Visual C# .Net using framework 4.5

Eng. Mahmoud Ouf

Lecture 06

Controls

Add Control to form Programmatically:

1. Define an object from the control
2. Create an object from the control
3. Set the object properties
4. Add the object to the form

Example: Add a button to a form:

```
1.  Button btn;
2.  btn = new Button();
3.  btn.Text = “Press the Button”;
    btn.Location = new Point(80, 80);
4.  this.Controls.Add(this.btn);
```

Controls

Add Control to form Using the Windows Forms Designer:

1. From the ToolBox, Drag the Control that you want to add to the form and place it in the Location you need it.
2. The windows Form designer will generate the code.
3. The windows Form designer usually put the code that it generates in a function named InitializeComponent(), and this is the function to be called in the form constructor

Setting the control property:

From Design View:

Select the control, set the property from the property window

Programmatically:

Write the control_name.the property_name = value

Events

All the GUI classes, have their own events (they are ready made classes).
The part of the Sender is already implemented, all I have to do is to implement the part of the Listener (attach the events of the sender with a firing methods through the delegate, and implement these methods)
In Windows programming, you can handle the events in 1 of 2 choices:
First: to use the delegate and the events
Second: to override the appropriate base class method
To Handle events you have to know 2 things:

1) The event itself
2) The delegate associated with this event

The Form class Events

The most important event in the **Form class** is the Paint event which occurs when a form is redrawn.

While this event is in the **Control class** and can be used with any class inherit from it, but it is widely used with the **Form class**

The event: Paint

The delegate:

public delegate void **PaintEventHandler**(object sender,PaintEventArgs e)

object: which represents the object sending the event

PaintEventArgs: which contains some relevant information such as object to Graphics class

The Form class Events

First:

In the constructor:

```
this.Paint += new PaintEventHandler(Form_Paint);
```

out of constructor, define the Form_Paint()

```
public void Form_Paint(object sender,PaintEventArgs e)
{
    Graphics g = e.Graphics;
    g.DrawString("Hello", new Font("Times New Roman",15,
                    new SolidBrush(Color.Black), 40, 10);
}
```

Second:

Override the OnPaint(PaintEventArgs e)

The Control class Events

The Control class also defines a number of events that can logically be grouped into two major categories :

Mouse events (Click, DoubleClick, MouseEnter, MouseLeave, MouseDown. MouseUp).

Keyboard events (KeyPress, KeyUp, KeyDown).

Handling Keyboard Events:

Keyboards events are generated when keys on the keyboard are pressed or released.

There are two types of Keyboard events:

KeyPress events (or override OnKeyPress) which fires when a key representing an ASCII character is pressed “The KeyPress event is not raised by noncharacter keys “

KeyUp and KeyDown events (or override OnKeyUp and OnKeyDown).

Handling Keyboard events

KeyUp and KeyDown events (or override.OnKeyUp and OnKeyDown)

The delegate:

public delegate void **KeyEventHandler**(object sender,KeyEventArgs e)

Whichever the way to handle the KeyUp and KeyDown events, you have an object of KeyEventArgs when a key is pressed or released. This object have the following properties:

Keycode (Keys type): property have which key is being pressed(Char, or ShiftKey, ControlKey, Menu"Alt").

The Modifiers (Keys type): property indicates whether the Alt, Ctrl, or Shift keys are also pressed (Alt, Shift, Control, None)

The KeyData (Keys type): combines the KeyCode and Modifiers.

These properties are from the Keys data type, which is an enumeration.

Handling Keyboard events

KeyUp and KeyDown events (or override.OnKeyUp and OnKeyDown)

Keys type:

Keys Enumeration (letters)

Member	Value	Member	Value
A	65	N	78
B	66	O	79
C	67	P	80
D	68	Q	81
E	69	R	82
F	70	S	83
G	71	T	84
H	72	U	85
I	73	V	86
J	74	W	87
K	75	X	88
L	76	Y	89
M	77	Z	90

Handling Keyboard events

KeyUp and KeyDown events (or override.OnKeyUp and OnKeyDown)

Keys type:

Keys Enumeration (number keys)

Member	Value
D0	48
D1	49
D2	50
D3	51
D4	52
D5	53
D6	54
D7	55
D8	56
D9	57

Handling Keyboard events

KeyUp and KeyDown events (or override.OnKeyUp and OnKeyDown)

Keys type:

Keys Enumeration (function keys)

Member	Value	Member	Value
F1	112	*F13*	124
F2	113	*F14*	125
F3	114	*F15*	126
F4	115	*F16*	127
F5	116	*F17*	128
F6	117	*F18*	129
F7	118	*F19*	130
F8	119	*F20*	131
F9	120	*F21*	132
F10	121	*F22*	133
F11	122	*F23*	134
F12	123	*F24*	135

Handling Keyboard events

KeyUp and KeyDown events (or override.OnKeyUp and OnKeyDown)

Keys type:

Keys Enumeration (keypad operators)

Member	Value	Description
Multiply	106	Numeric keypad *
Add	107	Numeric keypad +
Subtract	109	Numeric keypad –
Divide	111	Numeric keypad /

Keys Enumeration (keypad cursor movement)

Member	Value	Member	Value	Member	Value
Home	36	*Up*	38	*PageUp or Prior*	33
Left	37	*Clear*	12	*Right*	39
End	35	*Down*	40	*PageDown or Next*	34
Insert	45			*Delete*	46

Handling Keyboard events

KeyUp and KeyDown events (or override.OnKeyUp and OnKeyDown)

Keys type:

Keys Enumeration (ASCII control keys)

Member	Value
Back	8
Tab	9
LineFeed	10
Enter Return	13
Escape	27
Space	32

Member	Value	Member	Value	Member	Value
ShiftKey	16	*LShiftKey*	160	*RShiftKey*	161
ControlKey	17	*LControlKey*	162	*RControlKey*	163
Menu	18	*LMenu*	164	*RMenu*	165

Handling Keyboard events

KeyUp and KeyDown events (or override.OnKeyUp and OnKeyDown)

Keys type:

It also includes these modifiers code:

Keys Enumeration (modifier keys)

Member	Value
None	0x00000000
Shift	0x00010000
Control	0x00020000
Alt	0x00040000

These modifiers codes indicates if the Shift Ctrl or Alt keys were pressed (used with the Modifiers property)

Handling Keyboard events

Note:

It is not necessary to handle the KeyDown or KeyUp event to determine whether the Shift, Ctrl or Alt key is pressed. Suppose you are handling mouse event. You can obtain the current state of the three modifier keys using the static Control.ModifierKeys property

Suppose we want to do something when Shift or Ctrl key or Both were pressed

```
Keys keysMod = Control.ModifierKeys;

if(keysMod == (Keys.Shift | Keys.Control)) //Both pressed
{…}
else if(keysMod == Keys.Shift)
{…}
```

Handling Keyboard events

KeyPress (or override.OnKeyPress)

which fires when a key representing an ASCII character is pressed "The KeyPress event is not raised by noncharacter keys"

Using the KeyPress event, we cannot determine if modifier keys (such as Shift, Alt and Control) were pressed.

The delegate:

public delegate void **KeyPressEventHandler**(object sender, KeyPressEventArgs e)

The KeyPressEventArgs has a KeyChar property which hold the value.

With the Ctrl key down, you can generate control characters that are reported through the KeyPress event.

You get character codes 0x0001 through 0x001A by using the Ctrl key in combination with A through Z regardless the Shift Key status.

Handling Keyboard events

KeyPress (or override.OnKeyPress)

```
protected override void OnKeyPress(KeyPressEventArgs e)
{       switch((int)e.KeyChar)
        {       case 2:    //Blue
                        m_RectBackColor = Color.Blue;
                break;
                case 7:    //Green
                        m_RectBackColor = Color.Green;
                break;
                case 18:   //Red
                        m_RectBackColor = Color.Red;
                break;
        }
        Invalidate();
}
```

Handling Keyboard events

Notice that KeyPress occurs in between KeyDown and KeyUp, not after KeyUp

Example: if you press Shift+A, you'll get:

1. KeyDown: KeyCode=Keys.ShiftKey, KeyData=Keys.ShiftKey, Shift, Modifiers=Keys.Shift
2. KeyDown: KeyCode=Keys.A, KeyData=Keys.A | Keys.Shift, Modifiers=Keys.Shift
3. KeyPress: KeyChar='A'
4. KeyUp: KeyCode=Keys.A
5. KeyUp: KeyCode=Keys.ShiftKey

Handling Mouse events

- There are a number of events related to the user's employment of the mouse.
- These events include MouseDown, MouseUp, MouseMove, MouseEnter, MouseLeave, MouseHover.
- The MouseDown, MouseUp, and MouseMove events
- The delegate:

public delegate void **MouseEventHandler**(object sender, MouseEventArgs e)

MouseEventArgs has many properties that contains all information about the event.

Handling Mouse events

MouseEventArgs properties:

MouseEventArgs Properties			
Type	**Property**	**Accessibility**	**Description**
int	*X*	get	The horizontal position of the mouse
int	*Y*	get	The vertical position of the mouse
MouseButtons	*Button*	get	The mouse button or buttons
int	*Clicks*	get	Returns 2 for a double-click

The Button property indicates the mouse button or buttons involved in the event. The Button property is a MouseButtons enumeration value

MouseButtons Enumeration	
Member	**Value**
None	0x00000000
Left	0x00100000
Right	0x00200000
Middle	0x00400000

Handling Mouse events

MouseEventArgs properties:

Notice that the values are bit flags that can be combined. For example, if both the left and right buttons are pressed, the Button property equals 0x00300000.

```
public void OnMouseUp(object sender, MouseEventArgs e) {
        if(e.Button == MouseButtons.Left)
                MessageBox.Show("Left clicking!");
        else if(e.Button == MouseButtons.Right)
                MessageBox.Show("Right clicking!");
        else
                MessageBox.Show("Middle clicking!");
}
```