Programming with Delphi 2005, Part 1

Huw Collingbourne begins a new series of Delphi workshops by explaining how to make the most of Delphi 2005

ON THIS MONTH'S DVD
Borland's Delphi 2005 is one of the most powerful programming systems available for Windows. The exclusive Personal Edition included with the DVD Edition this month provides all the tools you need to develop your own programs for Windows, including Microsoft's new .NET Framework.

Delphi's integrated development environment lets you design, edit, compile and debug your programs quickly and simply. And it supports two programming languages: Delphi's traditional Object Pascal and Microsoft's new C# (pronounced 'C sharp'). This month, we introduce you to the main features of Delphi 2005 and get you up and running with some sample programs.

To create an Explorer-like file manager, just drop two controls onto a form and set three properties in the Object Inspector.

THE DELPHI DESIGNER

Let's begin by taking a quick tour of Delphi's visual design tools. Start Delphi and click New on the File menu. A submenu pops out listing the available project types. Scroll down this menu and select 'VCL Forms Application – Delphi For Win32'. At this point, a blank form appears. Forms are a visual method of designing your application's GUI, taking a lot of the donkey work out of point-and-click program design. We can create an application simply by dropping components onto this form from the Tool Palette at the bottom right-hand corner of the Delphi environment.

The components in the Tool Palette are arranged into groups. You can switch from one group to another by clicking the Categories button and making a selection from the drop-down list. For now, we'll stick with the Standard group. Click TEdit in the Tool Palette, then click somewhere on the blank form. A new TEdit control, named Edit1, will appear at the position where you clicked. You can move the TEdit control by dragging it with the mouse and resize it by selecting and dragging an edge or a corner. Go back to the Tool Palette and click TButton, then drop one of these onto the form. You'll now have a button named Button1. You can change the name of the button using the Object Inspector at the bottom left of the Delphi workspace. Scroll to the bottom of the Inspector and find the Caption property in the Visual group. Select 'Button1' in the right-hand column and edit this to 'Click Me!'. Notice that this causes the caption of the button on the form to change too.

COMMAND AND CONTROL

Having learnt how to put controls onto a form and alter their properties, it's remarkably easy to create applications of considerable complexity with minimal coding. To prove this, we'll create a Windows Explorer-style file browser. Start a new project with File | New | VCL Forms Application. When prompted, don't bother saving your changes to the current project. In the Tool Palette, scroll down to the Samples category at the bottom of the list. Select TShellTreeView and click the form to drop the ShellTreeView1 component onto it. Making sure this component is still selected, find its Align property in the Layout group of the Object Inspector. Currently, this property is set to 'alNone'. Click the down-facing arrow to the right of the property to display a list of available values. Select 'alLeft'. Note that the component now aligns itself to the left of the form.

Drop a TShellListView onto the grey area of the form. Set its Align property to 'alClient'. This causes it to occupy the blank (client) area so that it fills the part of the form not already taken up by the ShellTreeView component.

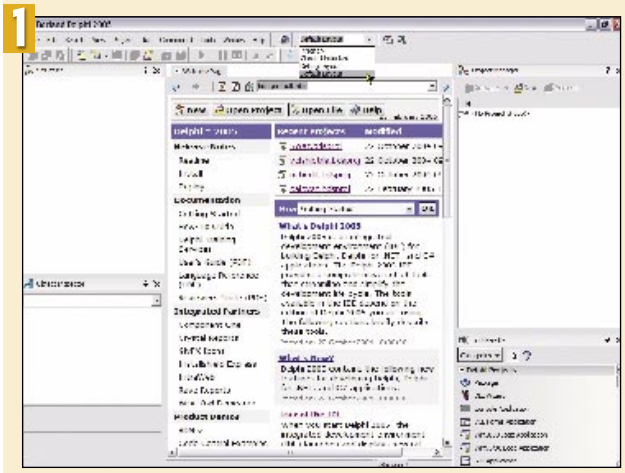
RUNNING YOUR APPLICATION

Now Press F9 or click the green arrowhead in the Delphi icon bar to run your application. As you can see, this shows the directory tree of your disk and the files in

Continued on p165

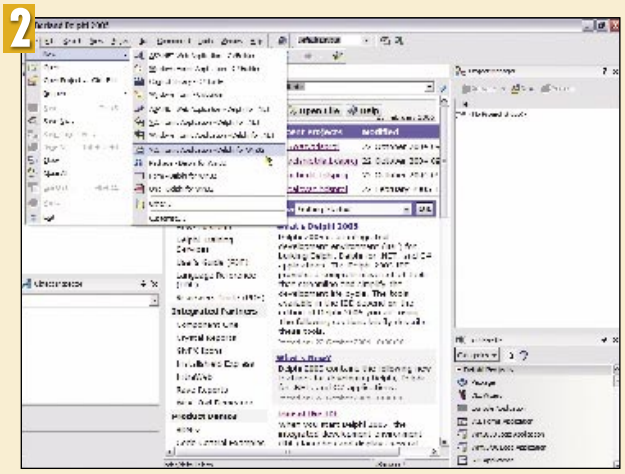
Walkthrough 1: setting up the IDE

1



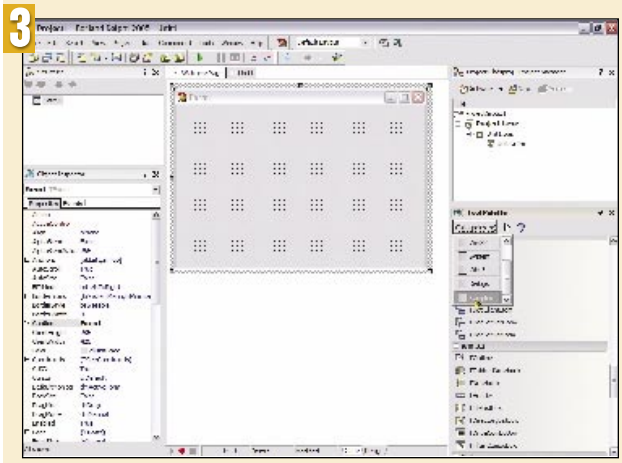
Delphi 2005 has an integrated development environment in which you can design, code, debug and compile programs. If you prefer free-floating windows, select Classic Undocked from the Layout combo at the top of the screen. The welcome page shown here has links to tutorials and help documents.

2



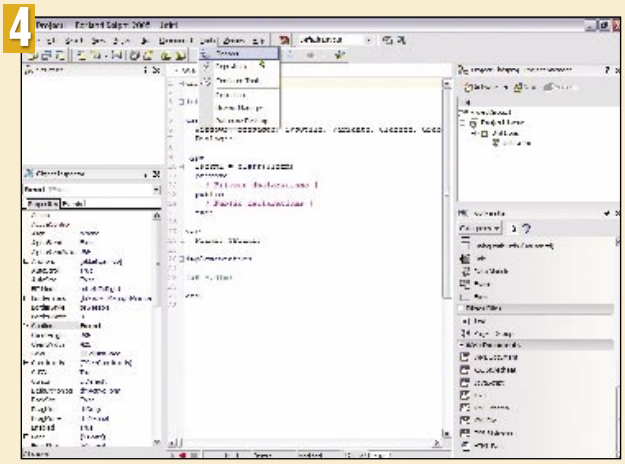
When beginning work on a new application, you first need to select the type of project from the File menu. Delphi 2005 supports both the Delphi Object Pascal language and the C# language. All this month's projects are VCL Forms applications using Delphi For Win32.

3



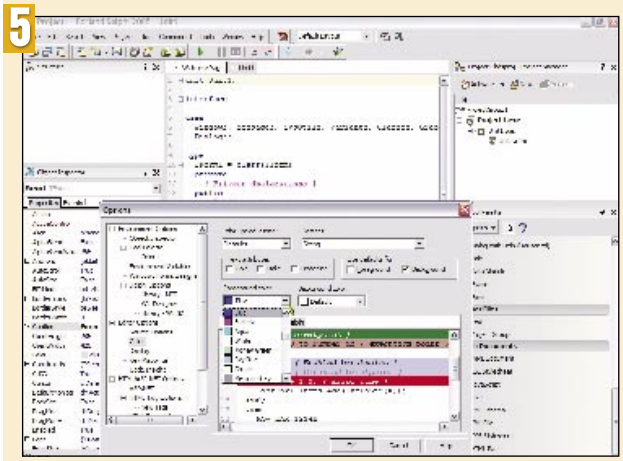
When you begin a new project, a blank form appears in the workspace. You can add controls to the form by dragging them from the Tool Palette seen here at the bottom right. Controls are grouped into categories that can be navigated by clicking the Categories button.

4



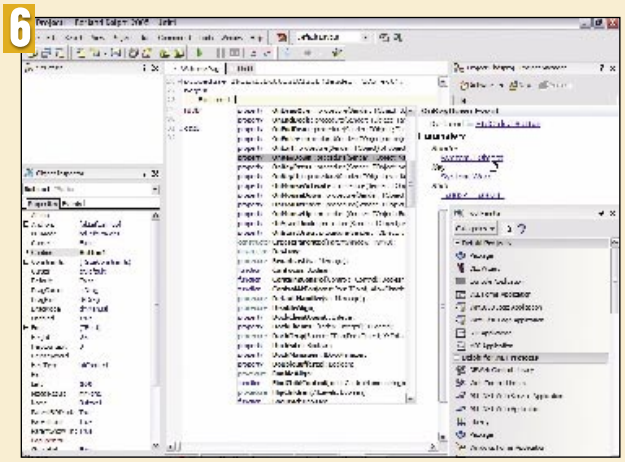
To write Object Pascal code in the editor, click the Code tab at the bottom of the workspace. Notice that different code elements such as keywords and comments are displayed in different colours. You can customise the colours and other features of the editor by selecting Options from the Tools menu.

5



Here, we're setting the colour coding for strings in Delphi's Options dialog box. First, we've selected String from the Element list then picked Blue as the Foreground colour. You can select other options such as Code Insight from the tree view on the left of the dialog.

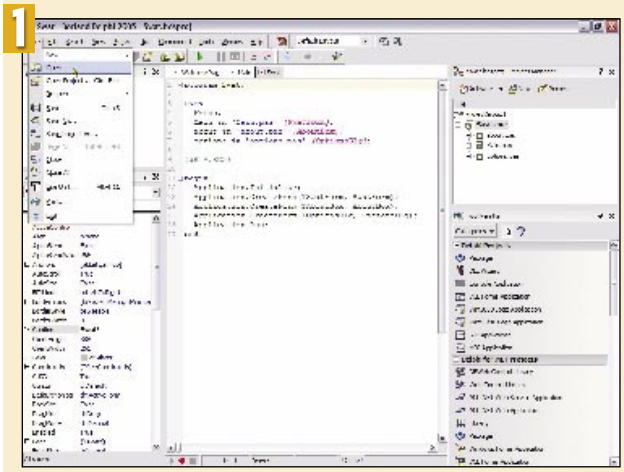
6



Now we're using Code Insight in the Delphi editor. We've entered a full stop after the name of one of the controls on the form, and a Code Insight selection list immediately shows us the available properties and 'methods' (procedures) that can be used with it.

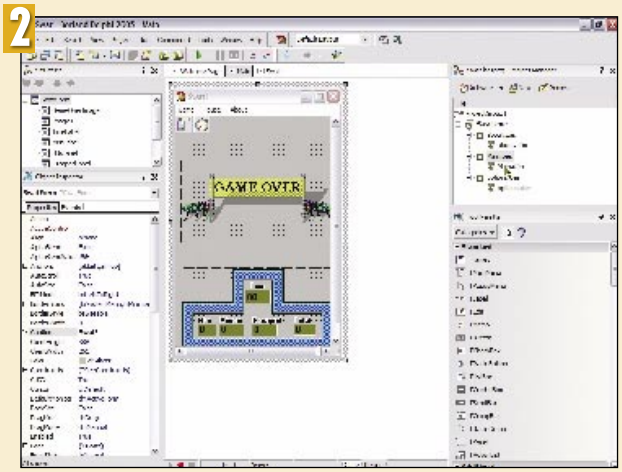
Walkthrough 2: exploring a demo project

1



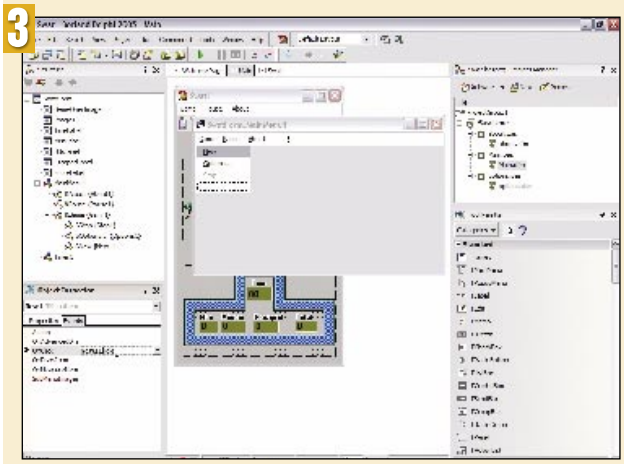
Delphi comes with many sample projects. To load one of them, navigate to the \Demos subdirectory beneath the \BDS\3.0 installation directory and open either the Borland Development Studio Project file (with the extension .bdsproj) or, for Delphi Win32 applications, the Delphi Project File (with the extension .dpr).

2



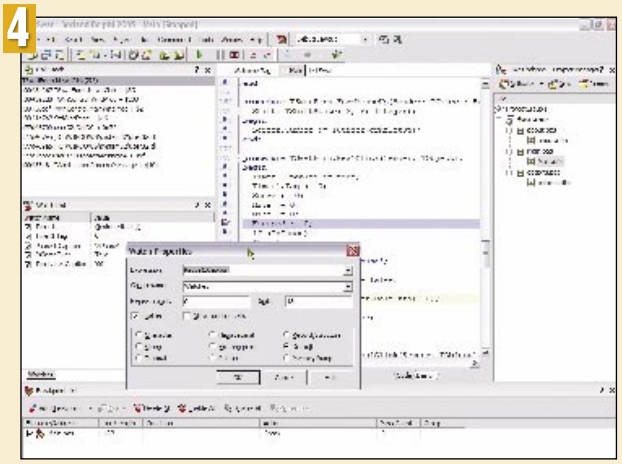
Here, we've loaded the swat.dpr project, which implements a simple game. Note that the project comprises three code ('.pas') units and three forms ('.dfm'). These are shown in the Project Manager at the top right. The form contains several controls, which are shown in the Structure panel at the top left.

3



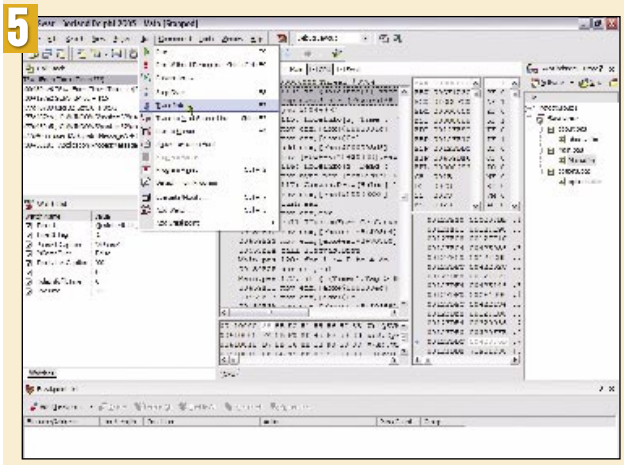
The MainMenu1 control, seen just beneath the Game menu on the form, isn't displayed at runtime. To create a menu system, double-click MainMenu1 to pop up the menu editor. Here the New menu item is selected and its OnClick event is shown in the Object Inspector panel.

4



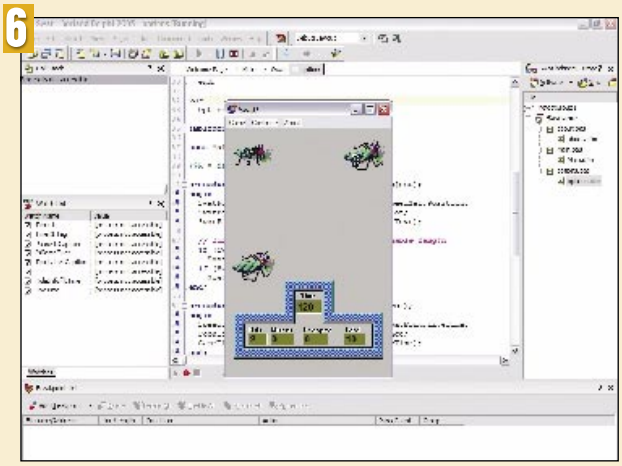
We've double-clicked the New item's OnClick event, which causes Delphi to display the source code of the event-handling method, New1Click(). We've double-clicked the left-hand margin of the editor in order to put a breakpoint on a line of code and add some variable names to the Watch List window.

5



The program pauses on our breakpoint and we can now examine the current values of the watched variables. If you're familiar with Assembly language, you can trace the execution of an application, one line at a time, in the CPU window. Otherwise, you can trace execution in Delphi code.

6



After a hard session of debugging, it's time to relax with a game. In Borland's Swat!, you appropriately have to kill as many bugs as you can by hitting them on the head with a little hammer. Pay attention to the way the Timer object is used to animate the graphics.

Continued from p162

the current directory. The only problem is that when you select a new directory in one control, this doesn't change the directory in the other one. That's easy to fix. Close the running application and in the design workspace select ShellListView1 at the right of the form. Scroll the Object Inspector to the Linkage group. Select the ShellTreeView property and, from the drop-down list, pick ShellTreeView1. Run the application again with F9. The two controls should now work in unison.

CRACKING THE CODE

Of course, there's a limit to what you can do by setting properties in the designer. In order to develop real-world applications, you'll need to write some program code. Rather than a trivial 'Hello World!' program, we'll try for something a bit more useful as a first real application.

Start a new VCL Forms Application for Win32. Drop onto the form three TEdit controls, one beneath the other, and set their Name properties to SubTotal, Vat and GrandTotal. This is going to be a VAT calculator, which will calculate tax based on a value entered into the SubTotal edit box. Drop a button onto the form, name it CalcBtn and set its Caption to 'Calculate'.

Double-click the button to create an event-handling procedure for the default Click event – you can code methods to handle other events such as key presses using the Events tab in the Object Inspector. Delphi creates an empty event-handling method called CalcBtnClick(). This method will be called automatically and its associated code run whenever the Calculate button is clicked. Edit this by adding the following code:

```
   procedure TForm1.CalcBtnClick(
Sender: TObject);
var
  st, vt, gt  : real;
  errcode : integer;
begin
  Val(SubTotal.Text, st, errcode);
  if errcode = 0 then
  begin
    vt := (st * 0.175);
    gt := vt + st;
    Vat.Text := FloatToStr(vt);
    GrandTotal.Text:= FloatToStr(gt);
  end;
end;
```

DELPHI SYNTAX

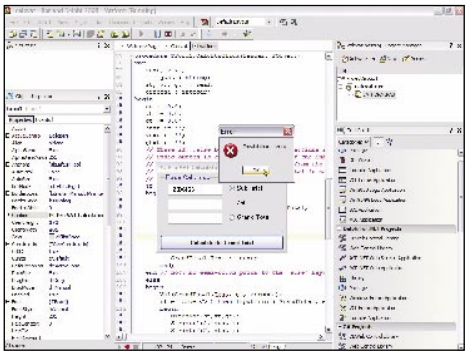
Before attempting to run this project, check your code carefully. Delphi's Object Pascal language requires that each variable be declared in the VAR section beneath the procedure header. Variables of the same type may be separated by commas, followed by a colon, the type name (for example, integer) and a semi-colon. Each procedure starts with the `begin` keyword and ends with the `end` keyword followed by a semi-colon. These two keywords perform a similar function to the curly-brace delimiters {} in C-like languages and Java. Code blocks are also delimited by `begin` and `end`. Expressions are terminated by semi-colons. Note that Pascal isn't case-sensitive, so you can mix cases at will.

We use the standard `Val()` procedure to convert the string `SubTotal.Text` to a real number, st. If this fails, the integer variable errcode returns the index of the problem character in the string. The `FloatToStr()` function returns the string representation of a floating-point value. You can find out more about Delphi's standard functions by highlighting the function name in the editor and pressing F1 to display help. The value returned by `FloatToStr()` can be assigned to a string variable or property using Pascal's ':=' (colon-equals) assignment operator. Note that a single equals sign in Pascal is used to test equality as in the expression `if errcode = 0`. When assigning a value, a colon must precede the equals sign.

Run the program. If there are any syntax errors, Delphi displays error messages at the bottom of the screen and highlights the erroneous line of code. Fix any errors, then run the program again. If you still have problems, you can load our test1.dpr project all ready to go on the cover DVD. Try entering a numerical value into the first edit box, SubTotal. When you click the button, the code above will be executed. Assuming there's no error in your input, the code converts the string in the SubTotal box to a floating-point value, multiplies this by 0.175 to obtain a 17.5 per cent VAT value and adds this to the original value to obtain the VAT-inclusive grand total. The values are then converted back to strings and displayed in the edit boxes Vat and GrandTotal.

MAKING REFINEMENTS

As it stands, this program does nothing when it encounters an input error. It would be more civilised if it at least produced a meaningful error message. We've rewritten the project to add this feature in the Calctest.dpr project, also on the cover DVD. It would be better still if the user could optionally calculate a subtotal from a grand total. We've added this feature to the project too. These projects also introduce some more complex Object Pascal coding techniques, which are described in comments in the source code. Take some time to study them and you'll get a better feel for Delphi's object-orientation features.



Our finished VAT calculator adds or subtracts VAT from a value and deals with user input errors.

PORTING AN APPLICATION TO .NET

Up to now we've been developing Win32 applications, which is the traditional method for Windows visual programs. However, what happens if we decide we'd like to port our applications to run under the Microsoft .NET Framework? The .NET Framework has a huge library of classes and routines, which hide and extend the capabilities of the standard Windows API.

Delphi Win32 projects use a different library called the VCL (Visual Component Library), which is, unfortunately, incompatible with the .NET Framework. Borland's solution to this problem has been to write a new version of the VCL to interface with .NET. This means that, in many cases, we can instantly convert an old Win32 Delphi program to run under .NET. There are some things that can't be automatically converted and will require recoding; these are documented in the Help system.

In order to convert our CalcVat program, you must first delete the Borland Development Studio Project file, CalcVat.bdsproj. We've supplied a copy of this project in the \calcvatToNET directory so that you can delete this file without any worries about damaging the original project. Now load the Win32 Delphi project file Calcvat.dpr into Delphi. A message appears, asking if you'd like to upgrade the project for .NET or Win32. Select Delphi For .NET. The project is immediately converted. You now have your first ready-to-run .NET application.

GOING FURTHER

If we've managed to whet your appetite for Delphi programming, you may want to try out some of the sample projects that you'll find in the \Demos subdirectory beneath your \Borland\BDS installation directory. Initially, we suggest you stick to the \DelphiWin32 projects. There are additional tutorial resources available from the Welcome screen, which can be selected from a tabbed page.

Next month: We'll continue to look at Delphi 2005 with a guide to developing applications for the .NET Framework.