

## Chapter Eight: Part Three of Another Controls

### 8.1 The Line Control

To draw a straight line, just click on the line control and then use your mouse to draw the line on the form.

#### 8.1.1 Line Tool Properties

<b>BorderColor</b>	Determines the line color.
<b>BorderStyle</b>	Determines the line 'shape'. Lines can be transparent, solid, dashed, dotted, and combinations.
<b>BorderWidth</b>	Determines line width.

#### Note:-

There are no events or methods associated with the line tool.

### 8.2 Shape Control

To draw a shape, just click on the shape control and draw the shape on the form. The default shape is a rectangle, with the shape property set at 0. You can change the shape to square, oval, circular and rounded rectangle by changing the shape property's value to 1, 2, 3, 4 and 5 respectively.

#### 8.2.1 Shape Tool Properties

<b>BackColor</b>	Determines the background color of the shape (only used when FillStyle not Solid).
<b>BackStyle</b>	Determines whether the background is transparent or opaque.
<b>BorderColor</b>	Determines the color of the shape's outline.
<b>BorderStyle</b>	Determines the style of the shape's outline. The border can be transparent, solid, dashed, dotted, and combinations.

<b>BorderWidth</b>	Determines the width of the shape border line.
<b>FillColor</b>	Defines the interior color of the shape.
<b>FillStyle</b>	Determines the interior pattern of a shape. Some choices are: solid, transparent, cross, etc.
<b>Shape</b>	Determines whether the shape is a square, rectangle, circle, or some other choice.

**Note:-**

Like the line tool, events and methods are not used with the shape tool.

**Example (1):-**The program allows the user to change the shape selectin a particular shape from a list of options from a list box. Write the code of list box to select the shape you want by using select case.

**Solution:-**

```
Private sub form_load()

List1.AddItem"rectangle"

List1.AddItem"square"

List1.AddItem"oval"

List1.AddItem"circle"

List1.AddItem"rounded rectangle"

List1.AddItem" rounded square"

End Sub
```

.....

```
Private sub list1_click()
```

```
Select case list1.listindex
```

```
Case0:Shape1.Shape = 0
```

```
Case1:Shape1.Shape = 1
```

```
Case2:Shape1.Shape = 2
```

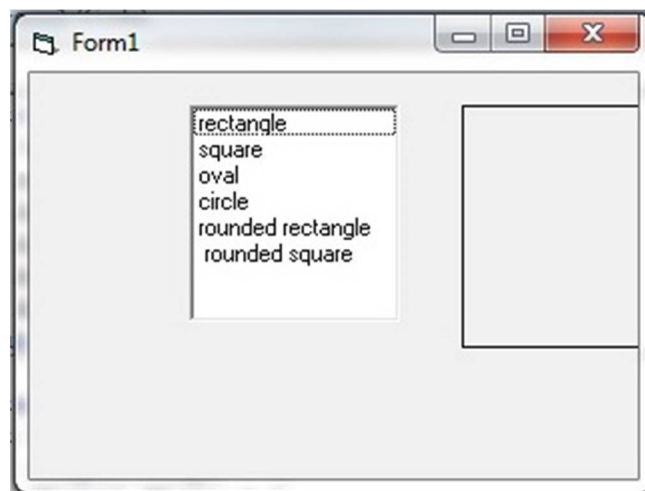
```
Case3:Shape1.Shape = 3
```

```
Case4:Shape1.Shape = 4
```

```
Case5:Shape1.Shape = 5
```

```
End Select
```

```
End Sub
```



### 8.3 Timer Control

The timer control allows you to perform a task (code) after each interval of time determined by the user is passed.

You will see the timer control only during design time because you need to be able to select the control and change its properties. Figure below shows the timer control as it appears when you place the control on a form. The timer control supports only several properties because the timer control never appears on the form at runtime. Therefore, the control has on need for many of the style and size properties used for other controls the user sees.

#### 8.3.1 The characteristics about the timer control

1. It is not visible to the user at run time.
2. The actions that you want to happen at the specified interval are coded in the timer's timer event procedure.
3. You determine the interval that you want actions to occur in the timer control's interval property. The value is specified in milliseconds (1000 milliseconds = 1).
4. You turn the timer "on" or "off" by setting its enabled property to true or false, respectively. The enabled property is set to true by default.

#### 8.3.2 The Timer Control Properties

property	Description
<b>Enabled</b>	Specifies whether the timer control can respond to events.
<b>Interval</b>	Determines the timer interval, in milliseconds, when you want a timer event to occur. The interval value can range from 1 to 65535.
<b>Left</b>	Specifies the number of twips from the left edge of the form window where the timer control resides.

<b>Name</b>	Contains the name of the control. Unless you change the name, visual basic assigns the default names of timer1, timer2, and so on as you add timer controls.
<b>Top</b>	Specifies the number of twips from the top edge of the from window where the timer control resides.

**Note:-**

Generally, the most important timer control property is the interval property. The interval property is especially important for determining the execution of the timer event procedures that you write for the timer control.

**8.3.3 Timer Event**

The timer control has one event called timer event. This event is executed every interval value (millisecond) passes. The procedure timer continues this until we disable the control by setting the enabled property to false.

**Example (2):-** Write the program in VB contain one form, two commands, label1 and timer1 write the code of:-

1. Command1\_click event to make the timer off.
2. Command2\_click event to make the timer on.
3. Timer1\_Timer event to change the back color of label1 by every three second by different color.

<b>Command1</b>	
Name	cmdstart
Caption	Start timer
<b>Command2</b>	

Name	cmdstop
Caption	Stop timer
Label1	
Caption	empty
Timer1	
Interval	3000

Dim C As Integer

Private sub cmdstart\_click()

C=0

Form1.cls

Timer1.Enabled =True

End Sub

.....

Private sub cmdstop\_click()

Timer1.Enabled =False

End Sub

.....

Private sub Timer1\_Timer()

C = C + 1

Label1.BackColor = vbcolor(C)

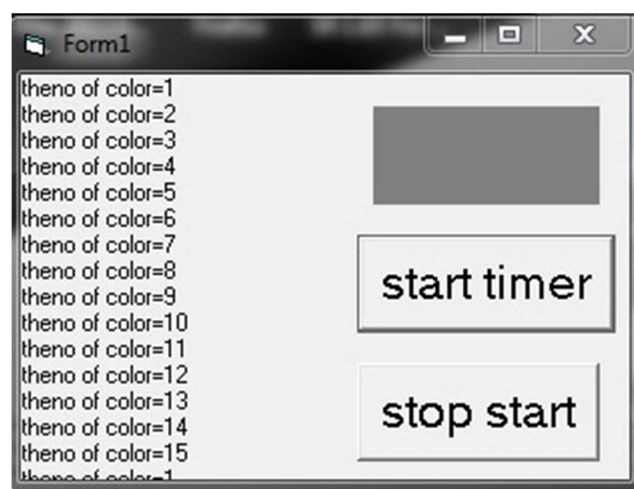
```
Print"the no of color ="& C
```

```
If C 15 then C =0
```

```
End If
```

```
End Sub
```

The program after running as shown below.



#### 8.4 Data Control

A **database** file is a collection of database tables. A **table** is nothing more than rows and columns. Columns are referred to as **fields** and always have **field names**. An entire row is one **record**. Each individual table in a database contains related information, but it's not uncommon for a database to consist of just one table. Designing a new database takes careful planning. For optimal performance in multiple table databases, related fields from different tables are linked by having the same field name.

The Microsoft Jet database engine, supplied with Visual Basic, gives you the ability to access many types of databases-Microsoft Access databases; other PC-based databases such as dBASE, FoxPro. Visual Basic provides two basic techniques for

working with the Jet database engine: the **data control** and the **data access objects (DAO)**. The **data control** requires less code, but **data access objects** are much more flexible.

Now we explain the first method that deal it with database

### **Data control object**

The **Data control** provides a means of quickly developing database applications with little or no code, but it limits your access to the underlying database.

Your project today is to create a completely functional data entry program using Visual Basic. The program you create will be able to access data tables within an existing database; it will also allow users to add, edit, and delete records.

#### **8.4.1 Create the database in Microsoft access system**

First we go to the access system to create database and create the tables the need to store the data within an existing this database. See the figure 1.



Figure 1



In our example the database is called **student** and contains one table called **info**. The table **info** contains the following fields as shown below. See the figure 2.



اسم الحقل	نوع البيانات	لوصف
name	نص	
dept	نص	
age	رقم	
average	رقم	
comment	نص	

خصائص الحقل

Figure 2

After design the table info as shown above, we will enter the data to the fields by clicking the command فتح as shown in fig1 above.

#### 8.4.2 Adding the Database Control

The first thing you need to do for the database program is open up the database and select the data table you want to access. To do this, double-click the **data control** in the Visual Basic toolbox (see Figure 3). This places a data control in the center of the form. When this is done, the form is ready to open a data table. At this point, your screen should look something like the one in Figure 3.

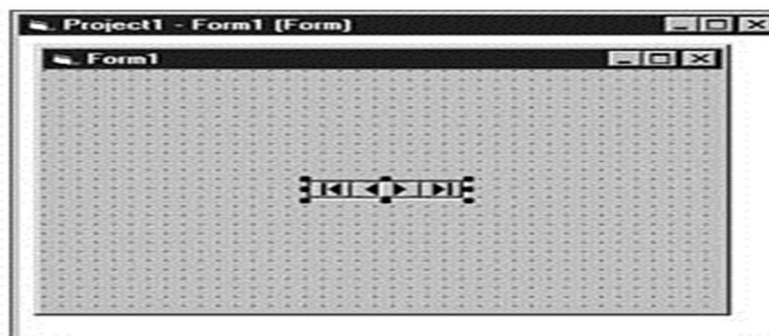


Figure 3

### 8.4.3 Setting the DatabaseName and RecordSource Properties

You must first set the following two properties when linking a data control to a database:

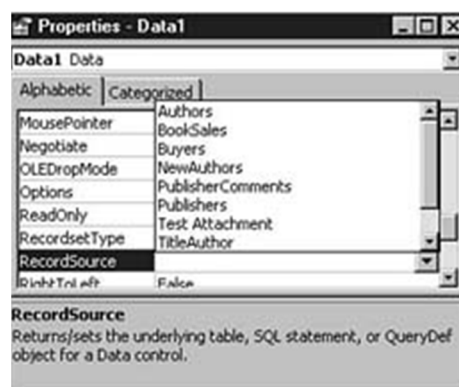
DatabaseName	Selected database
RecordSource	Selected data table in the database

The student.MDB database will be used in the exercise that follows.

To set the DatabaseName of the data control, first select the data control by single-clicking the control. This forces the data control properties to appear in the Visual Basic Properties dialog box. Locate the **DatabaseName** property, and click the property name. When you do this, three small dots (. . .), the properties ellipsis button, appear to the right of the data entry box. Clicking the ellipsis button brings up the Windows standard File | Open dialog box. You should now be able to select the **student.MDB** file from the list of available database files from the list of available database files (\\c:\my document \student.MDB). When you have located the **student.MDB** file and selected OK, Visual Basic inserts the complete drive, path, and filename of the database file into the input area, linking the database and your program together. Always double-check this property to make sure that you correctly selected the desired database.

Now that you know what database you will use, you must select the data table within that database that you want to access by setting the **RecordSource** property of the data control. You can do this by locating the RecordSource property in the Properties window, single-clicking the property, and then single-clicking the small down arrow to the right of the property input box. This brings up a list of all the tables in the

student . MDB database, you will use the info data table in the student.MDB database.



To select the Titles table from this list, simply click on it. Visual Basic automatically inserts the table name into the RecordSource property in the Properties window.

#### 8.4.4 Setting the Caption and Name Properties

You need to set two other data control properties in the project. These two properties are not required, but setting them is a good programming practice because it improves the readability of the programming code. Here are the optional properties:

Caption	Displayed name of the data control
Name	Program name of the data control

Setting the Caption property of the data control sets the text that displays between the record selections arrows on the data control. (Please note that you will need to

expand the width of the data control to read this text.) It is a good habit to set this to a value that makes sense to the user.

Setting the Name property of the data control sets the text that will be used by the Visual Basic programmer. This is never seen by the user.

For your program, set the **Caption property** of the **data control** to **student information** and the **Name property** of the **data control** to **datinfo**.

#### **8.4.5 Adding the Bound Input Controls (textbox)**

Now that you've successfully linked the **form** to a **database** with the **data control** and selected a data table to access, you are ready to add **input controls** to the **form**. Visual Basic 6 supplies you with **input controls** that can be directly bound (connected) to the **data table** you want to access. All you need to do is place several **input controls** on the **form** and assign them to an existing data control.

#### **Note:-**

Associating a control on a form to a field in a data table is referred to as **binding a control**. When they are assigned to a data source, these controls are called **bound input controls**.

Let's add the first bound input control to the **info table** input form. Place an input control on the form by double-clicking the **textbox control** in the Visual Basic toolbox. You could add additional input controls by double-clicking the textbox button in the toolbox as many times as you like. Set the Name property of this control. Add a label to describe this control by double-clicking the Label control. Set the label's caption property. In our example we need to five textbox (input

control) and five labels because the number of fields in the **info table** are five. The table below show the properties of textboxes and labels

<b>Text1</b>		<b>label1</b>	
Name	txtname	caption	the name of student
Text	blank	<b>label2</b>	
<b>Text2</b>		caption	the dept
Name	txtdept	<b>label3</b>	
Text	blank	caption	the age of student
<b>Text3</b>		<b>label4</b>	
Name	txtage	caption	the average
Text	blank	<b>Label5</b>	
<b>Text4</b>		caption	the comment
Name	txtaverage		
Text	blank		
<b>Text5</b>			
Name	txtcommend		
Text	blank		

#### 8.4.6 Setting the DataSource and DataField Properties for textbox control

You must set two textbox properties in order for the textbox control to interact with the data control. These are the two required properties:

DataSource	Name of the data control
DataField	Name of the field in the table

A relationship is established between a field (the DataField property) in a table (the DataSource property) and a bound control when you set these two properties. When

this is done, all data display and data entry in this input control is linked directly to the data table/field you selected.

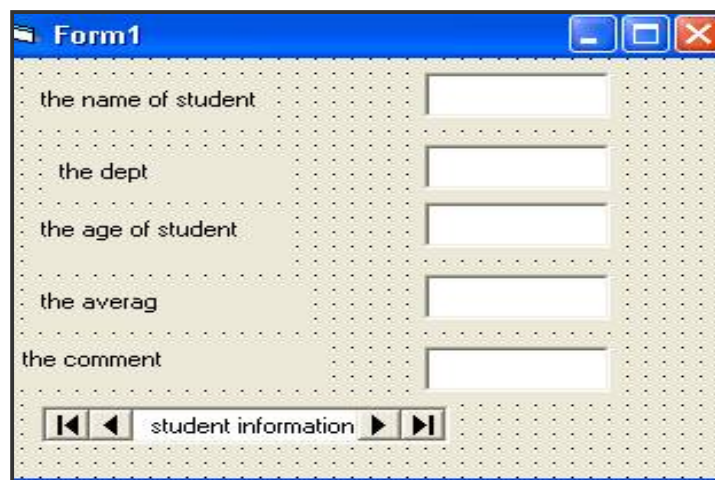
Setting the DataSource property of the textbox control binds the input control to the data control. To set the textbox DataSource property, first select the textbox control (click it once), and then click the DataSource property in the Property window. By clicking this property's down arrow, you can see a list of all the data controls currently active on this form. You have only added one data control to this form, so you see only one name in the list . Set the DataSource value to datinfo by clicking the word datinfo in the drop-down list box.

The second required property for a bound input control is the DataField property. Setting this property binds a specific field in the data table to the input control. Set the DataField property of the current input control by single-clicking the DataField property in the Property window and then single-clicking the down arrow to the right of the property. You now see a list of all the fields that are defined for the data table that you selected in the DataSource property. The tabel below show the DataSource and DataField properties for input control (textbox).

**Table 1. The Input Control DataSource and DataField properties for the Titles form.**

Textbox	DataSource	DataField
txtname	datinfo	name
txtdept	datinfo	dept
txtage	datinfo	age
txtaverage	datinfo	average
txtcomment	datinfo	comment

The figure below show the completed data entry form for info table.

The image shows a screenshot of a Visual Basic form titled "Form1". The form has a light gray background with a dotted grid pattern. It contains five text boxes arranged vertically, each with a label to its left: "the name of student", "the dept", "the age of student", "the averag", and "the comment". At the bottom of the form, there is a data control consisting of four arrows (left, right, left, right) and the text "student information" in the center.

You can now run the program and see the data control in action. You can walk through the data table by clicking the left and right arrows on the data control at the bottom of the form. The left-most arrow (the one with the bar on it) moves you to the first record in the data table. The right-most arrow (which also has a bar) moves you to the last record in the data table. The other two arrows simply move you through the data table one record at a time.

#### 8.4.7 Adding the New and Delete Command Buttons

Up to this point, you have not written a single line of Visual Basic code. However, in order to add the capability to insert new records and delete existing records, you have to write a grand total of two lines of Visual Basic code: one line for the add record function, and one line for the delete record function.

The first step in the process is to add two command buttons labeled **Add** and **Delete** to the form. Refer to Table 1.3 and Figure 1.8 for details on adding the command buttons to your form.

Table 2. Command Button properties for the Title form.

<i>Name</i>	<i>Caption</i>
cmdAdd	&Add
cmdDelete	&Delete

The form layout after adding the Add and Delete command buttons. Double-click the Add button to bring up the Visual Basic code window to add code behind the Add command button. You see the subroutine header and footer already entered for you. All you need to do is add a single line of Visual Basic code between them.

```
Private Sub cmdAdd_Click()
    datInfo.Recordset .AddNew           ` add a new record to the table
End Sub
```

Now open the code window behind the Delete button and add this Visual Basic code:

```
Private Sub cmdDelete_Click()
    datTitles.Recordset . Delete       ` delete the current record
End Sub
```