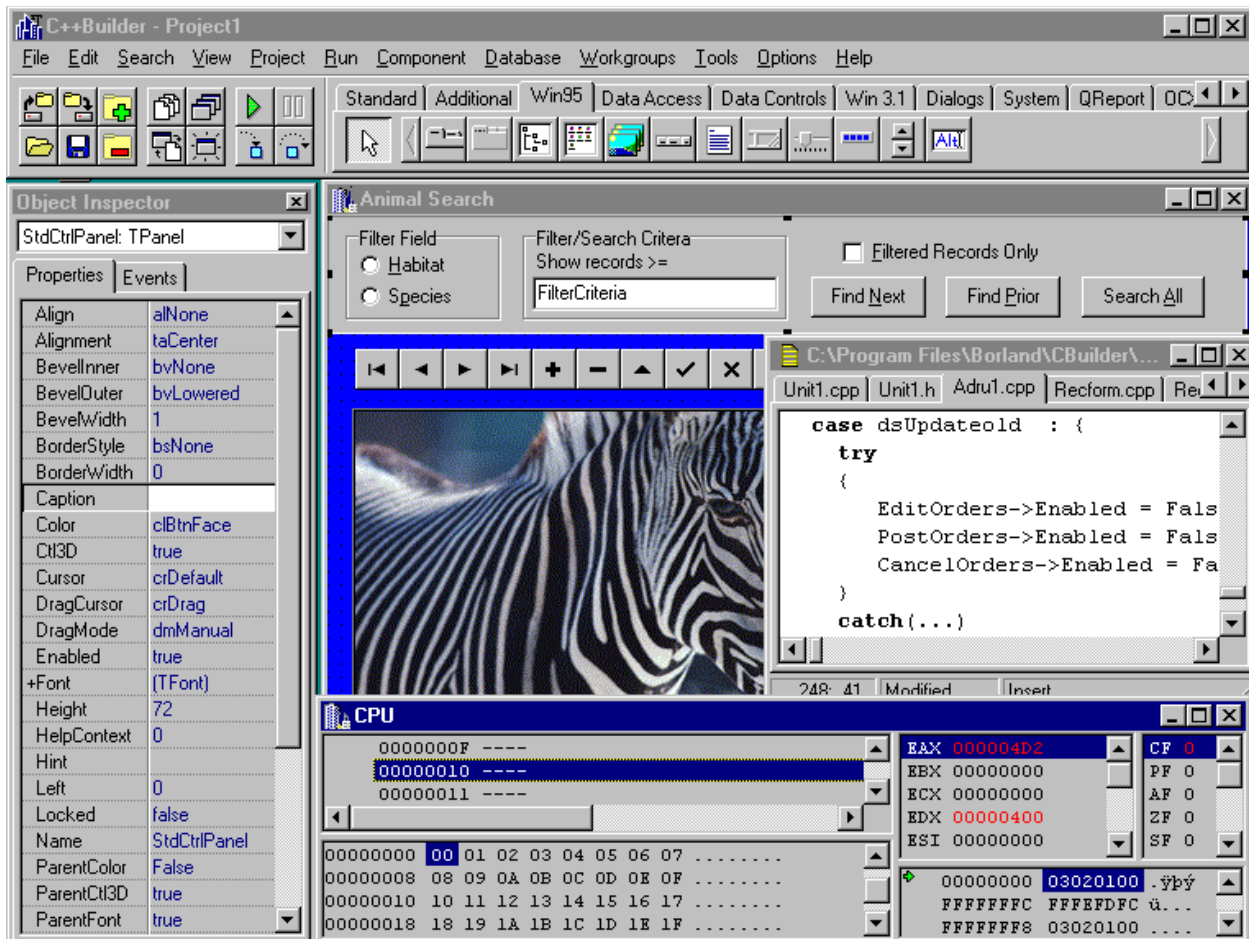


Borland C++Builder *Jump Start!*

A Quick Tour

Borland C++Builder for Windows 95 and NT gives you the speed of visual drag & drop development, the productivity of over 100 reusable components with source, and the flexibility of scalable database tools, combined with the power and control of industry-standard C++.



C++Builder combines the speed of visual development, the productivity of components and the power of C++.

Designed for rapid application development, C++Builder helps you build applications to gain and maintain a competitive advantage. You can quickly move your applications from prototype to production. To illustrate the power and flexibility of C++Builder, let's take a quick tour and build an application. To develop with C++Builder you need to understand its core development tools:

- **Component Palette**, with over 100 reusable components to build applications. C++Builder Professional ships with full source code to all the VCL components.
- **Object Repository**, a central location for all reusable objects like forms and data modules. Sharing of application objects reduces development time.

- The *Object Inspector* lets you set the Properties of objects visually without writing code. It shows the Events for an object and links to the code executed when events occur.
- *Form Design* area, to create the application user interface
- *Code Window*, to create event handlers and write code.
- *Database Explorer/Data Dictionary*, to browse information and protect data integrity.
- C++Builder's powerful *CPU View* consists of five separate panes that give you a view into a specific low-level aspect of your running application. Views include the disassembly pane, call stacks, flags, registers and memory dump.

A simple application

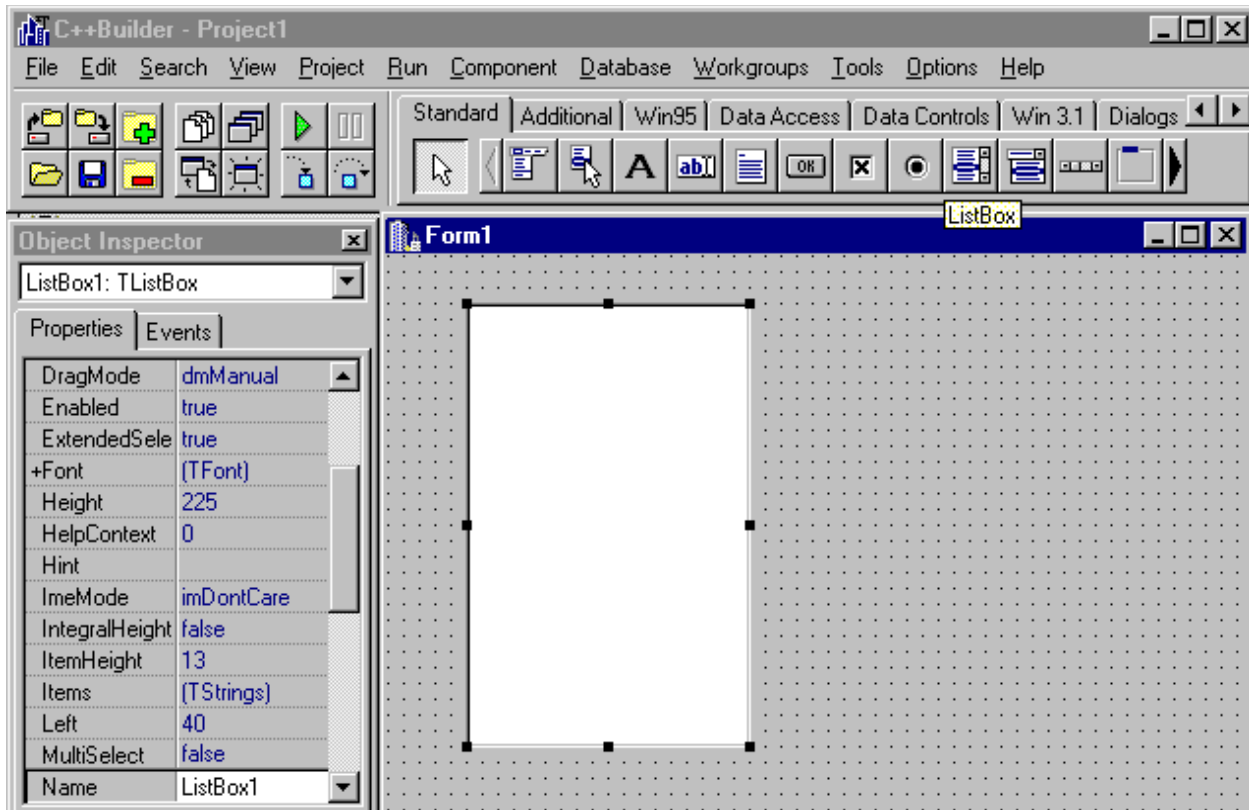
→ ***Throughout this Quick Tour, you may follow along in C++Builder by executing the steps denoted by an arrow.***

You build applications visually in C++Builder by selecting the component you want to use from the Component Palette. Every component, like a button, has a series of properties that change its look and behavior. Components also have a series of events that change the way components respond to actions. Our first application will use three objects: an edit field, a list box, and a button. Using the button, we will add items into our list box.

→ **To begin, open C++Builder and choose File | New Application**

→ **Click on the various tab pages at the top of the Component Palette.**

As you move from the Standard to the Win95 page, notice that the available components for your application change. (A list of all the C++Builder components follows in the next section.) Let's now drop a component on the form and begin enhancing our application.



C++Builder's true visual development reuses components from the Component Palette to speed development.

➔ **Click on the Standard Component page**

➔ **Click the ListBox  component**

➔ **Click anywhere in the form design area to drop the listbox**

➔ **Click on the Editbox component  and drop onto the form**

➔ **Click on the Button component  and drop onto the form.**

Real visual development, real C++

Completely integrated with its visual development features, C++Builder delivers extensive command-line tools, CPU view, and award-winning Borland C++ compiler technology with incremental/smart linking for the flexibility and incredibly fast build times expected by professional programmers.

At this point, you can move to the code editor to write and compile any ANSI C++ code including the newest ANSI features: Templates, Exceptions, Runtime Type Information (RTTI), and Namespaces. Simplify development with the Standard C++ Library. The Standard C++ Library gives you features like: string, complex, and numeric limits classes, and includes the Standard Template Library (STL) consisting of container and iterator classes. Plus, get new ANSI/ISO C++ features like bool, explicit, mutable, and typename.

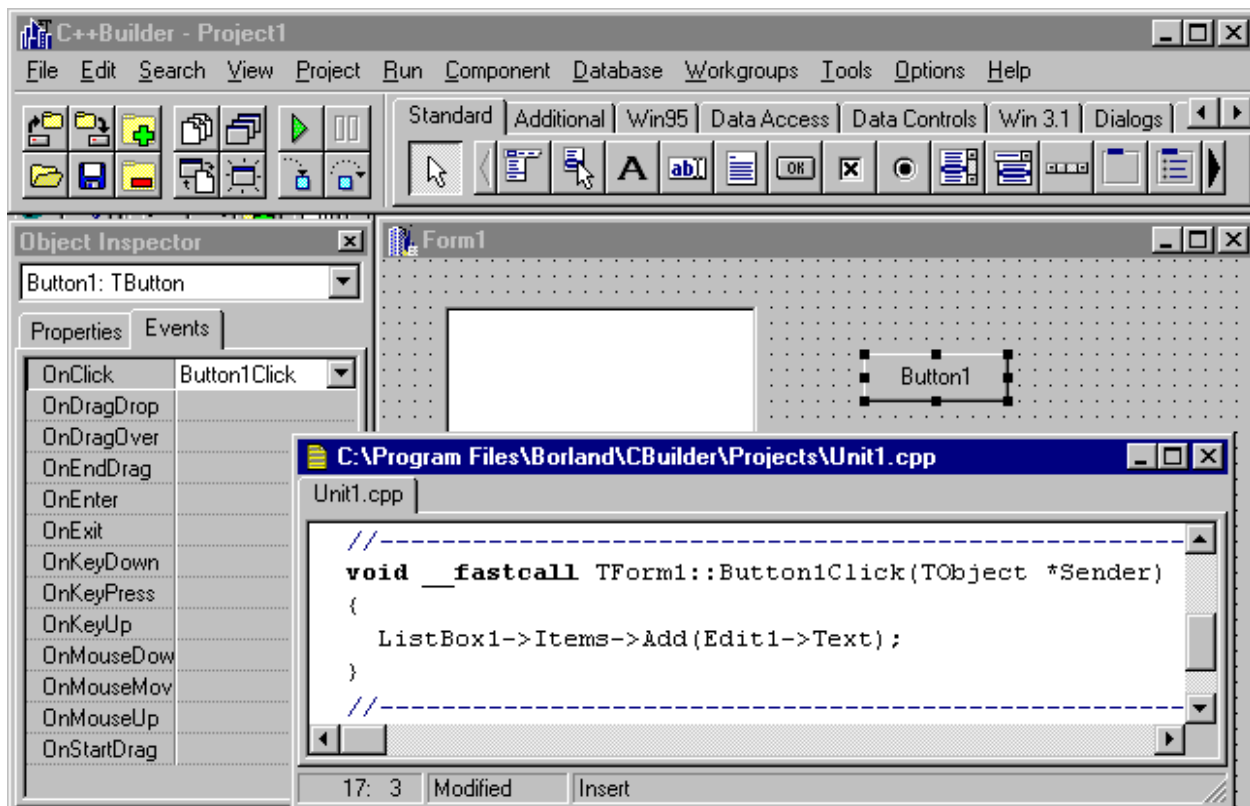
Properties, Methods, and Events

True rapid application development (RAD) means support for object properties, methods and events (PME). **Properties** allow you to easily set component characteristics such as the caption, helpcontext, or datasource. **Methods**, or member functions, support actions on objects like playing or rewinding a multimedia control. **Events** allow your code to respond to Windows messages such as button press notifications, edit changes, and control activation. In addition, events can respond to custom state changes, such as data updates for data-aware controls. Combined, RAD PME provides a robust, intuitive development environment for Windows programming.


- ➔ Click on the Object Inspector's **EVENTS** tab to see all the events for the currently selected object.
- ➔ Double click on the button you placed on your form.
- ➔ In the code editor type in the following code

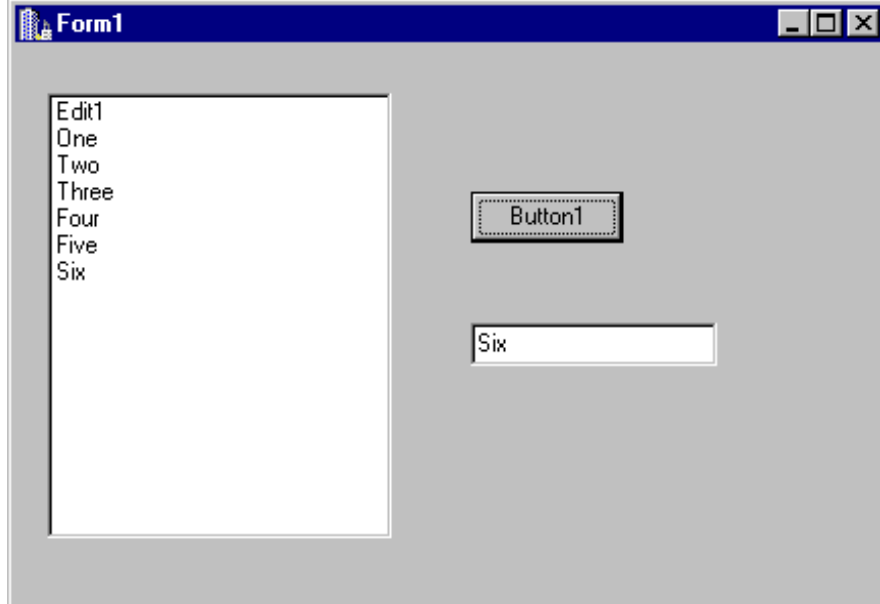
```
ListBox1->Items->Add(Edit1->Text);
```

You are telling the object (ListBox1) to increase its Items property using the Add method. The new items come from the values typed in at runtime to the Text property of the EditBox component.



Implementation code is entered in Unit1.cpp—C++Builder compiles any ANSI C++ code.

- ➔ Press the green arrow  to compile and link the application
- ➔ Once in run mode, click Button1 to add text from the Editbox to the Listbox.



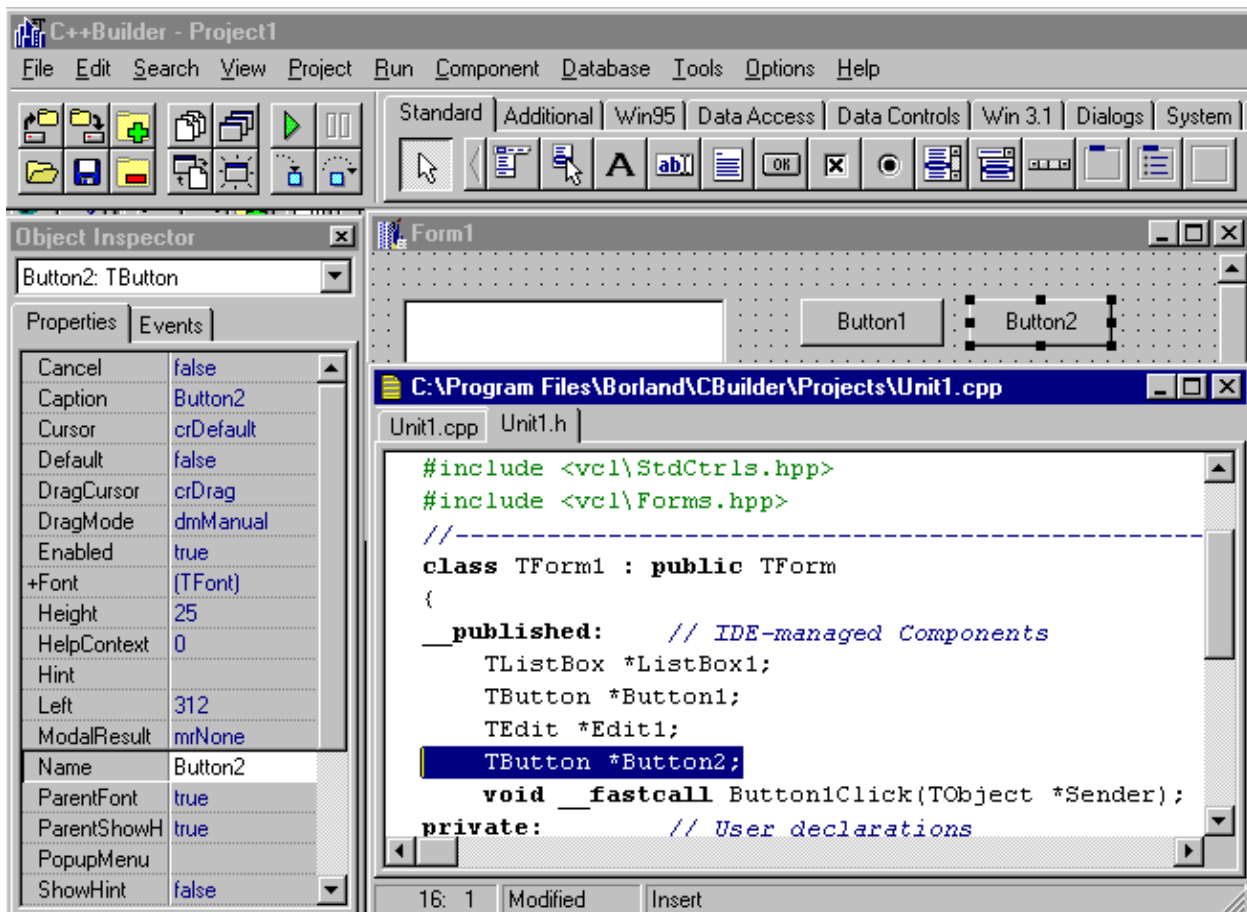
High performance C++ executable for unlimited distribution

Borland Two-Way Tools

C++Builder places no barriers between you and your code. With Borland Two-Way Tools, C++Builder lets you maintain control of your code by providing a seamless integration between the visual designers and the code editor. You have the flexibility to use the code editor or the visual designers interchangeably and know that the two are always synchronized.

To view two way tools in action:

- ➔ **Right click in Unit1.cpp and choose Swap CPP/Header Files**
- ➔ **Split the screen so that you can see both the form and the header file at the same time.**
- ➔ **Drop any object (like a button) onto the form.**



C++ Builder places no barriers between you and your code

Notice that the minute you drop an object onto the form it immediately appears in the header file. This seamless interaction between the visual environment and the code editor allows C++ developers to rapidly build visual applications while staying in complete control of the underlying source code.

Each C++Builder application begins with three files:

- **Unit1.cpp:** cpp files store your application's implementation code, it is in this file that you write a set of event handlers. The event handlers determine how your application responds to actions by the user or application.
- **Unit1.h:** The header file contains the constructors or declarations for each object. Fastcall is an intelligent way to pass the parameters. They are passed by register rather than in memory, because memory calls must be reloaded and called by routine.
- **Project1.cpp:** The project files manages all of the objects contained in your application. Each new form, unit, or data module is automatically added into the project file. You can view the project source by selecting View | Project Source from the main menu.

All of the files use precompiled headers, these binary, fast loading, prepared files speed up compile time. Again, all of the compiler directives and included files are completely customizable by the developer.

Borland Family of Tools

C++Builder adds the power and control of C++ to Borland's family of object oriented, rapid application development tools. C++Builder can be used to complement existing applications by using industry-standard C++ to build fast, productive front ends for your Visual C++ and Borland C++ applications. C++Builder protects your investment in legacy C++ code by compiling all ANSI C++ code.

Borland C++ and Borland C++Builder

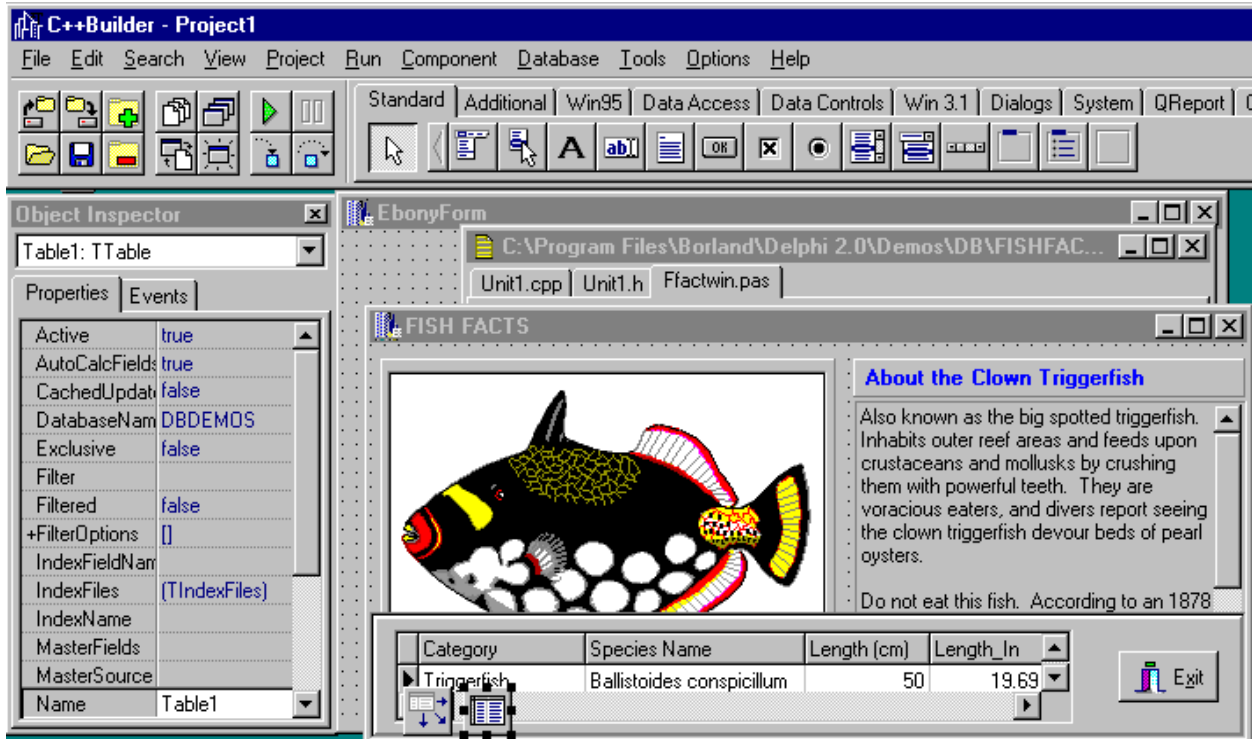
Leverage the built-in interoperability between Borland C++ and C++Builder to add rapid UI development to your development process. Complementary project file support, common compiler and incremental linker, and complete technical documentation allow you to deliver C++Builder front-ends to your C++ applications in record time. You can easily host C++Builder forms in your OWL and MFC applications or use Borland C++ to build DLLs, OBJs, or COM objects for your C++Builder applications. Plus with C++Builder you'll get full support for industry standards, including ANSI C++, the Win32 API, ActiveX, OLE Automation, ODBC, DCOM, MAPI, Unicode, WinSock, ISAPI, and NSAPI.

Delphi and C++Builder

As a Delphi developer, there are times when you need to use C++. C++Builder for Windows 95 and NT is the perfect complement to your Delphi applications. C++Builder lets you leverage your investment in Delphi knowledge and applications by sharing the same proven Visual Component Library, intuitive IDE, visual Two-Way Tools, Visual Form Inheritance, and scalable database connectivity. Your *Delphi code, components, data modules and forms can be reused* in C++Builder applications. Delphi continues to be the easiest to use, highest productivity rapid application development environment – while C++Builder provides the full power and control of C++ to those who prefer C++, but want the productivity of Delphi. Departmental organizations within corporations are empowered to use language of choice while seamlessly sharing objects and code. The result of this interoperability is unprecedented cross-departmental productivity.

To add an existing Delphi object to your C++Builder application:

- ➔ **Select File | Add to Project from the main menu.**
- ➔ **Change the name property of your C++Builder form to CPPForm. (This ensures you have unique names)**
- ➔ **Move to a directory containing a Delphi form
Program Files/Borland/Delphi 2.0/Demos/DB/FishFact**
- ➔ **Filter the drop down to show .PAS files. Click OK**
- ➔ **Press Shift F12 to view Form1**



C++Builder allows you to reuse your Delphi forms, Data Modules, components and code.

Now that the Delphi form is part of your application, you need to generate a C++ header file for the form.

- ➔ **Click any CPP form in your application and select File | Include Unit Hdr...**

You can now compile and run the application. Using Visual Form Inheritance, you can modify the Delphi form using C++.

The Component Palette

Borland C++Builder includes the Borland Visual Component Library (VCL), containing over 100 reusable components that developers “drag and drop” to create sophisticated Windows applications. The VCL includes full encapsulation of the Windows 95 user interface elements (Tree Views, Trackbars, sliders, progress bars, Toolbars, Rich Edit, List Views, Image Lists header and status bar controls, etc.), along with many additional components. These additional components complement the existing user interface, data-aware, and multimedia tools to help you build sophisticated applications and move from prototype to production quickly.

In order for professional developers to more rapidly create their own reusable custom components, C++Builder Professional and Client/Server Suite include source code for Borland Visual Component Library, which provides unique insights into the workings of C++Builder and its components. Source code availability lets you learn from examples and reuse existing components.

C++Builder lets you exploit the power of object-oriented programming to create robust, efficient applications. Build your own components with C++Builder's proven object-oriented component architecture. Because C++Builder is a completely object-oriented environment, integration of ActiveX controls is seamless. Use, customize, or subclass components to fit your development needs – all from within the C++Builder environment.

The components used in C++Builder are built upon an application framework, or a set of object types used to build the foundation of an application. This framework is called the Visual Component Library (VCL), and uses the object model implemented in C++Builder to give developers a true object-oriented development system.

VCL includes full encapsulation of the Windows 95 user interface elements along with many additional components. Out of the box, VCL includes most of the standard components used by Windows programmers such as user-interface objects for editing input, combo boxes, and list boxes, grid, tab and notebook objects.

A key feature of C++Builder is the ability to not only *use* components in a visual design environment, but to *create* new components as necessary. New components, through the inheritance feature of object-oriented programming, can be as simple as an existing component with some added functionality, or a completely custom component that is based entirely on new code.

The ability to create new components means developers don't have to switch to another development environment when new, custom components are required. Designing and implementing new components in C++Builder is a straightforward process that allows developers to change the defaults of a given control, give existing controls some new functionality, encapsulate existing third-party Windows control, or create components that do something completely new.

There are three steps to creating a new C++Builder component:

- inherit from an existing component type
- define new fields, properties, and methods
- register the new component

Standard Page Components

The components on the Standard page of the Component Palette make the standard Windows control elements available to your C++Builder applications.



TMainMenu	Creates menus for your form.
TPopUpMenu	Create popup menus for your form.
TLabel	Displays text the user cannot select or manipulate, such as title text.
TEdit	Displays an editing area where the user can enter or modify a single line of data.
TMemo	Displays an editing area where the user can enter or modify multiple lines of data.
TButton	Creates a pushbutton control that users choose to initiate action.
TCheckBox	Presents a control that a user can toggle between Yes/No or True/False.
TRadioButton	Presents a control that a user can toggle between Yes/No or True/False.
TListBox	Displays a scrolling list of choices.
TComboBox	Displays a list of choices in a combined edit and list box. Users can edit data in the edit box, or select data in the list box.
TScrollBar	Provides a way to change which portion of a list or form is visible, or to move through a range by increments.
TGroupBox	Provides a container to group related options on a form.
TRadioGroup	Creates a group box that contains RadioButtons on a form.
Tpanel	Creates a panel that can contain other components on a form. You can use the Tpanel to create ToolBars or Status lines.

Additional Page Components

The components on the Additional page of the Component Palette make specialized Windows control elements available to your C++Builder applications.



TBitButton	Creates a button component that can display a bitmap.
TSpeedButton	Creates a button that can display a glyph, group to create a speedbar.
TMaskEdit	An edit box to provide specific formats for data entry or display.
TStringGrid	Creates a grid that you can use to display string data in columns or rows.
TDrawGrid	Creates a grid that you can use to display data in columns or rows.

Timage	Displays a bitmap, icon, or metafile.
Tshape	Draws geometric shapes, including an ellipse or circle and rectangle or square.
Tbevel	Creates lines or boxes with a three dimensional or chiseled appearance.
TScrollBar	Creates a resizable container that automatically displays scrollbars if necessary.

Win95 Page Components

The components on the Win95 page of the Component Palette provide access to Win95 user interface common controls available to your C++Builder applications.



TTabControl	TTabControl component is a tab set which functions similarly to a TTabSet and has the appearance of notebook dividers.
TPageControl	TPageControl component is a page set which is used to make a multiple page dialog box. It displays multiple overlapping pages.
TTreeView	Displays a hierarchical list of items, such as the headings in a document, the entries in an index, or the files and directories on a disk.
TListView	Displays a list of items in a variety of ways. The ViewStyle property determines whether items are displayed in columns with column headers and sub-items, or vertically or horizontally, with small or large icons.
TImageList	TImageList component is a container for a group of graphic images.
THeaderControl	Contains multiple, movable headers and is similar to a THeader component.
TRichEdit	TRichEdit component is a Rich Text Format memo control.
TStatusBar	The TStatusBar component is a window that displays status information.
TTrackBar	A TTrackBar component is an optionally-ticked bar control that contains a slider which marks a current Position.
TProgressBar	Tracks the progress of a procedure within an application. As the procedure progresses, the rectangular TProgressBar gradually fills from left to right with the system highlight color.
TUpDown	The TUpDown component consists of a pair of Up and Down arrow buttons. Clicking on these buttons increments and decrements a numeric value held in the Position property.
THotKey	The THotKey component is used to set a shortcut property at run time. The user can enter a key combination, typically consisting of a modifier key (such as Ctrl, Alt, or Shift) and an accompanying key (such as a character key, an arrow key, a function key, etc.), into the hot-key control.

Data Access Page Components

The components on the Data Access page of the Component Palette make specialized database access elements available to your C++Builder applications. The following illustration shows the Data Access components.



Datasource	Acts as a conduit between ttable, tquery, tstoredproc components and data-aware components such as dbgrid.
TTable	Provides live access to database tables through the Borland Database Engine. TTable is the interface between the Borland Database Engine and TDataSource components.
TQuery	Enables applications to issue SQL statements to a database engine--either the BDE or an SQL server. TQuery provides the interface between an SQL server (or the BDE) and TDataSource components.
TStoredProc	Enables applications to execute server stored procedures.
Tdatabase	While not required for database access, provides additional control over factors that are important for client/server applications.
TSession	Provides global control over database connections for an application. C++Builder automatically creates a default TSession component at runtime for applications which use database controls.
TBatchMove	Enables you to perform operations on groups of records or entire tables.
TUpdateSQL	Provides a way to use cached updates support with read-only datasets.

Data Control Page Components

The components on the Data Controls page of the Component Palette make specialized database control elements available to your C++Builder applications. The following illustration shows the Data Controls components.



TDBGrid	Accesses the data in a database table or query and displays it in a grid. Your application can use the data grid to insert, delete, or edit data in the database.
TDBNavigator	(A database navigator) moves through the data in a table or query, and performs operations on the data, such as inserting a new record or posting a record.
TDBText	Displays text on a form in a data-aware control.
TDBEdit	Data-aware edit box with all the capabilities of an ordinary edit box.
TDBMemo	Displays text for the user and permits the user display and enter data into a field

	much like a TDBEdit component. The TDBMemo component permits multiple lines to be entered or displayed, including text BLOBs (binary large objects).
TDBImage	Displays a graphic image from a BLOB (binary large object) stored in a field of the current record of a dataset.
TdbListbox	Data-aware list box. It allows the user to change the value of the field of the current record in a dataset by selecting an item from a list.
TDBCCombobox	Data-aware combo box control allows the user to change the value of the field of the current record in a dataset.
Tdbcheckbox	Presents an option to the user; the user can check it to select the option, or uncheck it to deselect the option. A database check box is aware of the data in a particular field of a dataset.
TDBRadioGroup	Displays a group of data-aware radio buttons. Only one of the radio buttons can be selected at a time, so the radio buttons present a set of mutually exclusive choices.
TdbLookupList	Provides the user with a convenient list of lookup items for filling in fields that require data from another dataset.
TDBLookupCombobox	Provides the user with a convenient drop-down list of lookup items for filling in fields that require data from another dataset.
TDBCtrlGrid	Displays multiple records from a data source. You control the layout and appearance of each record in a TDBCtrlGrid.

System Page Components

The components on the System page of the Component Palette make specialized system control elements available to your C++Builder applications. The following illustrations show the System components.



Ttimer	Causes an OnTimer event to occur whenever a specified period of time passes.
TPaintBox	Provides a way for your application to draw on the form in a specified rectangular area, preventing drawing outside of the boundaries of the paint box.
TFileListBox	Lists all the files in the current directory. To display files in a different directory, change the value of the Directory property.
TDirectoryListBox	A list box that is aware of the directory structure of the current drive.
TDriveComboBox	A combo box that displays all the drives available when the application runs.
TFilterComboBox	A combo box that is used to present the user with a choice of file filters.
TMediaPlayer	Controls devices that provide a Media Control Interface (MCI) driver. The component is a set of buttons (Play, Stop, Eject, and so on) that controls a multimedia device such as a CD-ROM drive, a MIDI sequencer, or a VCR.

T OleContainer	Embeds or links OLE objects in your C++Builder application.
TDdeClientConv	Establishes a DDE conversation with a DDE server application.
TDDEClientItem	Defines the item of a DDE conversation.
TDDEServerConv	Establishes a DDE conversation with a DDE client application.
TDDEServerItem	Defines the item of a DDE conversation.

Dialogs Page Components

The components on the Dialogs page of the Component Palette make the Windows common dialog boxes available to your C++Builder applications. The common dialog boxes provide a consistent interface for file operations such as opening, saving, and printing files. The following illustration shows the Dialogs components.



TOpenDialog	Makes an Open dialog box available to your application.
TSaveDialog	Makes a Save dialog box available to your application.
TFontDialog	Makes a Font dialog box available to your application.
TColorDialog	Makes a Color dialog box available to your application.
TPrintDialog	Displays a Print dialog box that permits the user to select which printer to print to, which pages to print, how many copies to print, and if the print job should be collated.
TPrinterSetupDialog	Displays a Printer Setup dialog box in your application. Users can use the dialog box to setup their printer before printing a job.
TFindDialog	Provides a Find dialog box to your application.
TReplaceDialog	Provides a Replace dialog box your application can use. TReplaceDialog contains all the capabilities of the TFindDialog component, but it also allows the user to replace found text with a replacement string.

Rapid Application Development of Database Applications

Database Speed and Scalability

C++Builder features a *Database Application Architecture* where Rapid Application Design and Object-Oriented methodologies are applied to both the GUI and Database Business Logic. Database access occurs through the core, 32-bit Borland Database Engine. New Data Dictionary and Data Modules support business rule development. With centralized data integrity and business rules, you applications can quickly respond in a dynamic business environment.

All data access for C++Builder occurs through objects on the Data Access component page. Objects on the Data Access page encapsulate database source information, such as the database to connect to, the tables in that database to access, the query to use, and specific field references within the data. C++Builder uses the high performance Borland Database Engine with 32-bit native drivers to develop applications for DB2, Sybase, Microsoft SQL Server, Informix, Oracle, InterBase (4-user 95/NT license included), Paradox, dBASE and the Local Interbase (also included). C++Builder also has ODBC connectivity to access any ODBC compliant server such as Access, VSAM, or AS400. C++Builder's high speed native access to database servers means the highest performing Client/Server applications.

To create a database application:

- **Select File | New from the main menu**
- **Click on the tab labeled *Data Modules***
- **Click on Customer Data and then click OK**

C++Builder's Data Module Objects

C++Builder Client/Server Suite Suite Data Module Objects act as your applications information core by providing a designated central location for defining data access and business rules. The Data Module Object separates business logic from the GUI and acts as a codeless way to connect and manage this business logic from a single location.

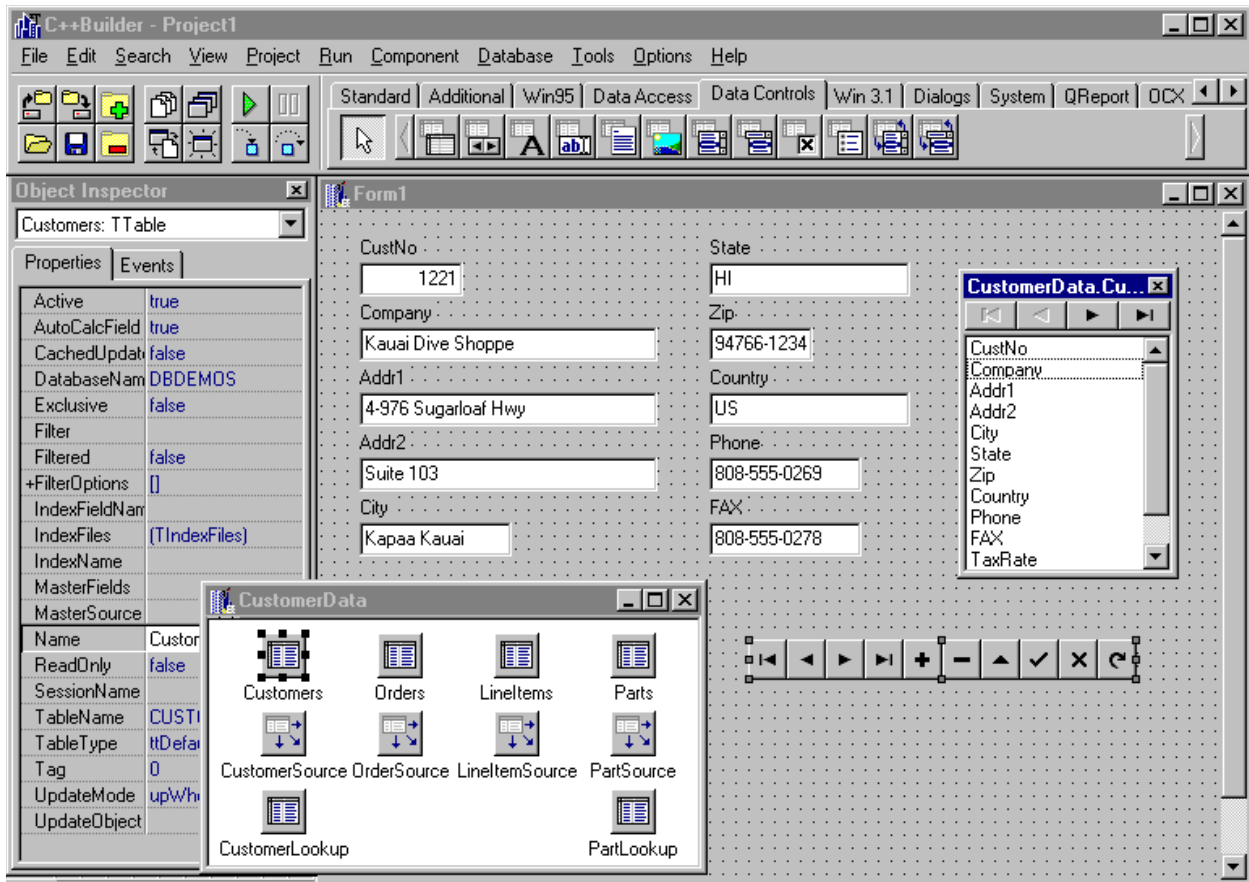
Once you have created a data module, it may be shared throughout an enterprise, any form may call into the central data module for its data display attributes and values. This separation of the GUI Design from the data and its associated logic means that changes in either of these areas does not impact the successful usage of the other. Changes in either the GUI or business logic can be implemented according to their own requirements. Developers can apply their skill sets appropriately. No longer does a single developer have to have expertise in Database Design, Business Methodologies, and GUI Design to create an effective Client/Server application.

Drag and Drop database Development

To enhance developer productivity, Borland C++Builder identifies and eliminates many repetitive tasks. Reusing Data Modules from the Object Repository lets you drag and drop your way to a database application in no time.

- **Double click on the Customers table in you data module, Customerdata.Customers appears.**
- **Multi select several fields by holding down the shift key and using the mouse.**
- **Drag and drop the selected fields onto your form.**
- **Drop a DbNavigator from the Data Control page and connect its DataSource Property to CustomerData->CustomerSource.**

You can now use live data in design mode to view all the records in the table. Click on the arrow in the Customer Data dialog that lists all of the fields in the table. Compile and run the application.



C++Builder has robust scalable database tools like Data Modules, data access components and data controls.

Conclusion

Borland C++Builder for Windows 95 and NT gives you the speed of visual drag & drop development, the productivity of over 100 reusable components with source, and the flexibility of scalable database tools, combined with the power and control of industry-standard C++. Leverage your existing investment in C++—C++Builder compiles all ANSI C++ code, and allows you to build fast, productive front ends for your Visual C++ and Borland C++ applications. Plus, you get full support for industry standards, including ANSI C++, the Win32 API, ActiveX, OLE Automation, ODBC, DCOM, MAPI, DirectX, Unicode, WinSock, ISAPI, and NSAPI.

For updates to this document, please view the Borland Web site at www.Borland.com

Copyright 1997 Borland International