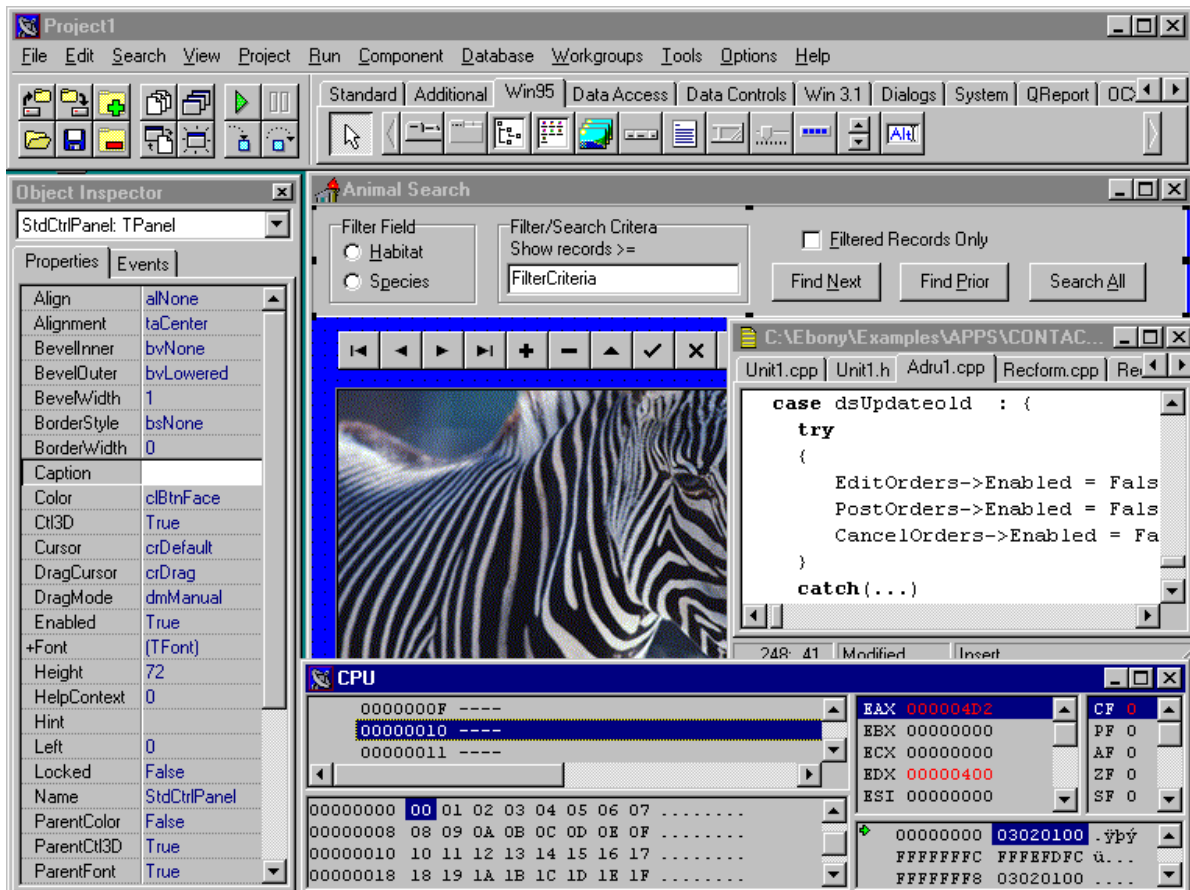


C++Builder *Jump Start!*

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

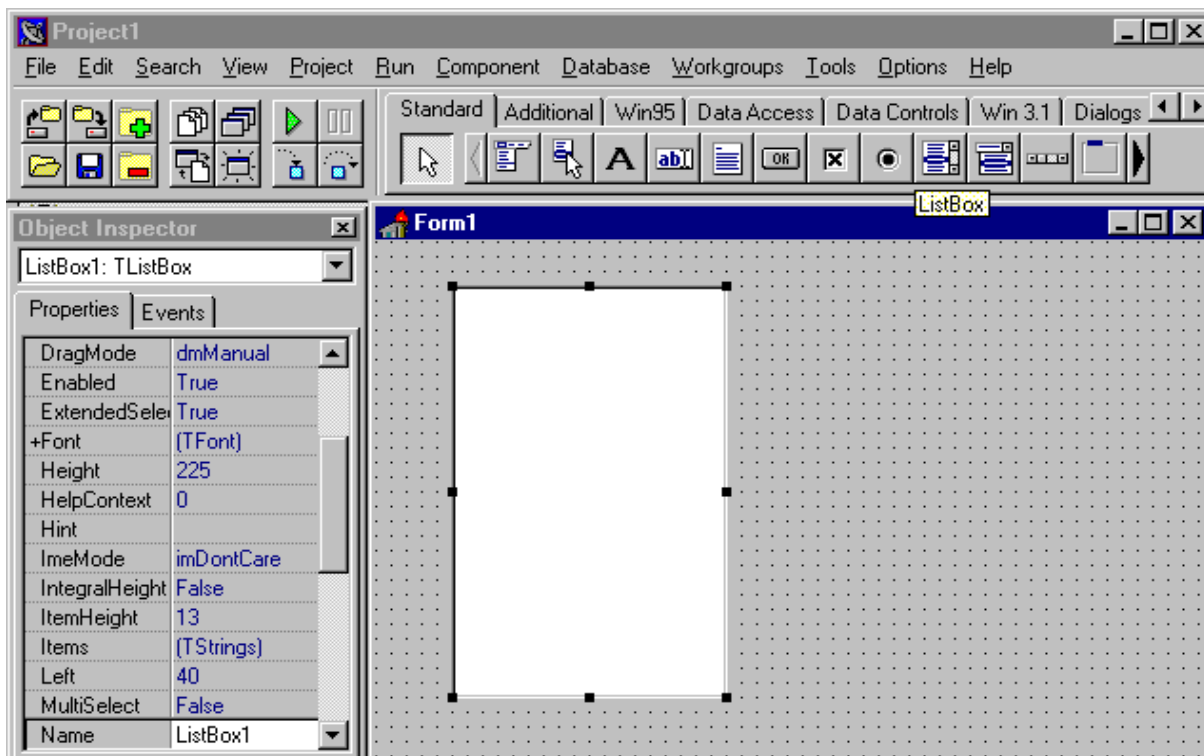
➔ *Throughout this Quick Tour, you may follow along in C++Builder by executing the steps denoted by an arrow.*

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.

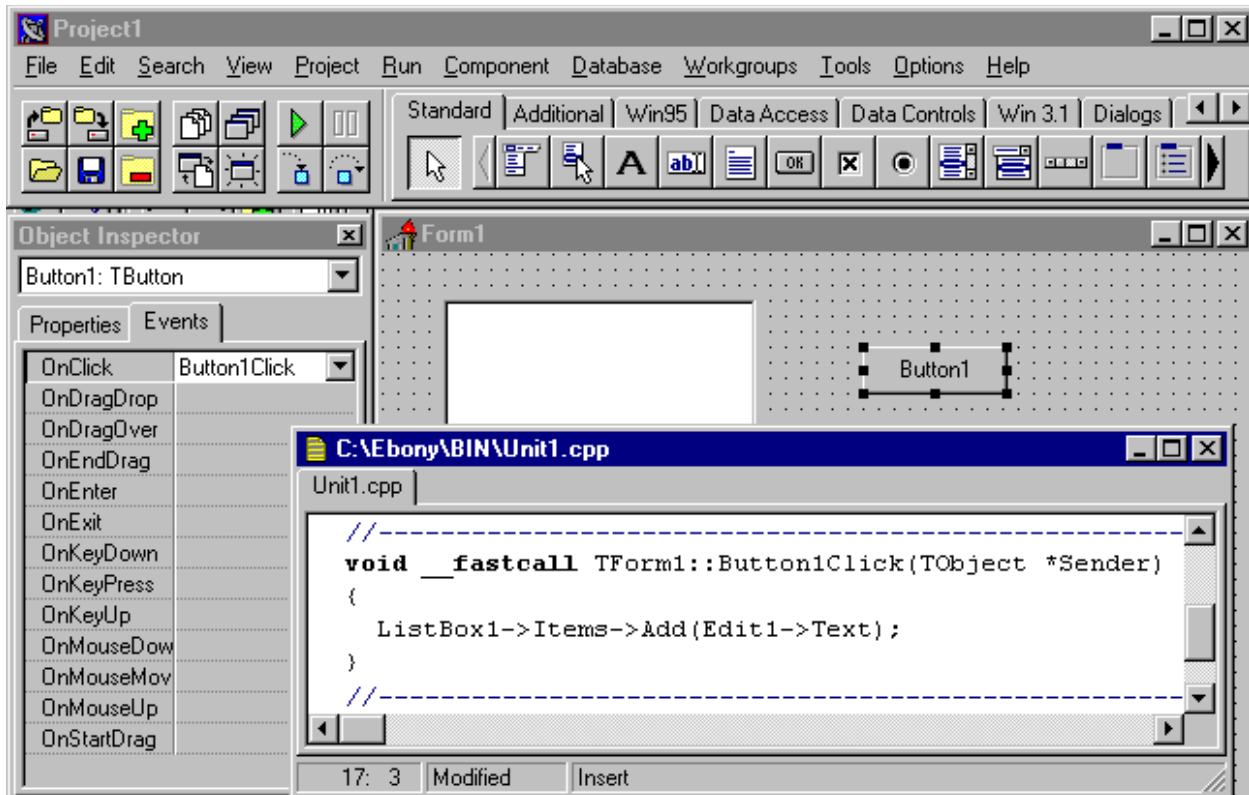
Properties, Methods, and Events

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.

The Component Palette

C++Builder includes an enhanced implementation of the Visual Component Library, containing over 100 reusable components that developers “drag and drop” to create sophisticated Windows applications. VCL32 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 includes source code for the enhanced VCL32, 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 OLE controls (OCX) 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.

VCL32 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. Out of the box, VCL32 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.
TDBCmbobox	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.
TOleContainer	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

C++Builder's object oriented architecture offers maximum reusability and maintainability - the result is shorter development cycles and easier to maintain code. C++Builder includes an enhanced implementation of the Visual Component Library, containing over **100 reusable components** that developers “drag and drop” to create sophisticated Windows applications.

To create a database application, select an object from the Component Palette and drop it onto the form.

- ➔ **From the standard page of the component palette, drop a caption on the form designer.**
- ➔ **Use the Object Inspector to modify the caption properties: enlarge the font.**

One of C++Builder's most important capabilities is the full support for object-oriented programming that it provides. Because C++ has full support for encapsulation, polymorphism and inheritance in the language, C++Builder has the capability of allowing developers to create their own custom objects, whether they subclass (derive) from existing visual components or represent entirely new abstract business objects. Many third party vendors and C++Builder developers have created a large and growing marketplace for these components.

Windows 95 User Interface

In addition to 32-bit support for long filenames, multi-threading, and the complete Windows 95 and Windows NT APIs, C++Builder includes a complete suite of Windows 95 controls: TreeViews, Header controls, Status Bars, progress meters and more.

- ➔ **Click on the Win95 tab page of the component palette, and drop a tabbed notebook.**
- ➔ **Right Click the tabbed notebook to add two new pages.**

Because building on an existing foundation reduces development time, C++Builder's *Object Repository* stores forms, data modules, and complete applications for reuse.

- ➔ **Choose FILE | NEW from the menu, select a DATA MODULE object from the available options.**

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 (2-user NT license included), Paradox, dBASE and the Local Interbase Server (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.

- ➔ **Click on the Data Access component page. Drop 4 TTables and 4 TDataSource components into your data module.**

Whether local or remote, an *alias*, or shortcut to your data, maintains a flexible pointer to where the data resides.

- ➔ **Select each TTable and use the object inspector to set the DatabaseName property to DBDEMOS.**
- ➔ **Use the dropdown on the TableName property of a TTable to set the table to CustOly (the second to Events, the third to Venues and the fourth to Reservat). Change each TTables Name property to be more descriptive.**

C++Builder's Data Module Objects

C++Builder Client/Server 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.

- ➔ **Click in the Data Module and name it *OlympicModule*.**

➔ **Right click in the Data Module and select Add to Repository.**

➔ **When prompted, Save Unit 2 as OLYMODU.CPP**

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.

Business logic can be applied to Tables, Stored Procedures, and Queries by creating methods on Before and After events such as posts, deletes, inserts and edits. This allows you to create new business objects more easily.

Object Repository

C++Builder Client/Server Suite Object Repository stores and manages application objects: Forms, Data Modules, Experts, and DLLs. In essence, it centrally locates corporate assets so that they may be leveraged by the team to eliminate redundant development efforts. As objects proliferate, the repository increases in importance. Any new application can inherit, use, or simply copy an existing structure - you pick the architecture that best fits your development needs. This gives an organization the flexibility to decide:

- Who shares the code?
- Where does the code live? (server, network, client)
- How can developers use the code? (copy, inherit, reference)

➔ **Select the Data Module notebook page from the Object Repository, right click and View Details.**

➔ **Press the Cancel button.**

Drag and Drop database Development

To enhance developer productivity, C++Builder identifies and eliminates many repetitive tasks.

➔ **View your data module, OlympicModule (Shift F12 to locate)**

➔ **Double-click the Events Table and then right click to Add Fields. Click OK to add all selected fields.**

Adding a Lookup

Database applications must seamlessly connect to a variety of corporate databases. C++Builder's automatic lookups display data from multiple tables in a single source. The resulting application displays the data the users need in a format that is easy for developers to establish and maintain.

➔ **Right-Click in the fields editor and choose New Field. Fill in the blanks as shown below.**

When you drag and drop fields from the Fields Editor, C++Builder automatically formats the display. Using the data dictionary, you can customize the display of field objects and the data displayed within them.

- ➔ **Select all the fields, except Event_Description and Event_Photo and drag and drop them onto the tabbed notebook. Now select Event_Description and Event_Photo and drop them in the second column.**
- ➔ **Drop a DbNavigator from the Data Control page and connect its DataSource Property to Events.**

Compiler

Compile and run your application. Now go back and make a minor change to some code. Recompile and let the incremental linker have you running your new executable in no time at all!

Visual Form Inheritance

C++Builder takes fundamental OOP capabilities and extends them to the visual environment in order to make inheritance easier to use and more accessible. You can now visually inherit from forms while in the design environment, without writing code, and immediately see the effects of changes. For example, in many corporate environments it's desirable to create a standard "template" form, say for data entry, which will be used as the basis for many other forms. By using Visual Form Inheritance, you can be assured that as changes are made to the standard form they can be immediately inherited by other forms.

Visual Form Inheritance allows you to inherit all of the code, objects and properties with as many levels of inheritance as you like without runtime performance penalties. Other systems that attempt to implement inheritance have severe performance penalties which render the feature unusable in the real world.

Developing a corporate standard in applications is important. Ensuring that these standards are adhered to is more difficult. Visual Form Inheritance and Form Linking extends object oriented programming to a visual paradigm ensuring that corporate and programmatic standards are maintained from project to project. In conjunction with the Object Repository, these standards are centrally managed resulting in faster project turn-around time.

Visual Form Inheritance allows anyone to take advantage of object-oriented reusability and maintainability by providing a codeless way to use inheritance, encapsulation, and polymorphism.

- ➔ **Move Form2 down**
- ➔ **Drop a menu**
- ➔ **Form 2->ShowModal();**

Database Explorer

The Database Explorer provides a graphical way of managing all of your database demands, it supports the creation and modification of tables, indices, and aliases. Its integrated database schema and content management utility is tailored to the needs of professional database developers.

The DataGrid has also been enhanced to be independent of the Dataset. This provides the flexibility to use different grids to look at the same table, query, or stored procedure in different ways. The developer

can apply column attributes such as position, fonts, colors, headers, width, etc. to highlight important information and to convey different messages about the data.

Enhanced DataGrid with Codeless Lookups

Applications like Executive Information Systems and Decision Support Systems require consistent data for accurate reporting of daily operations. The new DataGrid in C++Builder Client/Server Suite allows for codeless lookups between tables which ensures data validation and consistency. Drop down lists can now be used to supply information from one table to store in another. The developer can now create a data entry model that provides for consistent data entry.

➔ **Double-Click to customize grid**

Filters

After a general query of a database it is common to want to successively pare down the list of results or to move through the list based on further criteria. Filters offer a flexible mechanism for subsetting the result set either on the client or on the server. In this way, the developer can choose what will offer the highest performance with the most flexibility.

Powerful filter expressions are easily written in C++ and do not have the limitations that are inherent in 4GL languages; limitations such as the scope and breadth of function creation.

➔ **OlympicModule->Reservat->Filtered = CheckBox1->Checked;**

➔ **OlympicModule->Reservat->Filter = ComboBox1->Text;**

Filters facilitate the way people work by allowing users to scan through data more effectively. Being able to drill into a result set is one way to turn data into information by mapping to the way people work.

DataListBox and DataComboBox

The DataListBox and the DataComboBox seamlessly link information between multiple Datasources whether they are queries, stored procedures, or tables. Developers can now codelessly order pick lists so that users can find data more easily and enhance their productivity. Client/Server applications rely on normalized, validated and coherent data so that important business decisions can be made accurately.

SQL/Database Explorer

The SQL Explorer provides the information center for your database management demands; it supports the creation and modification of tables, aliases, stored procedures, triggers and business rules through interactive SQL. This graphical tool is an integrated database schema and content management utility tailored to the needs of professional database developers.

The SQL Explorer, unique to C++Builder, makes database administration easier and more intuitive than having to use a separate non-integrated tool. A simple to use graphical interface is a perfect way to represent the complex relationships that exist in a database server. The SQL Explorer presents schema information from Oracle, Sybase, InterBase, Informix, DB2 and others. The developer can drag and drop fields, tables, and stored procedures onto the C++Builder application form to build Client/Server database applications quickly. The developer can also issue SQL statements directed to multiple servers and multiple databases.

The SQL Explorer also manages the Scalable Data Dictionary. The simple to use interface enables the developer to easily define new domains of extended field attributes and then associate those to a field. The next time the field is used in an application, all the attributes are automatically applied.

C++Builder Client/Server Suite provides the Multi-Object Grid and enhanced DataGrid with codeless lookup functionality, and enhanced database list and combo boxes. These controls offer the flexibility to convey information to the user in a way that corresponds to business needs and application requirements.

The multi-object grid allows the developer to place other database controls into a single row of this grid. CheckBoxes, Edits, and so on can be stacked or visually placed where you like, and with full replication. The flexibility and simplicity of conveying information in this way aids your application's presentation of information and ultimately enhances the productivity of your users.

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 recompiles 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.