An Overview of the Delphi 2.0 Windows 95 Common Controls

Diane Rogers

Delphi Product Manager

Introducing Delphi 2.0 for Windows 95 and Windows NT

Delphi 2.0 brings the unique combination of an optimizing native code compiler, scalable database architecture, and visual development environment to the 32-bit platforms of Windows 95 and Windows NT. Architected to leverage the advantages of 32-bit Windows, Delphi 2.0 introduces performance, reuse, rapid application development and scalability enhancements.

Performance improvements include:

- Optimizing 32-bit compiler for apps that run 3-4 times faster
- True 32-bit Development with full Win95/NT Support
- Shared backend with C++
- Higher performance 32-bit BDE
- Ability to build stand alone executables and reuseable DLLs that are up to 25% smaller

State of the art reuse with

- Visual Form Inheritance
- Full OLE control (OCX) support
- OLE automation controller, server

Tools for rapid application development from prototype to production:

- Over 90 32-bit Visual Components
- Drag and Drop Database Development
- Object Repository
- Improved error messages, diagnostics

Robust database development features:

- Scalable Data Dictionary
- New high level business model support
- New data-aware controls, Multi Object Grid
- New Fast Filters, Smart lookups, Cached updates

Leveraging Windows 95 and NT

Together, these features make a comprehensive package for the development of high performance Client/Server, database, and multi-purpose windows applications. For applications specifically targeted to meet the Windows 95 User Interface standards, Delphi includes the complete suite of Windows 95 components to support features like rich text editing, Windows 95 style notebook tabs, progress bars, OLE controls (OCXs) and so on. Developers simply add these components to their applications from the component palette just as they would any other Delphi component. Delphi offers true 32-bit development with complete access to the Windows 95 and NT APIs. Programmers can exploit the power of multi-threading, MAPI, Plug'n'Play, Unicode and make direct calls to any Windows 95 or NT API.

OCX support

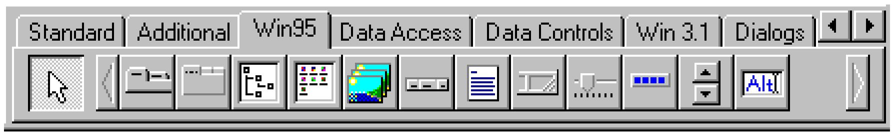
Delphi 2.0 fully supports the emerging 32-bit OLE Custom Control (OCX) standard for add-in software components. Developers can add third-party or custom-made OCXs to the component palette quickly and easily, all OCX properties immediately appear in the object inspector. Because Delphi is object oriented, you can customize OCX's via inheritance. Placing an OLE control in an application is like placing a button or graph into a form. A series of versatile sample OCX components for graphing, charting, spreadsheet operations, and spell-checking OCXs are included for immediate use in applications.

OLE Automation Server and controller support

New automation support lets programmers use OLE automation to create or control other applications such as Microsoft Word and Excel, Lotus 1-2-3, dBASE, and Paradox. Because Delphi 2.0 can create high performance OLE automation controllers and OLE automation servers, Delphi 2.0 will allow developers to create partitioned applications easily with Network OLE when that technology becomes available. Delphi 2.0 fully supports in-process and out-of-process local OLE automation servers and is fully compatible with the forthcoming Network OLE. Use Delphi 2.0 with the automation technology included in Visual Basic 4.0 and you get the added advantage of faster performance due to our optimizing native code compiler.

Rapid Windows 95 UI Development

Delphi 2.0's Visual Component Library offers approximately 100 reuseable components for rapid application development. With a suite of new Windows 95 components, you can design elegant Windows 95 logo compliant applications. To build an application, simply drag and drop components from the palette onto the form. The Win95 user interface common controls were introduced with the release of the Microsoft Windows 95. Declared at the API level, the Win95 control behavior is sufaced in Delphi with easy to use components:

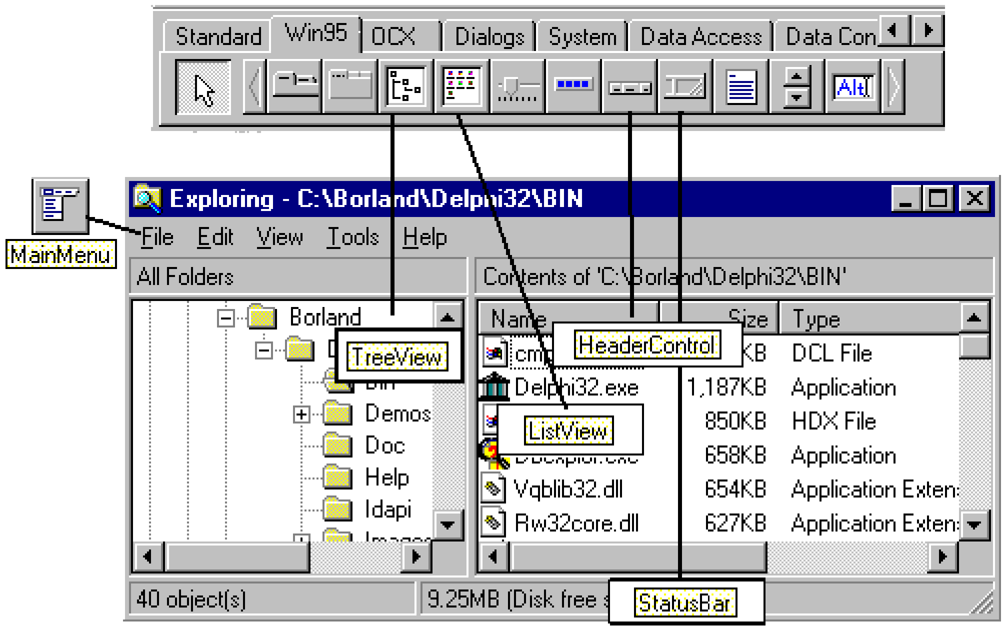


As you move your mouse over any of the components, a Tooltip appears to descibe that control. The Windows 95 UI controls available in Delphi 2.0 are:

TabbedControl	This component is a tab set, which functions similarly to a TTabSet
PageControl	A page set which is used to make a multiple page dialog box.
TreeView	Displays lists of objects as an indented outline
ListView	Visual display of a list in columns.
ImageList	Controls a list of images for display in treeviews or listboxes.
HeaderControl	Movable headers, similar to a THeader component.
RichEdit	Rich Text Format memo control.
StatusBar	Area to post the status of actions at the bottom of the screen
TrackBar	Slider and ticked bar control.
ProgressBar	Tracks progress of a procedure, similar to a TGauge.
UpDown	Up and down arrow buttons to increment and decrement values
HotKey	Attaches a hot key to any component

Where to use the controls

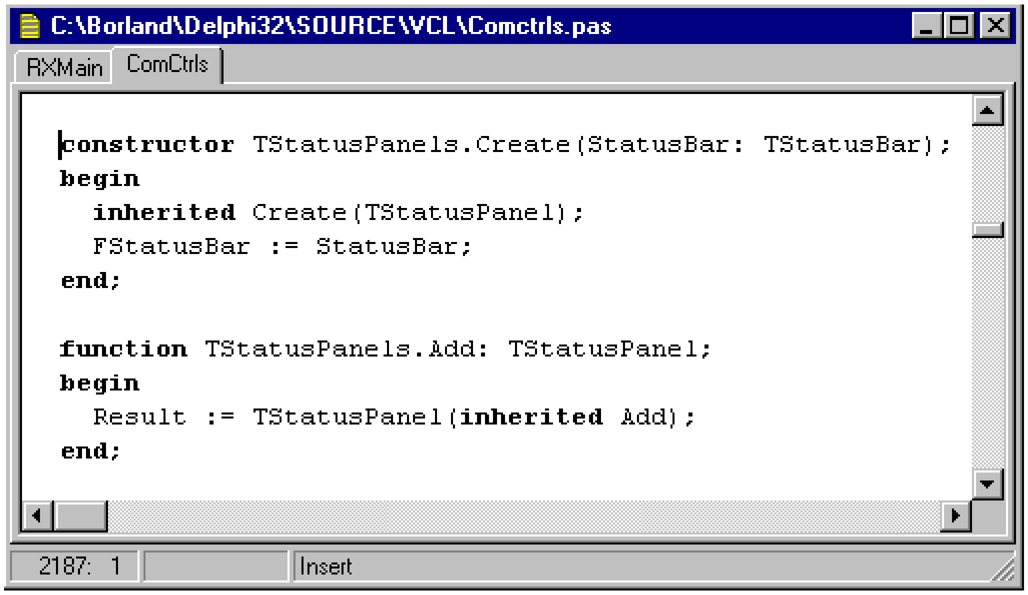
The most familar example of an application that uses the new Win95 UI Controls is the Windows Explorer. It's easy to identify the Delphi components you could use to build an "Explorer" like application.



Object Oriented programming with Win95 Controls

All Delphi components can be customized, subclassed, and reused across applications. The source code for Delphi 2.0's Windows 95 components is in DELPHI32\SOURCE\VCL\COMCTRLS.PAS. To build a Win95 UI compliant application, developers interact with a components' published properties through the visual interface: the *property inspector*. The *events page* of the property inspector helps the application respond to actions from the user or Windows itself. Delphi's comprehensive set of components makes it possible to solve the majority of application demands by working directly with an exisiting objects' properties and events. Where further customization is necessary, developers can implement their own functions, behaviors and properties to a component via inheritance. All of the Win95 controls can be subclassed. To subclass a Win95 component, first identify the functionality you want to add or override then select **Component | New** from the Delphi main menu. Select the parent control from the list of components in Delphi's VCL.

Because Delphi 2.0 ships with full source code for the Win95 controls, it is easy to research the exact behavior of an object. The picture below shows the Delphi Win95 component source code. Delphi's component architecture provides a uniquely extensible environment. Delphi's object oriented ability to customize controls includes the ability to subclass OCXs. To developers, this extensible architecture means that it's easy solve problems and surpass application barriers.



```
constructor TStatusPanels.Create(StatusBar: TStatusBar);
begin
  inherited Create(TStatusPanel);
  FStatusBar := StatusBar;
end;

function TStatusPanels.Add: TStatusPanel;
begin
  Result := TStatusPanel(inherited Add);
end;
```

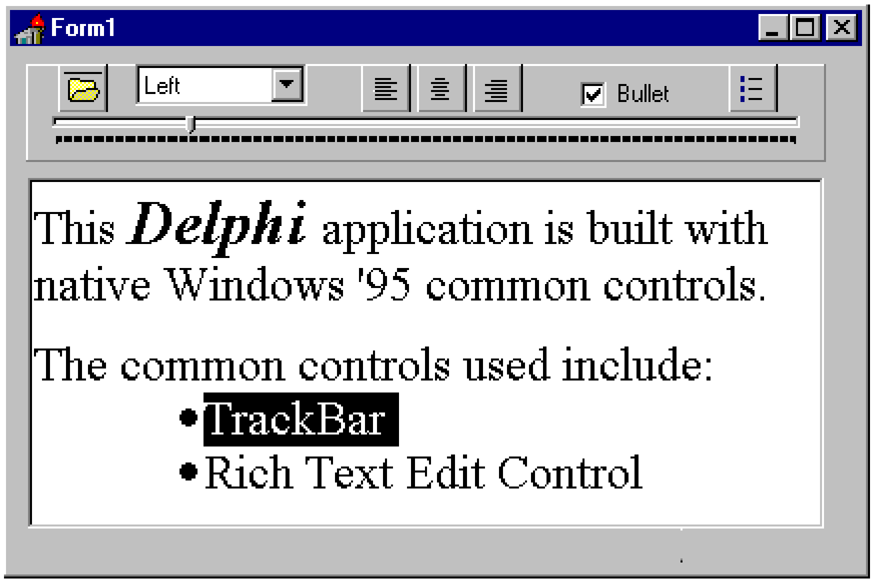
Delphi's Windows 95 Controls

Controls are graphic objects that represent the properties or operations of other objects. Some controls display and allow editing of particular values. Other controls start an associated command. Each control has a unique appearance and operation designed for a specific form of interaction.

Like most elements of the interface, controls provide feedback indicating when they have the input focus and when they are activated. For example, when the user interacts with controls using a mouse, each control indicates its selection upon the mouse down, but does not activate until the user releases the button.

TRichEdit, a Rich Text Format memo control

The application below uses two Windows 95 controls: a *rich text edit* and a *trackbar*.



This ***Delphi*** application is built with native Windows '95 common controls.

The common controls used include:
- TrackBar
- Rich Text Edit Control

By default, the rich text editor supports

- font properties, such as typeface, size, color, bold, and italic format,
- format properties, such as alignment, tabs, indents, and numbering
- Automatic drag and drop of selected text

Using Object.Property

All components in Delphi respond to assignments by name and property. For example, to change the paragraph alignment property of all text in a rich edit control, attach the following code to a radio button toggle:

```
richedit1.paragraph.alignment := TparaAlignment(paright);
```

Using Object.property you can make two objects can interact. The application above uses the trackbar to change the indentation of the rich edit control:

```
richedit1.paragraph.firstindent := trackbar1.position;
```

Building a Data Aware Rich Text editor

If you want to build a rich text control that displays formatted information from a table, simply use the Win95 rich text control as the parent object, and use Component | New to subclass a new control. The control will have two new procedures to read and write information into a blobstream and back out to a TTable. To test the blobstream read and write before you build a control, attach the following code to buttons:

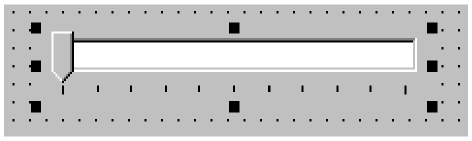
```
procedure TForm1.Button1Click(Sender: TObject);
var
 b: Tblobstream;
begin
 table1.append;
 B:= Tblobstream.create(table1RTFText, bmreadWrite);
 {field and mode to read}
 Richedit1.lines.savetostream(B);
 b.free;
 Table1.post;
end;
procedure TForm1.Button2Click(Sender: TObject);
var
  b: tblobstream;
begin
 B:= Tblobstream.create(table1RTFText, bmread);
 richedit1.lines.loadfromstream(b);
 b.free;
end;
```

TTrackBar

The trackbar used above to set the indent location of the rich text can be used in any application for setting or adjusting values on a continuous range, such as volume or brightness. A trackbar is a control that consists of a bar that defines the extent or range of the adjustment, and an indicator that both shows the current value for the control and provides the means for changing the value.

Trackbars support a number of options. You can set the trackbar orientation as vertical or horizontal, define the length and height of the slide indicator and the slide bar component, define the increments of the trackbar, and whether to display tick marks for the control.

The user moves the slide indicator by dragging to a particular location or clicking in the hot zone area of the bar, which moves the slide indicator directly to that location.



It's properties include:

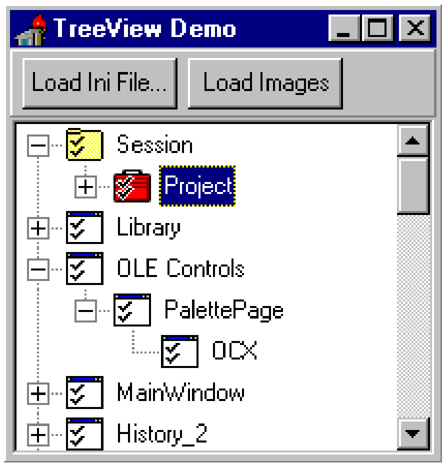
LineSize property	Number of ticks moved using arrow keys.
Max property	Upper range of trackbar.
Min property	Lower range of trackbar.
Orientation property	Horizontal or vertical
PageSize property	Number of ticks moved using page up/down.
Position property	Current value of trackbar.
SelEnd property	End point of selection range.
SelStart property	Start point of selection range.
TickMarks property	Position of tick marks.
TickStyle property	Controls auto/manual tick mark styles.
OnChange event	Occurs as trackbar is moved.

TTreeView a tree outline structure

A tree view control is a special list box control that displays a set of objects as an indented outline based on their logical hierarchical relationship. The control includes buttons that allow the outline to be expanded and collapsed. You can use a tree view control to display the relationship between a set of containers or other hierarchical elements.

You can optionally include icons with the text label of each item in the tree. Different icons can be displayed when the user expands or collapses the item in the tree. In addition, you can also include a graphic, such as a check box, that can be used to reflect state information about the item.

The control also supports drawing lines that define the hierarchical relationship of the items in the list and buttons for expanding and collapsing the outline.

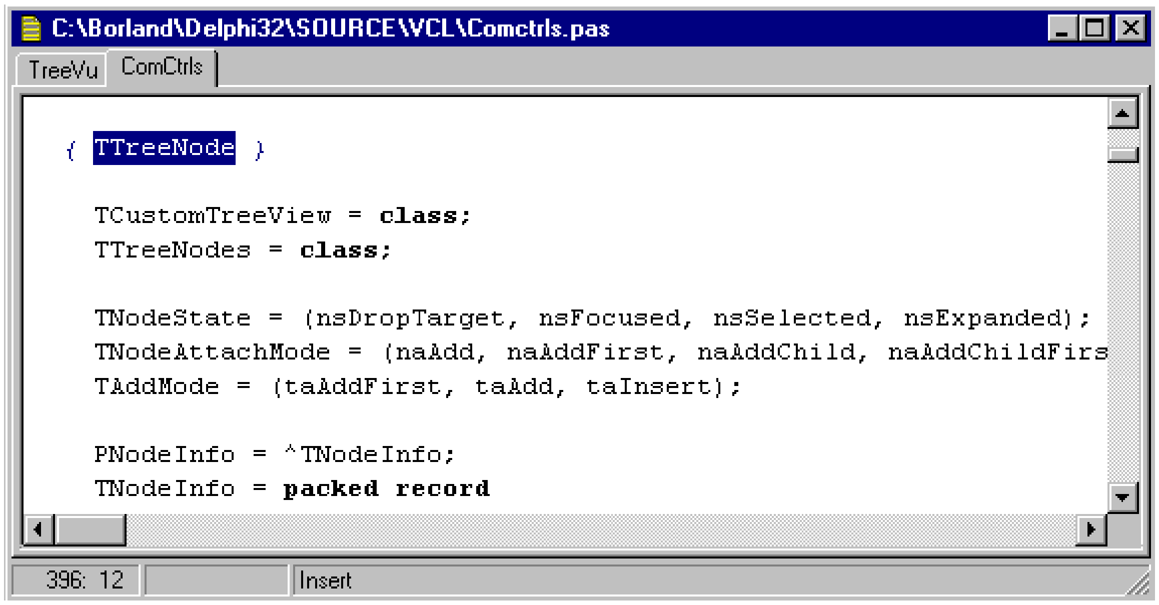


Loading Images to TreeView

Each node of a treeview can be read, assigned a value, and display bitmaps for a variety of states. The application above shows the Delphi32.ini file loaded into a treeview. The first node in the tree "session" has been assigned a static bmp from the imagelist. The second node of the tree "project" has been assigned a dynamic bmp from the imagelist, it will appear only when the second node is selected. Overall, the programmer has ultimate control over loading images and treeview values from tables, and changing the display of a treeview in response to user actions.

TTreenode source

Treeviews are made up of a series of nodes in hierarchical order. Each node of the tree has properties and can respond to events. The image below shows c:\borland\delphi32\source\vcl\comctrls.pas open to the TTREENODE definition.



```
{ TTreeNode }

  TCustomTreeView = class;
  TTreeNodes = class;

  TNodeState = (nsDropTarget, nsFocused, nsSelected, nsExpanded);
  TNodeAttachMode = (naAdd, naAddFirst, naAddChild, naAddChildFirs
  TAddMode = (taAddFirst, taAdd, taInsert);

  PNodeInfo = ^TNodeInfo;
  TNodeInfo = packed record
```

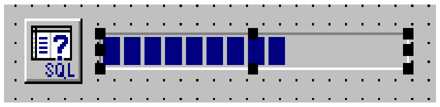
As you can see, Treenodes have three advanced properties (*TNodeState, TNodeAttachMode, TAddMode*) that give you detailed information about a node. Using these properties, you can

- check for the state of a node, i.e. is it selected, is it expanded?
- gather information about a node, i.e. does it have children?

- attach a new node as the first child, the last child, above, below or at the top of a tree.

TProgressBar, Tracks progress of a procedure

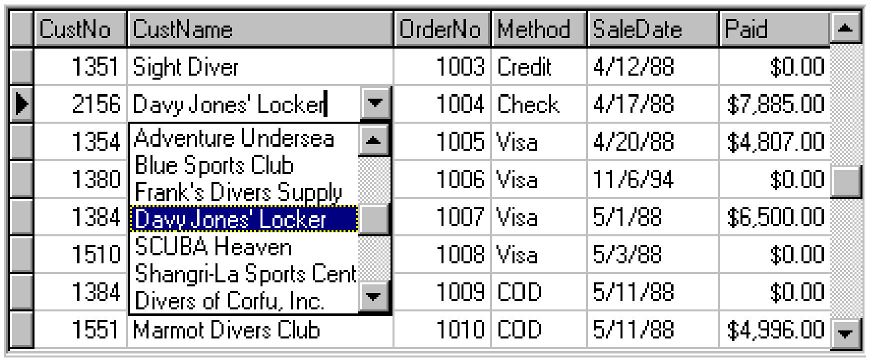
A progress bar is a control you can use to show the percentage of completion of a lengthy operation. It consists of a rectangular bar that “fills” from left to right. You see it all the time in Windows 95 when you copy files in the Windows Explorer. Because a progress indicator only displays information, it is typically noninteractive. Use the control as feedback for long operations or background processes. The control provides visual feedback to the user about the progress of a process.



THeaderControl, Movable headers

The best example of a header control is in the Windows Explorer (new File Manager). Using a header control, you can display a heading above columns of text or numbers. You can divide the control into two or more parts to provide headings for multiple columns. You can align the title elements left, right, or centered. You can configure each part to behave like a command button to support a specific function when the user clicks on it.

Test the Windows Explorer's sort capabilities by displaying the list view in detail mode, and then clicking on one of the headers: "name", "size", "type" or "modified".



CustNo	CustName	OrderNo	Method	SaleDate	Paid
1351	Sight Diver	1003	Credit	4/12/88	$0.00
2156	Davy Jones' Locker	1004	Check	4/17/88	$7,885.00
1354	Adventure Undersea	1005	Visa	4/20/88	$4,807.00
1380	Blue Sports Club Frank's Divers Supply	1006	Visa	11/6/94	$0.00
1384	Davy Jones' Locker	1007	Visa	5/1/88	$6,500.00
1510	SCUBA Heaven	1008	Visa	5/3/88	$0.00
1384	Shangri-La Sports Cent Divers of Corfu, Inc.	1009	COD	5/11/88	$0.00
1551	Marmot Divers Club	1010	COD	5/11/88	$4,996.00

To extend this pushbutton sorting capablity to a Delphi TTable, drop down a header control and attach the following code.

```
procedure TForm1.HeaderCustNo(Sender: TObject);
begin
  table1.indexfieldnames:= 'CustNo';
{sorts the table by customer number when header is clicked}
end;
```

To make the code generic for any field with an index:

```
table1.indexfieldnames:= tcontrol(sender).name;
```

The control also supports the user dragging on the divisions that separate header parts to set the width of each column. As an option, you can support double-clicking on a division as a shortcut to a command that applies to formatting the column, such as automatically sizing the column to the largest value in that column.

TUpDown, Up and Down arrow buttons

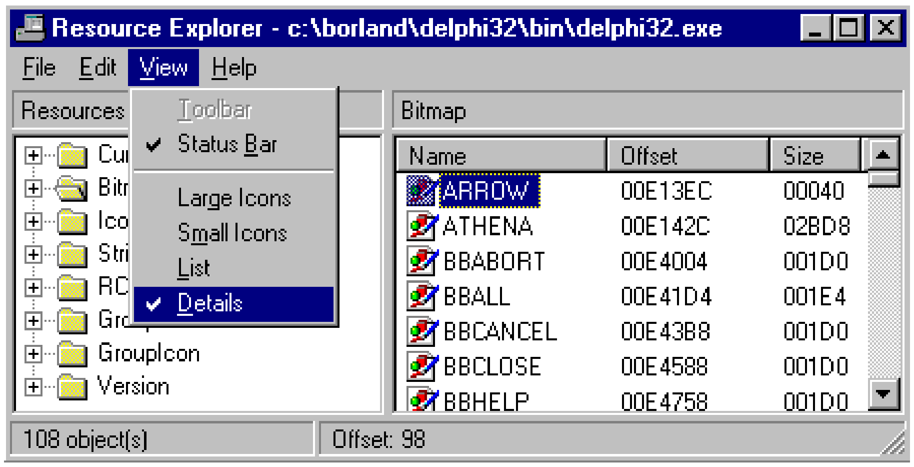
Up Down controls are text boxes that accept a limited set of discrete ordered input values that make up a circular loop. An up down control is a combination of a text box and a special control that incorporates a pair of buttons. Similar to the 16-bit TSpinButton.

When the user clicks on the text box or the buttons, the input focus is set to the text box component of the control. The user can type a text value directly into the control or use the buttons to increment or decrement the value. The unit of change depends on what you define the control to represent.

You can also use a single set of spin box buttons to edit a sequence of related text boxes, for example, time as expressed in hours, minutes, and seconds. The buttons affect only the text box that currently has the input focus.

TListView,Visual display of a list in columns

List views display data in a variety of views. The figure below shows a Delphi application to edit resources. It uses a menu to toggle the list view displayed in the right hand panel.



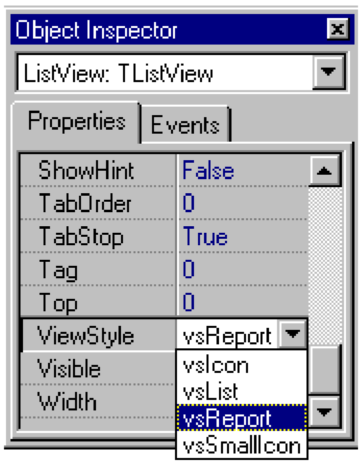
Name	Offset	Size
ARROW	00E13EC	00040
ATHENA	00E142C	02BD8
BBABORT	00E4004	001D0
BBALL	00E41D4	001E4
BBCANCEL	00E43B8	001D0
BBCLOSE	00E4588	001D0
BBHELP	00E4758	001D0

The Windows Explorer can quickly show you the different display possibilties of a list view. Select **View |**

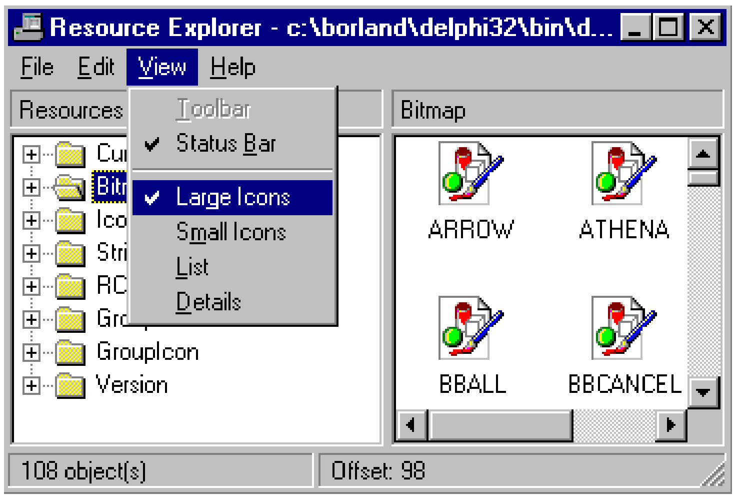
- Large Icons
- Small Icons
- List
- Details

The viewstyle property toggles listview display with one line of code:

```
procedure TMainForm.SelectListViewType(Sender: TObject);
begin
  ListView.ViewStyle := TViewStyle(TComponent(Sender).Tag);
  {0 Tag = VSReport, 1 = VSList, 2 = VSReport, 3 = VSSmallIcon}
end;
```



ShowHint	False
TabOrder	0
TabStop	True
Tag	0
Top	0
ViewStyle	vsReport
Visible	
Width	



TTabControl, a tab set

TTabControl functions similarly to a TTabSet. To create a multiple page dialog box, use a TPageControl. To use simply a tab set, use the TTabControl.

TPageControl, a page set used as a multiple page dialog box

A tab control is analogous to a divider in a file cabinet or notebook. You can use this control to define multiple logical pages or sections of information within the same window.

Conclusion

Applications are easier to use if they have a uniform design, people naturally know what to do. When a Windows application maintains user interface conformity, users instantly know how to use the application whether it is a spreadsheet, database, or decision support tool. With a common UI, applications seamlessly integrate into the environment of ole automation. Delphi 2.0 provides a powerful rapid application development environment with full Windows 95 and NT support.