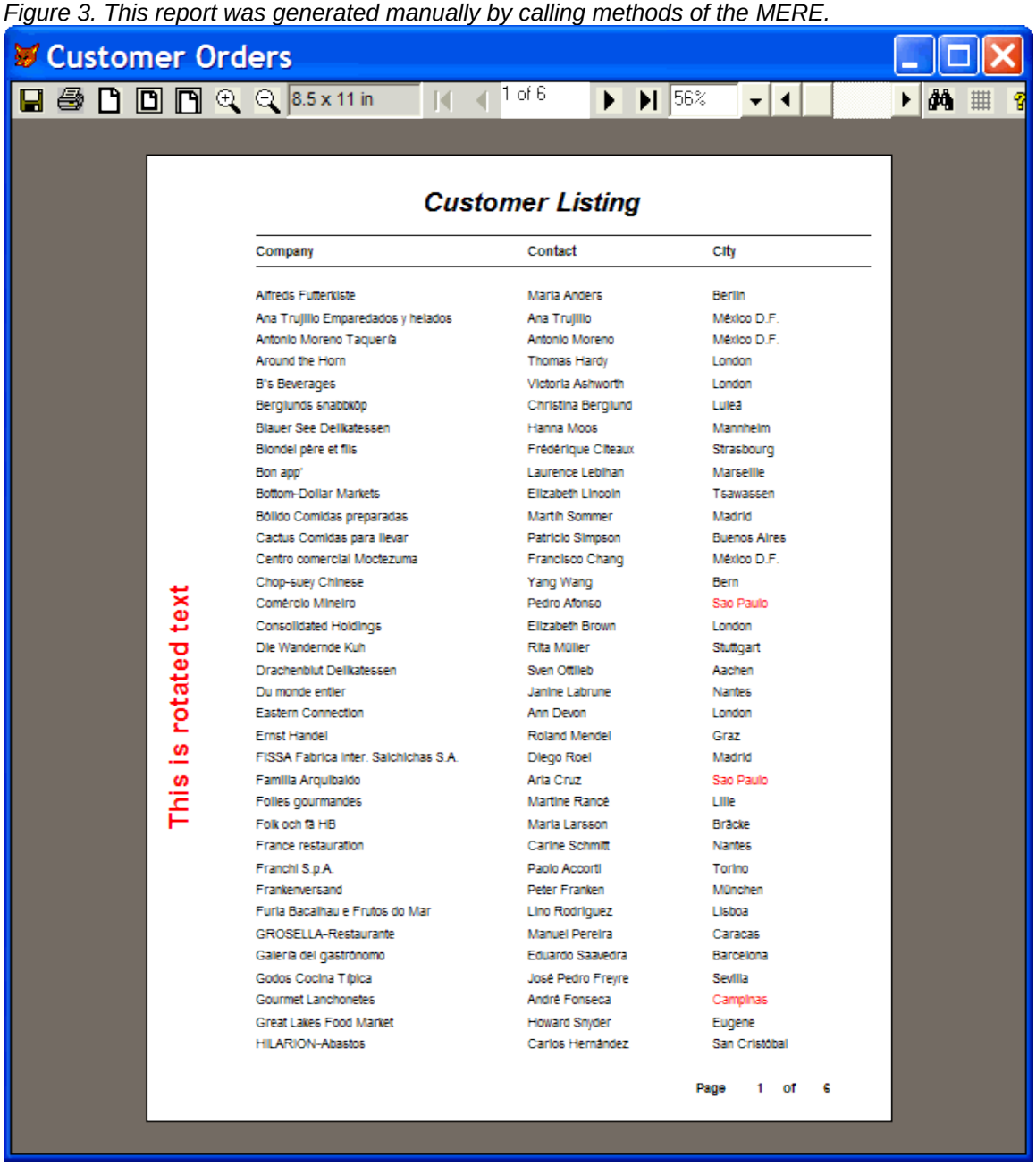
- Create reports with multiple detail bands. Crystal Reports has a "subreport" feature that allows you to embed one report within another. This is often used to create reports with multiple detail bands, such as showing both credit notes and invoices for a customer. You can fake this in VFP (see Cathy Pountney's book for details) but it's very cumbersome to do. With the MERE, you can output whatever you want, including different types of details.
- Support two-pass reporting. Because VFP's report writer only does a single-pass through the data and report layout, it doesn't support things that require two or more passes, such as printing percentages of totals or even printing "Page x of y". Again, you can fake it in VFP by calculating totals in advance or running the report twice (even VFP 8, which provides a _PAGETOTAL system variable to support "Page x of y" output, determines the total page count by running the report twice). However, since you can address each page in the MERE in a similar manner to addressing elements in an array, after rendering a report, you can go back to earlier pages and add additional content like percentages.
- Include graphs in a report. Adding a graph to a report requires using the MSGraph ActiveX control, a cursor with a General field the MSGraph control is bound to, and some fancy data manipulation. The MERE, on the other hand, has built-in charting properties and methods.

Render.PRG results in the customer listing shown in Figure 3. We won't look at all of the code for space reasons, but here is some representative code:

```
with loRE.oCanvas
  scan

* If we're at the bottom of a page, create a new page and
* print the fixed information.

    if lnVPos > 9.5
      if lnPages > 1
        .NewPage()
      endif lnPages > 1
      lnPages = lnPages + 1

* Page header and "sidebar" text.

* AddText parameters: cName, cText, nHPos, nVPos,
*    [nWidth], [nHeight], [cFontName], [nFontSize],
*    [nFontForeColor], [nFontBackColor], [cFontStyle],
*    [nRotationAngle], [nBackStyle], [nAlignment],
*    [cSubType]
      .AddText('', 'Customer Listing',     2.90, 0.50, ;
        4.00, 0.50, , 24, , , 'BI')
      .AddText('', 'This is rotated text', 0.25, 7.50, ;
        4.00, 0.50, , 24, cnCOLOR_RED, , 'B', 900)

* Footer.

      .AddText('', 'Page', 6.00, 10.50, 4.00, 0.25, , , ;
        , ,'B')
* AddReportField parameters: cName, cExpression, nHPos,
*    nVPos, [nWidth], [nHeight], [cPicture], [cDataType],
*    [cFontName], [nFontSize], [nFontForeColor],
*    [nFontBackColor], [cFontStyle], [nRotationAngle],
*    [nBackStyle], [nAlignment], [cSubType]
      .AddReportField('', '_PAGENO', 6.50, 10.50, 0.50, ;
        0.25, '99', 'N', , , , ,'B', , , ;
        cnALIGNMENT_RIGHT)
      .AddText('', 'of', 7.00, 10.50, 0.50,.25, , , , ;
        ,'B')
      .AddReportField('', '_PAGETOTAL', 7.25, 10.50, ;
        0.50, 0.25, '99', 'N', , , , ,'B', , , ;
        cnALIGNMENT_RIGHT)
      lnLines = 1
    endif lnVPos > 9.5
```

```
* Print the columns we want for each record.

    lnVPos = lnTopMargin + lnLines * 0.25
    .AddText('', alltrim(COMPANY), 1.00, lnVPos, 3.00, ;
      0.25)
    .AddText('', alltrim(CONTACT), 4.10, lnVPos, 2.00, ;
      0.25)
    .AddText('', alltrim(CITY),    6.20, lnVPos, 2.00, ;
      0.25, , , iif(COUNTRY = 'Brazil', cnCOLOR_RED, ;
      cnCOLOR_BLACK))
    lnLines = lnLines + 1
  endscan
endwith
```

*Figure 3. This report was generated manually by calling methods of the MERE.*

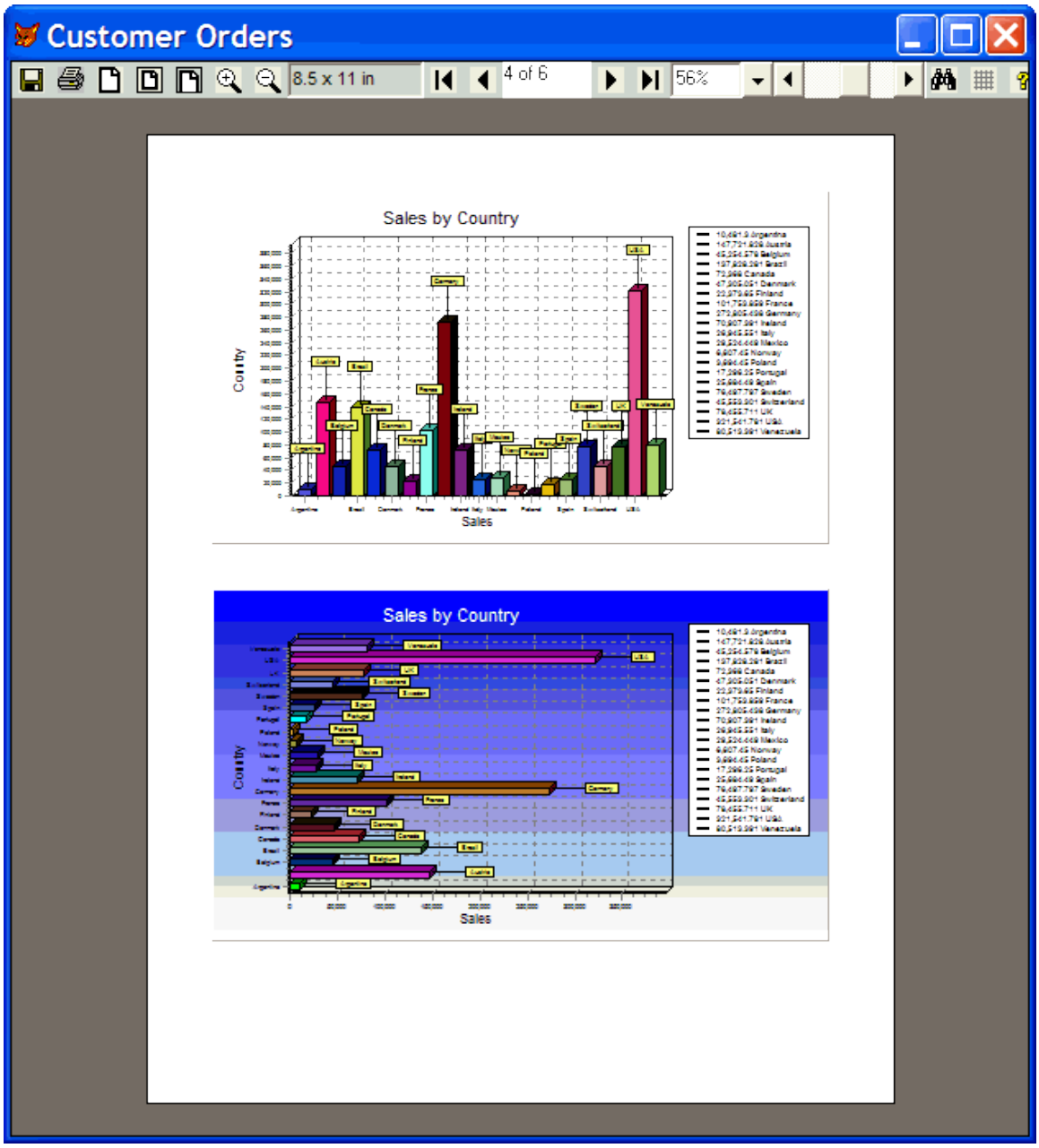


As you can see, every object has to be placed on the page manually, and you're responsible for pagination. However, there are several interesting things to note about this report. First, MERE supports rotated text. Second, note that certain cities appear in red. That's because of the IIF() statement near the end of the code. This acts like a Print When clause in the VFP Report Designer, but has a lot more flexibility, because you can control more than just whether a field prints or not. In this case, the city for Brazilian customers appears in red and in black for other countries.

Render.PRG has additional code that adds five different graphs of total sales by customer country to the last few pages of the report. Figure 4 shows two of the graphs. Here is some of the code that generated these graphs:

```
with loRE.oCanvas
* AddChart parameters: cName, nChartType, nHPos, nVPos,
*    nHeight, nWidth
  .AddChart('', cnCHART_TYPE_BAR, 0.50, 0.50, 7.00, 4.00)
  .ChartXAxisTitle = 'Sales'
  .ChartYAxisTitle = 'Country'
  .ChartTitle      = 'Sales by Country'
  scan
    lnRedPen   = rand() * 255
    lnGreenPen = rand() * 255
    lnBluePen  = rand() * 255
    .ChartAddXYData(TOTSALES, 0, trim(COUNTRY), ;
      rgb(lnRedPen, lnGreenPen, lnBluePen))
  endscan

  .AddChart('', cnCHART_TYPE_COLUMN, 0.50, 5.00, 7.00, ;
    4.00)
  .ChartXAxisTitle         = 'Sales'
  .ChartYAxisTitle         = 'Country'
  .ChartTitle              = 'Sales by Country'
  .ChartTitleFontColor     = cnCOLOR_WHITE
  .ChartGradientVisible    = .T.
  .ChartGradientStartColor = cnCOLOR_WHITE
  .ChartGradientEndColor   = cnCOLOR_BLUE
  scan
    lnRedPen   = rand() * 255
    lnGreenPen = rand() * 255
    lnBluePen  = rand() * 255
    .ChartAddXYData(TOTSALES, 0, trim(COUNTRY), ;
      rgb(lnRedPen, lnGreenPen, lnBluePen))
  endscan
endwith
```

*Figure 4. You can easily create graphs using methods of the MERE.*




## Pricing and Licensing

MERE is available in two versions: Standard and Professional. The Standard version, which doesn't include PDF output or support for charts or barcodes, is $295, while the Professional version is $695. You can distribute it royalty-free with your application, with only one restriction: you cannot distribute it as part of a product sold to other software developers (for obvious reasons). A demo version is available for download at http://mindseyeinc.com/ReportEngine/. It's a full-working version but adds a "demo version" watermark to every page of the report. Download and install the demo version if you want to try out the sample code included in this article.

## Summary

MERE is an incredible new tool that will make you forget the VFP Report Preview window. It provides more features than VFP, is more flexible and easier to control, and even produces PDF output without requiring any other components. MERE is a work-in-progress; Rich seems to add new features every day. It

also has some relatively minor bugs and a few rough spots (including the help file, which doesn't currently document all PEMs and needs considerable more work). However, even with these issues, it's still a solid product, and I highly recommend it.

*Doug Hennig is a partner with Stonefield Systems Group Inc. He is the author of the award-winning Stonefield Database Toolkit (SDT) and Stonefield Query, author of the CursorAdapter and DataEnvironment builders that come with VFP 8, and co-author of "What's New in Visual FoxPro 8.0", "What's New in Visual FoxPro 7.0", and "The Hacker's Guide to Visual FoxPro 7.0", from Hentzenwerke Publishing. Doug has spoken at every Microsoft FoxPro Developers Conference (DevCon) since 1997 and at user groups and developer conferences all over North America. He is a Microsoft Most Valuable Professional (MVP) and Certified Professional (MCP). Web: www.stonefield.com and www.stonefieldquery.com Email: dhennig@stonefield.com*