Virtual vsView.vbx Printer Demo

The Problem:

Typically, the vsView control allows the programmer to pass a single string variable containing the complete table, and then the control handles all formatting, headers, page breaks, etc.

However, because the table is stored as a string variable, it is limited to a total size of 64K. Additionally, the usual method limits the document to a total of 100 pages, and requires a very considerable amount of time to process the entire table at one time (20 minutes or so for 20 pages, depending on the amount of information to be printed).

The Solution:

This demonstration utilizes a virtual-memory approach that prints the document one page at a time, building the table one line at a time, thereby overcoming the 64K and 100-page limitations. Additionally, the procedure is very fast, requiring only about 3-5 seconds per page.

Description of Code:

The query results are stored in the global dynaset gDS. The bookmarks for each record in the dynaset are stored in the global array gBookMarks().

At the start of the document, the procedure sets the page formatting information, headers, borders, and on the first page, the titles, if any have been specified.

At the start of each page (current page stored in CurPage), the procedure prints the formatted column labels (TableHeader), and then calls GetNextLine to retrieve the next record in the dynaset.

GetNextLine is called after each line is printed until the control invokes the NextPage event (or when no more records exist). The document is then terminated (vPrint.Action = 6).

An array is maintained of the starting row number of each page: PageRow(). The user can control the current page displayed by use of the btnMove buttons (First,Next,etc). Upon selection of any page command the appropriate data for that page are retrieved and displayed, one record at a time.

Limitations:

One limitation to this method is that the total number of pages in a given report is not known until the final page is drawn (i.e., the user has paged through each page). One 'solution' to this is to determine the estimated number of pages (EstNumPages) by dividing the number of records in the dataset by the number of rows printed on the first page of the document. Notice that in the demo, the original estimate is 4 pages, which changes to 3 pages when the last page is printed (I quess I need to tweak the EstNumPages procedure some more - the parent app allows the user to specify up to 3 lines for the titles, each of which may be of different fonts and sizes).

New Problem:

One additional problem, which I am desperately trying to solve, is the fact that Headers and Footers in multiple-page table documents default to the width of the table. If you run the demo you will notice that the headers & footers on pages 1 and 2 are the width of the table, but on page 3 (the final page) the header & footer print correctly. I cannot explain why this occurs, and have tried every workaround I have been able to conceive. It must have something to do with an internal End of Page routine, since that event occurs on pages 1 & 2 but not on page 3 of the demo (which does not trigger an End of Page event).

Basically, the problem is as follows:

- On single-page tables, the header & footer print correctly.
- On the last page of a document, the header & footer print correctly.
- However, on all but the last page, the header and footer will be sized to the width of the table.
- This occurs even though each page in the Virtual Demo is a separate and supposedly unrelated entity (what gives?).

The demo was developed with vsView version 1.03. Interestingly, version 1.0 had the opposite problem: the header & footer printed correctly on the first page, but defaulted to the width of the table on subsequent pages. I inquired to VideoSoft about the problem, and they wrote back to state that it had been fixed in 1.03.

The demo requires:

VB 3.0
crystal.mdb (supplied with VB 3.0 pro)
vsView.vbx 1.03, although 1.0 may work

I've stripped out just about everything irrelevant to the preview function, but I think you'll find that the source code could easily be modified to display the results of any query for any database (which is what the parent application is for). As such, does away with the need for an intermediary grid control (which can be confusing for novice users).

Included in the code is an illustration of the graphics distribution methods described in a recent issue of VB Programmers Journal (notice the interaction between the ArtForm and the CheckMoveBtns procedure). I wish I had used this in the parent application!

The Plea:

If you have any questions or comments, please forward them.
If you find the demo useful, please let me know.
If you can solve the header problem, then I beg of you - please share the solution!

Brian C. Hayes
CIS 74653,1760