

*Delphi advanced programming technology*



# Chapter 1

## INTRODUCING DELPHI

Professor Zhaoyun Sun





# Outline

---

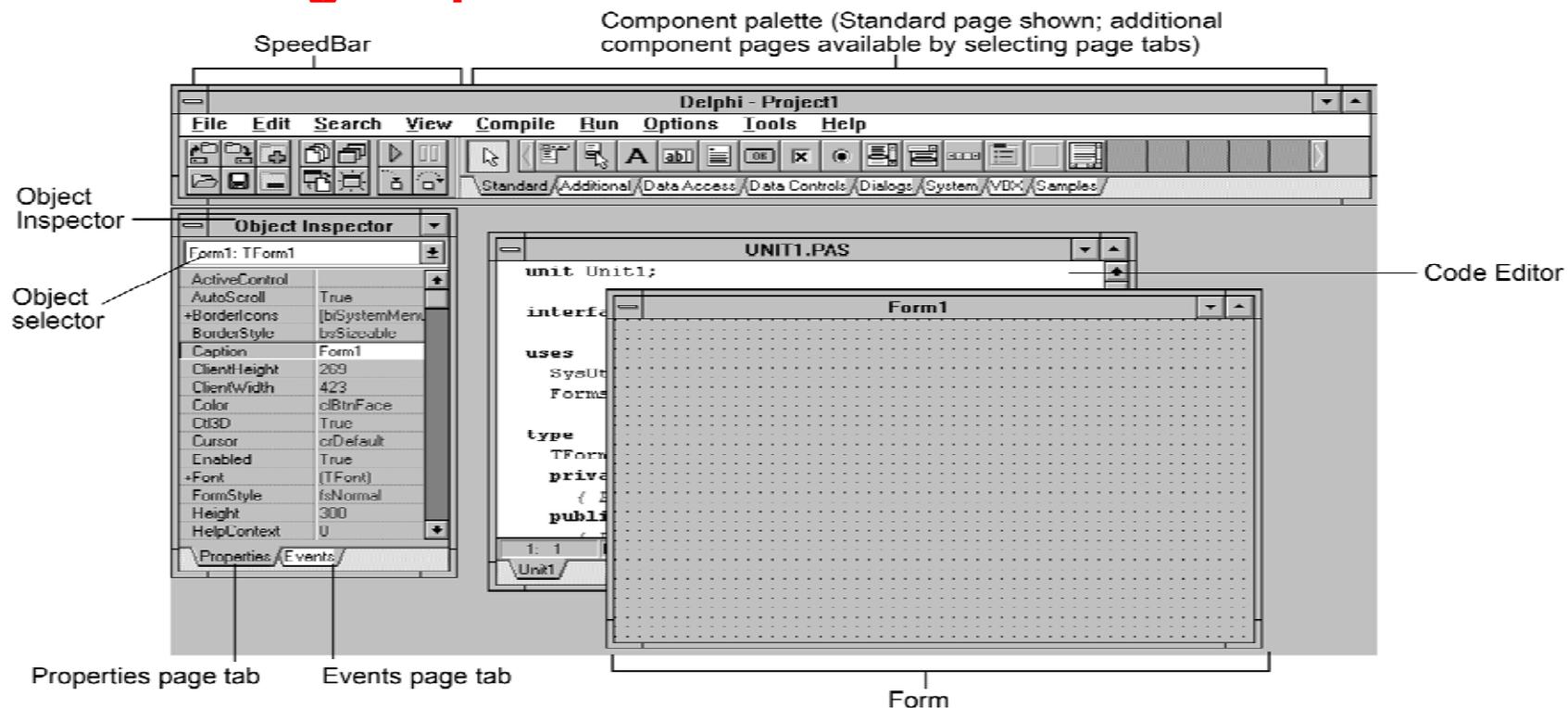
- The Delphi programming environment
- Elements of the Delphi interface
- The Delphi development model
- Overview of Delphi projects
- Setting environment preferences





# 1.1 The Delphi Programming Environment

## □ Starting Delphi



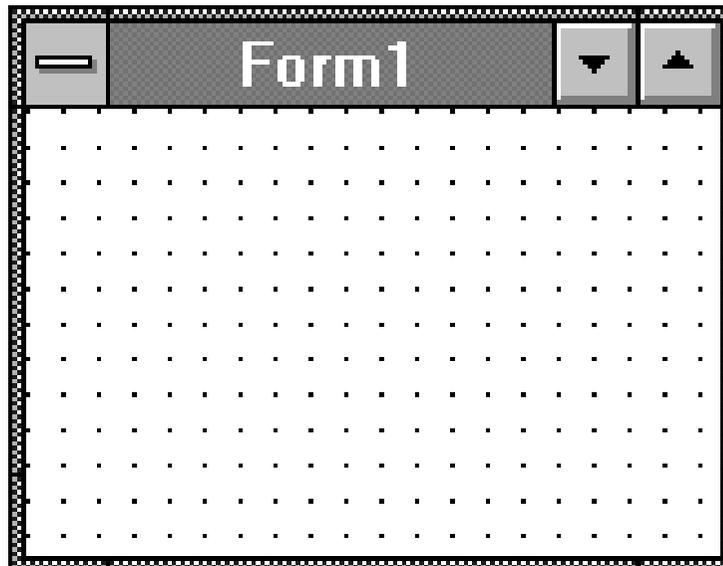
The Delphi programming environment

Department of Electric Information Engineering





## 1.2 Elements of the Delphi interface



Form

### □ Forms

■ Forms are the focal point of nearly every application you develop in Delphi. **You use the form** like a canvas, placing and arranging components on **it to design** the parts of **your user interface**.

■ **Components** are the building blocks of Delphi applications. They **appear on the Component palette**, displayed in the top right-hand part of the screen.





## Elements visible upon starting Delphi

### □ Forms

■ You can **think of a form as a component** that can contain other components. **Your application's main form and its components interact with other forms** and their components to create your application interface.

■ **The main form is your application's main interface**; other forms can include dialog boxes, data entry screens, and so on.





# Elements visible upon starting Delphi

## Forms

- You can resize the form and move it anywhere on your screen. A form includes standard features such as

- Control menu
- Minimize and maximize buttons
- Title bar
- Resizeable borders





## Elements visible upon starting Delphi

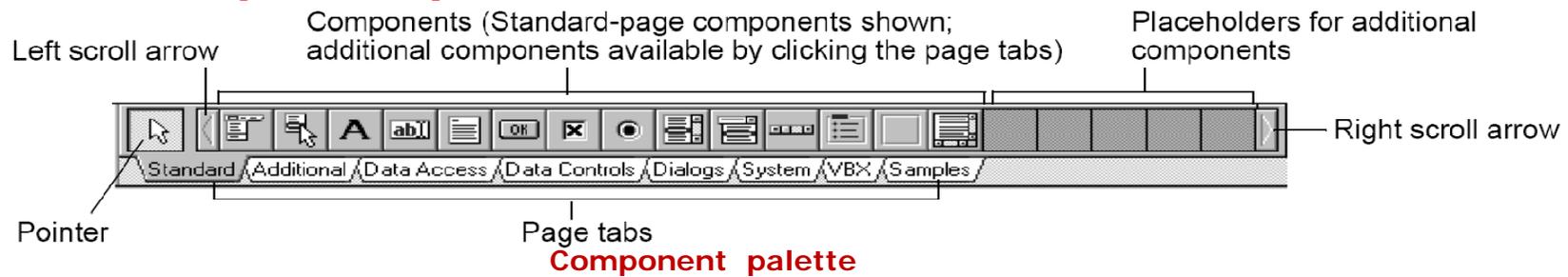
- You can change these features, as well as other properties of the form by using the **Object Inspector** to edit the form during design time—the time during which you are designing, rather than running, your form.
- Properties define a component's **appearance** and **behavior**.





# Elements visible upon starting Delphi

## □ Component palette



■ **Components are the elements** you use to build your Delphi applications. They include all the **visible parts** of an application, **such as dialog boxes and buttons**, as well as those that **aren't visible** while the application is running, such as **system timers** or **Dynamic Data Exchange (DDE) servers**.





## Elements visible upon starting Delphi

■ You can **create your own** custom **components** and **install them** onto the Component palette, making the Delphi environment fully extensible

### □ **Object Inspector**

■ The Delphi Object Inspector **enables you to easily customize the way a component appears and behaves in your application.** The properties and events of the component that is selected in the form are displayed in the Object Inspector.





## Elements visible upon starting Delphi

- You use the **Properties page** of the Object Inspector **to customize components** you've placed on a form (or the form itself), and the **Events page to generate and navigate** among certain parts of program code, called ***event handlers***. Event handlers are specialized procedures.





# Elements visible upon starting Delphi

## ❑ Object Inspector

The screenshot shows the Object Inspector window for a form named 'Form1: TForm1'. The window is divided into two main sections: 'Properties' and 'Events'. The 'Properties' section is currently active and displays a list of properties for the selected object. The properties are listed in two columns, separated by a vertical line. The first column contains the property name, and the second column contains the property value. The 'Font' property is expanded, showing its subproperties. The 'Events' section is currently inactive.

Property Name	Value
ActiveControl	
AutoScroll	True
+BorderIcons	[biSystemMenu]
BorderStyle	bsSizeable
Caption	Form1
ClientHeight	269
ClientWidth	423
Color	clBtnFace
Ctl3D	True
Cursor	crDefault
Enabled	True
+Font	(TFont) ...
FormStyle	fsNormal
Height	300
HelpContext	0

Annotations:

- Object selector (shows name and type of selected object)
- Movable column separator (drag horizontally to resize columns)
- Value column
- Nested property (double-click to view subproperties)
- Events page

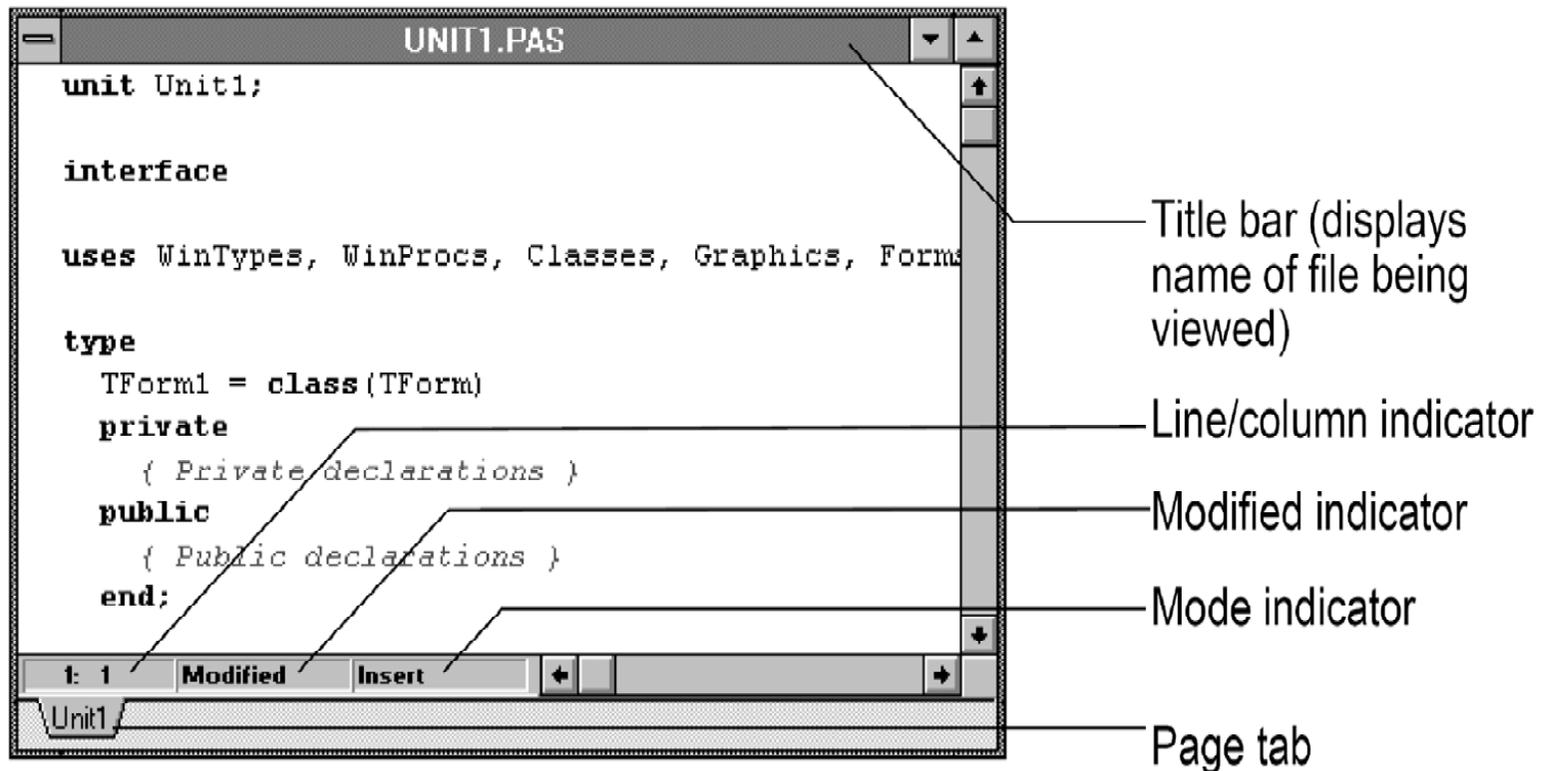
Properties page





# Elements visible upon starting Delphi

## □ Code Editor





## Elements visible upon starting Delphi

### Code Editor

■ The Delphi **Code Editor** is a full-featured **editor** that provides access to all the code in a given application project.

■ The Code Editor **includes many powerful features** such as Brief-style editing, color syntax highlighting, and virtually unlimited **Undo**.





## **Elements visible upon starting Delphi**

### **□ Code Editor**

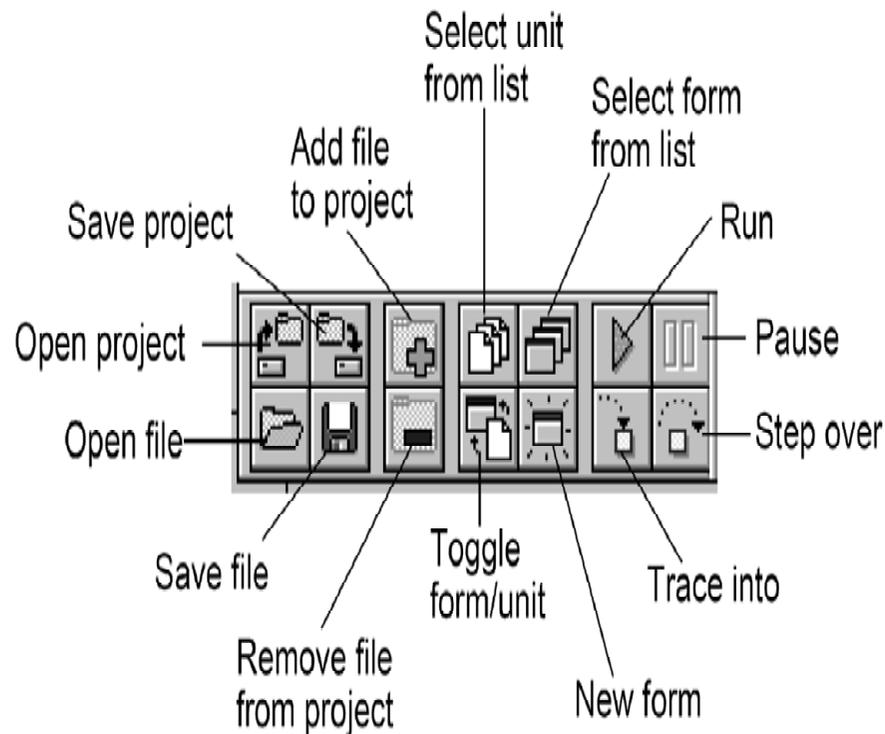
- When you open a new project, Delphi generates a page in the Code Editor for **the unit source code (.PAS) file**.**
- To view the source code for a particular unit, simply click that file's page tab. The Code Editor **title bar displays the name of the file** in the active page of the Code Editor.**





# Elements visible upon starting Delphi

## □ SpeedBar



■ The Delphi **SpeedBar**, in its default state, **provides** you with **shortcuts** to some of the more **common commands** from the **File, Edit, View, and Debug** menus.





## Elements not visible upon starting Delphi

- The following **elements** of the Delphi interface **are not visible** when you first start Delphi, but **you can access them quickly from the menu bar.**



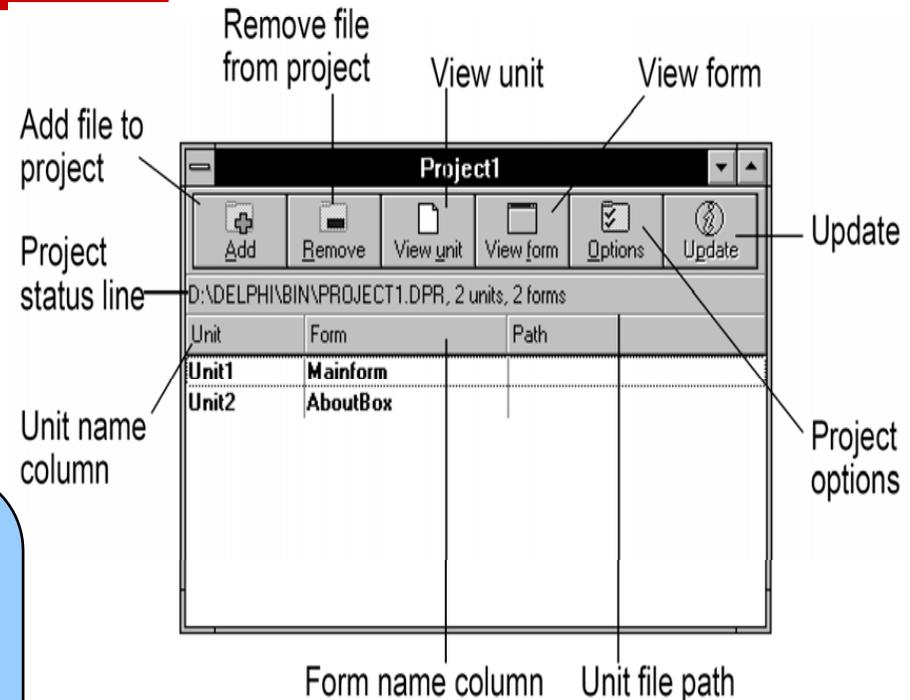


# Elements not visible upon starting Delphi

## □ Project Manager

■ The Delphi Project Manager lists the files that make up your application, and enables you to easily navigate among them.

■ You can use buttons on the Project Manager SpeedBar to generate new forms and units, to view files in the current project, and to save modifications to all opened project files.



To display the Project Manager, choose **View|Project Manager**.

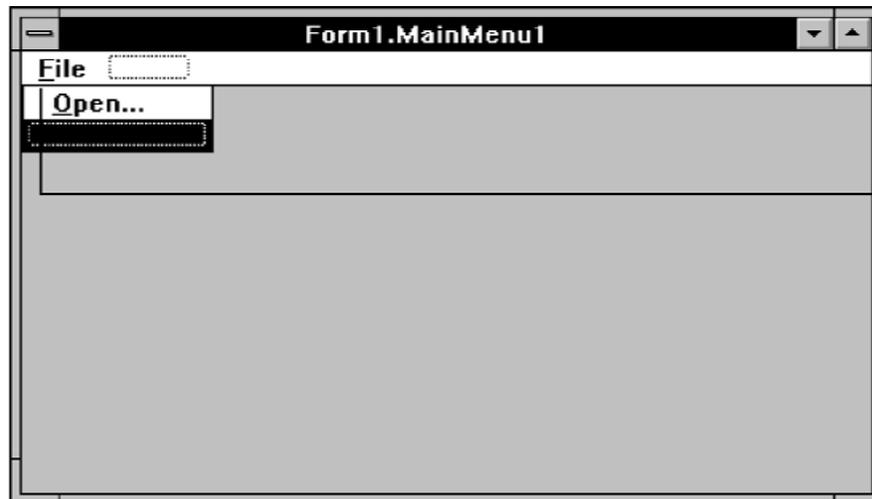




# Elements not visible upon starting Delphi

## ❑ Menu Designer

- The Delphi Menu Designer enables you to easily add menus to your forms.



Menu item  
(Menu commands appear  
below menu items)





## Elements not visible upon starting Delphi

### Integrated debugger

- Delphi **provides** a fully **integrated debugger** so you can debug your source code without exiting the development environment.





## **1.3 The Delphi development model**

- This section describes the fundamental steps involved in developing your own custom projects with Delphi.**





## Designing a form

---

□ This section demonstrates the following concepts:

- **Creating a new form**
- **Adding components to the form**
- **Setting component properties**
- **Running the program**





# Designing a form

The screenshot shows the Object Inspector window for a form named 'Form1: TForm1'. The 'Caption' property is selected and highlighted in blue. The window is divided into two columns: the left column lists properties, and the right column shows their values. At the bottom, there are two tabs: 'Properties' (selected) and 'Events'. Labels with arrows point to various parts of the window: 'Property column' points to the left column, 'Caption property setting for Form 1' points to the 'Caption' row, 'Value column' points to the right column, and 'Page tabs' points to the bottom tabs.

Property	Value
ActiveControl	
AutoScroll	True
+BorderIcons	[biSystemMenu]
BorderStyle	bsSizeable
Caption	Form1
ClientHeight	269
ClientWidth	423
Color	clBtnFace
Ctl3D	True
Cursor	crDefault
Enabled	True
+Font	(TFont) ...
FormStyle	fsNormal
Height	300
HelpContext	0

**Object Inspector with Caption property selected**





# Handling events

The screenshot shows the 'Object Inspector' window for 'Form1'. It displays a list of events in the 'Handler column' and a corresponding 'Value column'. The 'Handler column' lists events such as OnActivate, OnClick, OnClose, OnCloseQuery, OnCreate, OnDbClick, OnDestroy, OnDragDrop, OnDragOver, OnEnter, OnExit, OnKeyDown, OnKeyPress, OnKeyUp, and OnMouseDown. The 'Value column' is currently empty. The window has tabs for 'Properties' and 'Events' at the bottom.

Handler column	Value column
OnActivate	
OnClick	
OnClose	
OnCloseQuery	
OnCreate	
OnDbClick	
OnDestroy	
OnDragDrop	
OnDragOver	
OnEnter	
OnExit	
OnKeyDown	
OnKeyPress	
OnKeyUp	
OnMouseDown	





## Handling events

---

- ❑ **Events represent user actions** (or internal system occurrences) that your application can recognize.
- ❑ **The Events page of the Object Inspector displays all events associated with the selected component.**





## **Starting a new project**

---

- You can **use** the **New Project** command from the **File** menu **to start a new application project**, or to open any of the template applications provided with Delphi.

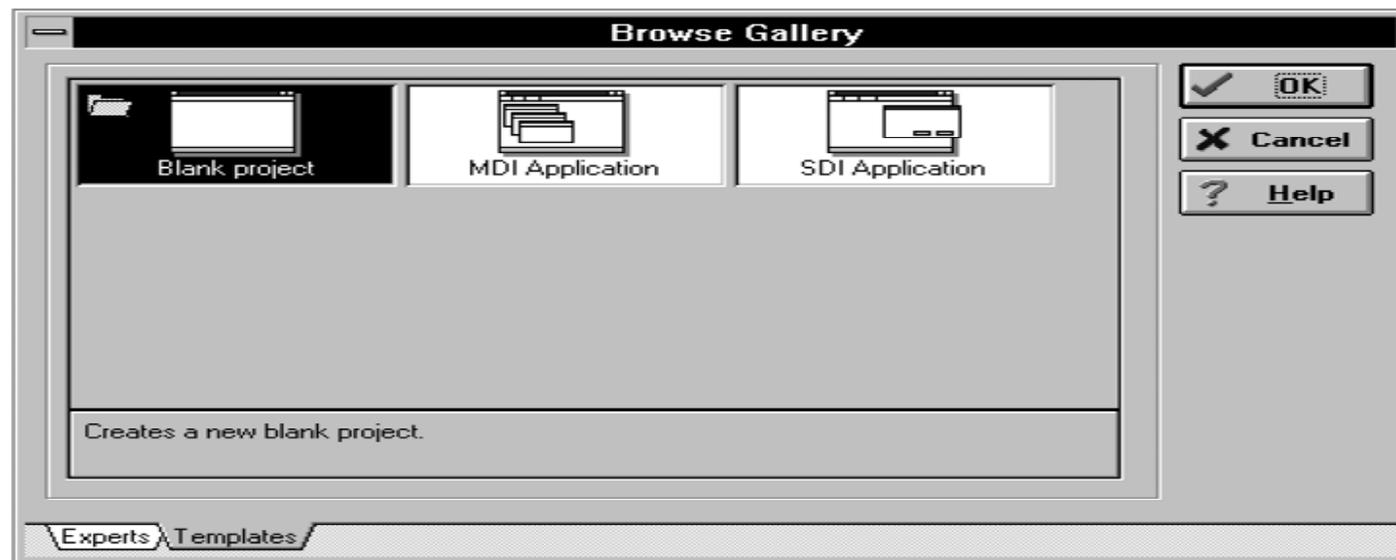




## Starting a new project

---

- To start this project, choose **File|New Project**



Browse Gallery dialog box displaying application templates





## Starting a new project

---

- **Calling procedures and functions from event handlers**

■ **So far, this chapter has discussed **how to set properties at design time and run time.** However, setting properties is only a minor aspect of code development in Delphi.**





## Starting a new project

---

- Calling procedures and functions from event handlers

■ ***Procedures* and *functions***, also known as routines, usually **constitute** the bulk of your program code.





## Starting a new project

---

- **Calling procedures and functions from event handlers**

- **For the sample application, write the following two event handlers:**

- **Select *AddBtn*, generate an *OnClick* event handler and type the statement shown inside the **begin..end** block:**

```
procedure TForm1.AddBtnClick(Sender: TObject);  
begin  
Listbox1.Items.Add(Edit1.Text);  
    {add this line of code}  
end;
```





## Starting a new project

---

- Calling procedures and functions from event handlers

- The code you typed calls the *Add* method of the list box in response to a click on the *AddBtn*. The parameter being passed is the *Text* property of *Edit1*. This specifies that when the user clicks *AddBtn*, any *text* in *Edit1* is *added to the items in the list box*.





## Starting a new project

---

- Calling procedures and functions from event handlers

- Now generate the *OnClick* event handler for *ClearBtn*, and type the statement shown inside the following **begin..end** block:

```
procedure TForm1.ClearBtnClick(Sender: TObject);  
begin  
  ListBox1.Items.Clear;  {add this line of code}  
end;
```





## Starting a new project

---

- ❑ Calling procedures and functions from event handlers

■ This code calls the *Clear* method of the list box in response to the *ClearBtn* click. As you might guess, the *Clear* method clears the text from the list box.





## **1.4 Overview of Delphi projects**

- When you create a Delphi application, you can start with a blank project, an existing project, or one of Delphi's application or form templates. A project consists of all the files needed to create your target application.**





## **1.4 Overview of Delphi projects**

- **This section introduces you to the “core” files in a Delphi project. It discusses the following topics:**
  - **The project file (.DPR)**
  - **The unit file (.PAS)**
  - **The form file (.DFM)**
  - **Source code for units without forms**





## **The project (.DPR) file**

---

- For each application you develop in Delphi, **there is one project (.DPR) file that keeps track of all the unit and form files** in the application project.
- When you begin a new project, **Delphi generates the project file, and maintains this file** throughout the development of the project.





## The unit (.PAS) file

---

- The **unit file** is the Object Pascal **source code file**, saved with a **.PAS** extension.





## **The form (.DFM) file**

---

- **The form is the focal point for programming in Delphi. Whether you're **adding components to the form, changing their properties** using the Object Inspector, or **typing code** in the Code Editor for the unit associated with the form, you're really editing the form.**





## **Source code for units without forms**

- Although most Delphi units are associated with forms, you may want to create or use units that have no forms associated with them.**
- For example, you might create a separately compilable unit of nonvisual objects, or import a library of certain mathematical functions.**





## Source code for units without forms

- This is the only code that Delphi generates when you **add a new unit to a project** without adding a form:

```
unit Unit2;  
  
Interface  
  
implementation  
  
end
```





## 1.5 Setting environment preferences

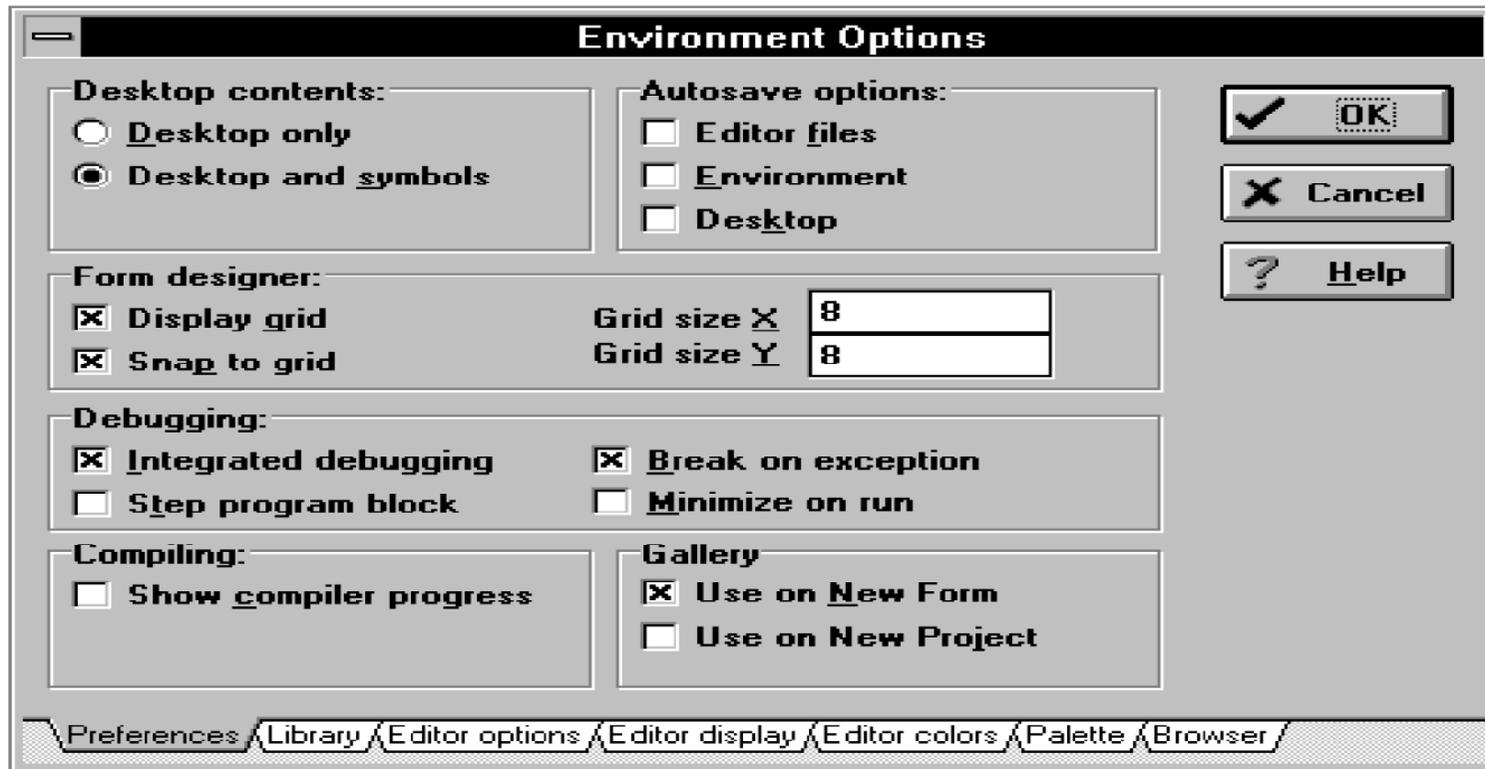
### Accessing environment options preferences

- To display the Environment Options dialog box, choose **Options|Environment** from the Delphi menu bar, then choose the Preferences page tab.





# Accessing environment options preferences



Environment Options dialog box, Preferences page

