MOUSE EVENTS

- Related to graphics methods are mouse events. The mouse is a primary interface for doing graphics in Visual C# Express. We've already used the mouse to Click on controls. Here, we see how to recognize other mouse events in controls. Many controls recognize mouse events - we are learning about them to allow drawing in panel controls.

MOUSE DOWN EVENT

- The MouseDown event method is triggered whenever a mouse button is pressed while the mouse cursor is over a control. The form of this method is:

```
private void controlName_MouseDown(object sender,
MouseEventArgs e)
{
    C# code for MouseDown event]
}
```

- This is the first time we will use the arguments (information in parentheses) in an event method. This is information C# is supplying, for our use, when this event method is executed.

- Note this method has two arguments: **sender** and **e**. **sender** is the control that was clicked to cause this event (MouseDown) to occur. In our case, it will be the panel control.

- The argument **e** is an event handler revealing which button was clicked and the coordinate of the mouse cursor when a button was pressed. We are interested in three properties of the event handler **e**:

Value	Description
e.Button	Mouse button pressed. Possible values are: **MouseButtons.Left, MouseButtons.Center, MouseButtons.Right**
e.X	X coordinate of mouse cursor in control when mouse was clicked
e.Y	Y coordinate of mouse cursor in control when mouse was clicked

- Only one button press can be detected by the **MouseDown** event - you can't tell if someone pressed the left and right mouse buttons simultaneously. In drawing applications, the **MouseDown** event is used to initialize a drawing process. The point clicked is used to start drawing a line and the button clicked is often used to select line color.

EXAMPLE

```
private void panel1_MouseDown(object sender, MouseEventArgs e)
{
            switch (e.Button)
            {
                case MouseButtons.Left:
                    label1.Text = "Left";
                    break;

                case MouseButtons.Middle:
                    label1.Text = "Middle";
                    break;

                case MouseButtons.Right:
                    label1.Text = "Right";
                    break;
            }
 label2.Text = Convert.ToString(e.X) + "," + Convert.ToString(e.Y);

}
```

MOUSE UP EVENT

- The **MouseUp** event is the opposite of the MouseDown event. It is triggered whenever a previously pressed mouse button is released. The method format is:

MOUSE UP EVENT

- The **MouseUp** event is the opposite of the MouseDown event. It is triggered whenever a previously pressed mouse button is released. The method format is:

```
private void controlName_MouseUp(object sender, MouseEventArgs e)
{
    [C# code for MouseUp event]
}
```

- Notice the arguments for MouseUp are identical to those for MouseDown. The only difference here is e.Button tells us which mouse button was released. In a drawing program, the MouseUp event signifies the halting of the current drawing process.

MOUSE MOVE EVENT

- The **MouseMove** event is continuously triggered whenever the mouse is being moved. The event method format is:

```
private void controlName_MouseMove(object sender, MouseEventArgs e)
{
    [C# code for MouseMove event]
}
```

- And, yes, the arguments are the same. **e.Button** tells us which button is being pressed (if any) as the mouse is moving over the control and (e.X, e.Y) tell us the mouse position. In drawing processes, the **MouseMove** event is used to detect the continuation of a previously started line. If drawing is continuing, the current point is connected to the previous point using the current pen.