Using The Borland Visual Solutions Pack

by Jeroen W Pluimers

Delphi has enormous potential for using components. Not only does it have its own sophisticated VCL-based components, but it also supports many of the VBX controls widely available today.

An introduction to what VBXs can do for you is the Borland Visual Solutions Pack (BVSP). Although not expensive, it contains a number of useful controls. Note that when considering other VBXs you do need to remember that Delphi only supports *Version 1.0* VBXs – if in any doubt, ask your retailer or (perhaps even better) the manufacturer.

Installation

Installing VBX controls in Delphi is easy. For the sake of simplicity, I'll assume you are using the default directory structure. Thus, you have a directory `C:\BVSP` which contains the BVSP files. Also, create a subdirectory `C:\BVSP\DELPHI`, where we will store the Delphi component wrappers around the VBXs.

Let's start installing. From the Delphi menu bar, choose `Options | Install Components`. Figure 1 shows the dialog box which is displayed, listing the components that have already been installed. Click the `VBX` button to enter a file selection dialog. Navigate to wherever the VBXs are installed (usually `WINDOWS\SYSTEM`) and select the VBX file you want to install.

After selecting the file, you end up in the dialog shown in Figure 2. The defaults are filled in, so you can just press OK, but you can also change the name of the unit file or the classnames of the controls in the VBX. In this case, I've decided to place the Delphi wrapper unit files in `C:\BVSP\DELPHI`. Delphi automatically generates a Pascal source unit containing a VCL component wrapper around the VBX.

A VBX can contain more than one component – just like regular Delphi units, which can contain more than one component as well.

Although you can install all VBX files from the BVSP, you do not need to do so. Some of the VBX components duplicate functionality that is already included in Delphi. For instance, the `SQC.VBX` (containing Database Controls) is completely covered by Delphi. Also you do not need the `SAXTABS.VBX` as the Delphi Tabs and Notebook components give better functionality.

A summary of the VBXs in the BVSP which are of use to Delphi developers is shown in Figure 4. The column with glyphs shows you the bitmaps which will appear on the VBX page of the component palette. The classname is the hint text you see when you move the mouse over the corresponding glyph. The VBX filename is included so you can easily select the VBX files you need.

Using The BVSP VBXs

The BVSP can be split conceptually into two categories. The first group consists of VBXs that encapsulate alone, or in combination with others, large parts of an application. The second group consists of gadgets: VBXs that perform only a tiny part of an application, but give it a specific look and feel.

The charting, spreadsheet and word-processing VBXs are clearly part of the first group. Depending on its usage also the SaxComm VBX can be added to it. All the other VBXs are part of the second group. You will often need to do a lot of coding yourself to create a fully functional application around these components.

With the first group of VBXs, it is possible to create complete working applications with almost no code. For instance, using the `TX4VB.VBX` and only a few lines of code, you can create a working word-processor that can import and export rich text format (RTF) files, with multiple fonts etc.

Gadgets

Not all components from the second group need much programming. For instance, the animated button can be used to quickly create a multi-state button. The card deck is ideal for setting up a card game, (it's a pity the decks lack animation like Solitaire and Hearts). With a good playing strategy, you could write your own black jack game! Combined with the clock, you can stress the player by limiting playing time.

The dice can be easily configured. With `AutoSize` disabled and a lot of colours on the dice sides, you could write a program to teach counting to children.

A combination of gauges, sliders and spin buttons could be used to wrap up the user interface for a scientific program. Then the marquee control can be used to show floating text on the screen (although its performance is unfortunately not very good).

Fully Fledged Controls

This group contains the really useful VBXs. Although most of the controls are not the most recent versions, you get a good impression of what is possible with them. If you make heavy use of any of the more complete controls, I'd recommended you buy the full current version.

For instance, the SAX communications control lacks certain protocols (like Z-Modem) that are used very widely nowadays. However, it is a good starting point if you want to see what communications could do for your application.

Another useful combination is the charting control and the spreadsheet control. This way, you could show a graphical representation of the data a user enters. Remember, though, that Delphi itself has a more powerful `ChartFX` VBX control.

Figure 3 shows a sample application written using the BVSP. With seven components and about 90 lines of hand-written code, I ended up with a complete word processor using the RTF file format. It supports multiple fonts and pages, text with attributes, paragraphs, search & replace, etc.

A large part of the code (almost 20 lines) is to make sure the `TextControl` and its bars resize within the Form. In contrast with Delphi's native components, the VBX controls lack an alignment property, so you have to do the aligning yourself.

The rest of the code is for the `File|Open` and `File|Save/SaveAs` logic. Only a tiny bit of code is needed to link the menu to the actions – one line per menu action suffices.

The resulting application .EXE file itself is only about 200Kb in size. The additional files are much larger: the `BIVBX11.DLL` (see next section) is about 80Kb and the `TX4VB.VBX` and its support files add up to 240Kb, giving just over half a megabyte in total.

The source for this example will be on the free disk with Issue 2 of The Delphi Magazine.

Distribution

When distributing an application you need to pay special attention if it uses VBXs. You will need to distribute the VBX files with your application and be careful to ship the correct support files. Also, you will need to include the BIVBX11.DLL, which is the Borland support DLL that interfaces between 16-bit applications and VBX files.

The reason behind the complexity is twofold. First of all, VBX files are external DLLs that in turn can use other external files. Second, VBXs need to have a means to distinguish between design-time

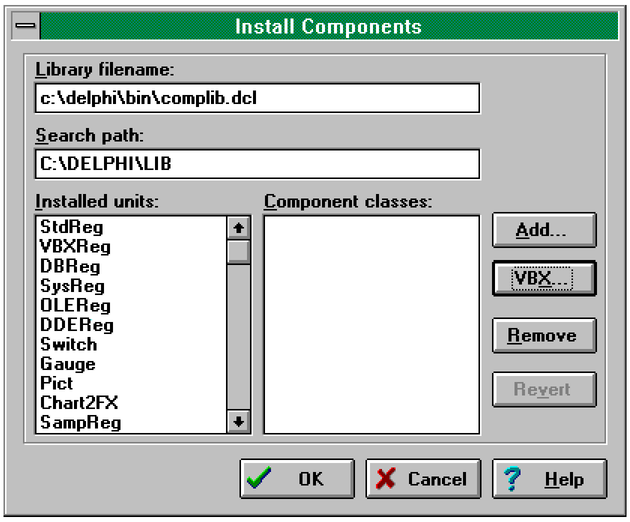


Figure 1 Installing a new VBX into Delphi

Install Components

Library filename: c:\delphi\bin\complib.dcl

Search path: C:\DELPHI\LIB

Installed units: StdReg, VBXReg, DBReg, SysReg, OLEReg, DDEReg, Switch, Gauge, Pict, Chart2FX, SampReg

Component classes:

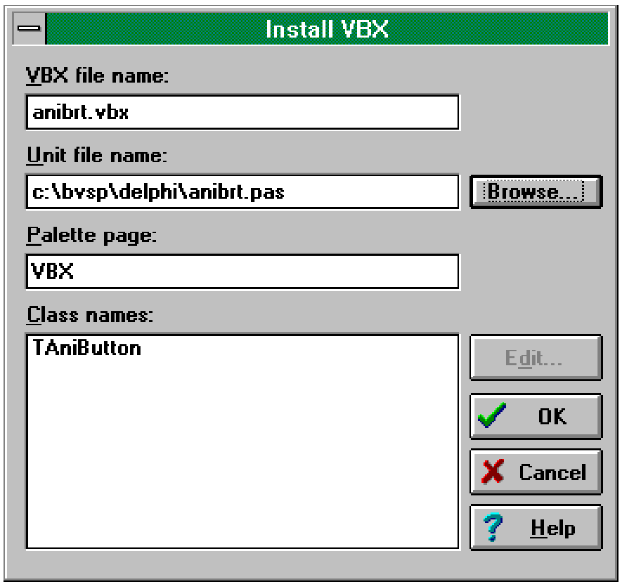


Figure 2 Specifying a Delphi Unit file name and Class names for a new VBX

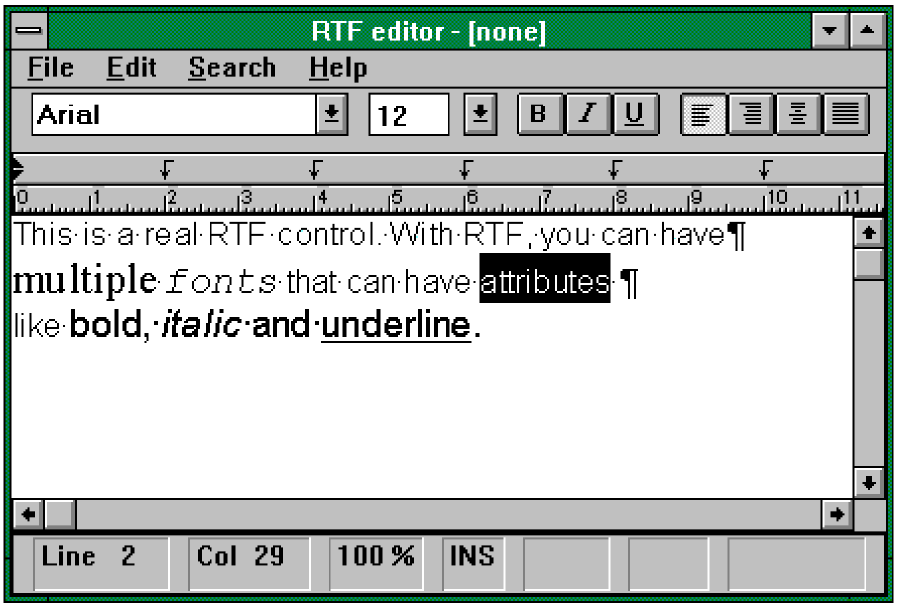
Install VBX

VBX file name: anibrt.vbx

Unit file name: c:\bvsp\delphi\anibrt.pas

Palette page: VBX

Class names: TAniButton



RTF editor - [none]

This·is·a·real·RTF·control.·With·RTF,·you·can·have¶
multiple·fonts·that·can·have·attributes·¶
like·**bold**,·*italic*·and·underline.

Line 2 | Col 29 | 100 % | INS

Figure 3 A fully functional rich text format word processor built in Delphi using controls from the Visual Solutions Pack

and run-time behaviour (otherwise anyone with a VBX could use it to develop a new application, without purchasing it).

The technique behind this is called licensing. Because the VBX standard was not well designed, it lacked standard support for licensing. As a result, most VBX vendors invented their own licensing mechanism. Mostly, they need special design-time files – which are called licensing files – to be present in the Windows System directory. Another way is to have separate VBX files for design-time and run-time.

`KNIFE.VBX` is an example of this approach – you should be very careful to copy `KNIFERUN.VBX` onto your installation disks and *rename it* to `KNIFE.VBX` at installation time.

All the other installable files are explained fully in the BVSP documentation. The file `REDIST.TXT` contains more information.

Jeroen Pluimers has been a Pascal programmer since 1983. He lives and works in The Netherlands and may be contacted by email as jeroenp@dragons.nest.nl or on CompuServe as 100013,1443

Figure 4: Useful VBXs from the Visual Solutions Pack

Glyph	VBX File	Class Name	Functionality
	ANIBRT.VBX	AniButton	Animated button with multiple bitmaps. Different frame states allow simulation of multistate buttons.
	KNIFE.VBX	PicBuf	Picture buffer for editing and showing bitmaps.
	MHAL200.VBX	MhIAlarm	Alarm clock with alarm interval and sound.
	MHCD200.VBX	MhCardDeck	Playing card that mimics the cards used in Solitaire and Hearts. It supports non-animated card-backs only.
	MHCL200.VBX	MhClock	Digital or analogue clock showing current time, or time offset to a specific value.
	MHDC200.VBX	MhDice	Playing dice with pictures showing top, left and right sides of dice. Properties for colours and bitmaps.
	MHGA200.VBX	Mhgauge	Gauges that can be horizontal, vertical, circular with pointing needles.
	MHMQ200.VBX	MhMarque	Label-like component with moving caption and moving bitmaps. Attracts attention – ideal for running demos.
	MHSL200.VBX	MhSlide	Slider control useful for simulating audio and industrial equipment.
	MHSN200.VBX	MhSpin	Spinbutton with embedded value component. Buttons are either shown horizontally and vertically.
	SAXCOMM.VBX	Comm	Serial communications with TTY and ANSI emulation. Supports X-modem file transfers.
	TKCHART.VBX	Chart	Chart drawing component. Both application supplied data and database supplied data.
	TX4VB.VBX	TextControl	Rich text component for editing texts with multiple fonts. Has hooks for the ruler, buttonbar and statusbar.
	TX4VB.VBX	TXRuler	Ruler component. For showing positional information of the TextControl.
	TX4VB.VBX	TXButtonBar	Speedbar with buttons and comboboxes to change appearance of text in the TextControl.
	TX4VB.VBX	TXStatusBar	Status line showing positional information of the TextControl.
	VTSS.VBX	Sheet	Spreadsheet component. Automatically links to an SSEdit component for editing if it is available.
	VTSS.VBX	SSEdit	Editor portion of the Sheet spreadsheet component.