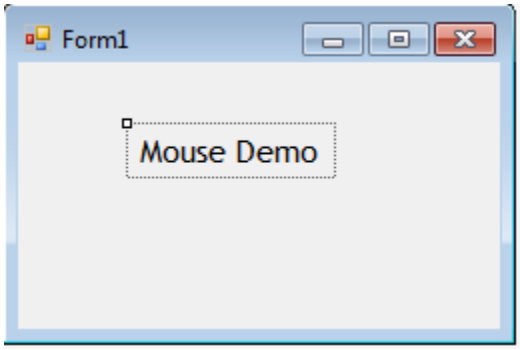
Mouse Events

Windows applications are in general mouse driven applications that is the application can be better control through the various mouse events. The following are the various mouse events.

MouseEnter	This event occurs when the mouse enters a control
MouseHover	This event occurs when the mouse hovers or keeps stationary for a few seconds.
MouseLeave	This event occurs when the mouse leaves a control.

Note: All the above events take the arguments **System.Eventargs.** It will not identify the mouse buttons.

MouseDown	This event occurs when any one of the mouse button is pressed.
MouseUp	This event occurs when any one of the mouse buttons pressed and released.
MouseMove	This event occurs when the mouse is move without pressing any button.
MouseClick	This event occurs when we click any button on the mouse.
MouseDoubleClick	This event occurs when we double click the mouse.




```
//Mouse Down
        if (e.Button == MouseButtons.Left)
           MessageBox.Show("Left Button is pressed");
        else if (e.Button == MouseButtons.Right)
           MessageBox.Show("Right Button is pressed");
        else
           MessageBox.Show("Middle Button is pressed");
//MouseUp
        if (e.Button == MouseButtons.Left)
           MessageBox.Show("Left Button is pressed");
        else if (e.Button == MouseButtons.Right)
           MessageBox.Show("Right Button is pressed");
        else
           MessageBox.Show("Middle Button is pressed");
//Click
        MessageBox.Show("Mouse Clicked", "Click", MessageBoxButtons.OK, MessageBoxIcon.Information);
//Double Click
        MessageBox.Show("Mouse Double Clicked", "Double Click", MessageBoxButtons.OK, MessageBoxIcon.Information);
//Hover
        MessageBox.Show("Mouse Hovered", "Hover", MessageBoxButtons.OK, MessageBoxIcon.Information);
//Enter
        MessageBox.Show("Mouse Entered", "Enter", MessageBoxButtons.OK, MessageBoxIcon.Information);
//Leave
        MessageBox.Show("Mouse Leaved", "Leave", MessageBoxButtons.OK, MessageBoxIcon.Information);
```

Create an interface to implement graphic methods through mouse events

Step 1: Create a form using the following properties.

Bgcolor: Black
WindowState: Maximize
Text: Implementation of graphics through mouse events.

Step 2: Add the following events for form.

```
//MouseDown
            Random rnd = new Random();
            int a, r, g, b;
            a = rnd.Next(0, 255);
            r = rnd.Next(0, 255);
            g = rnd.Next(0, 255);
            b = rnd.Next(0, 255);
            Graphics gr;
            gr = this.CreateGraphics();
            SolidBrush sb=new SolidBrush(Color.FromArgb(a,r,g,b));
            gr.FillRectangle(sb,e.X,e.Y,200,200);
            if(e.Button==MouseButtons.Right)
                this.Invalidate();
//MouseUp
            Random rnd = new Random();
            int a, r, g, b;
            a = rnd.Next(0, 255);
            r = rnd.Next(0, 255);
            g = rnd.Next(0, 255);
            b = rnd.Next(0, 255);
            Graphics gr;
            gr = this.CreateGraphics();
            SolidBrush sb = new SolidBrush(Color.FromArgb(a, r, g, b));
            gr.FillRectangle(sb, e.X, e.Y, 200, 200);
            if (e.Button == MouseButtons.Right)
                this.Invalidate();
```

Create an interface to draw multiple rectangles over the form using mouse events.

Step 1: Create a form using the following properties.

Bgcolor: Black
WindowState: Maximize
Text: Implementation of graphics through mouse events.

Step 2: Add the following events for form.

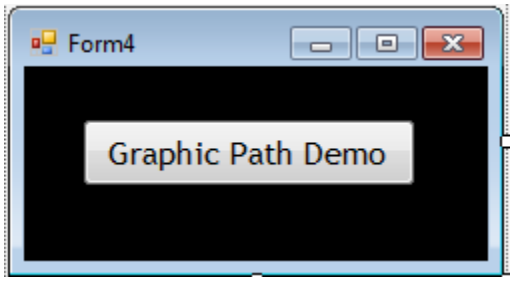
```
//MouseDown
            Random rnd = new Random();
            int a, r, g, b;
            a = rnd.Next(0, 255);
            r = rnd.Next(0, 255);
            g = rnd.Next(0, 255);
            b = rnd.Next(0, 255);
            int x1, y1, x2, y2, x3, y3, x4, y4;
            x1 = rnd.Next(0, 1024);
            y1 = rnd.Next(0, 768);
            x2 = rnd.Next(0, 1024);
            y2 = rnd.Next(0, 768);
            x3= rnd.Next(0, 1024);
            y3 = rnd.Next(0, 768);
            x4 = rnd.Next(0, 1024);
            y4 = rnd.Next(0, 768);
            Graphics gr;
            gr = this.CreateGraphics();
            Pen p = new Pen(Color.FromArgb(a, r, g, b));
            System.Drawing.Rectangle[] rects = new System.Drawing.Rectangle[4];
            rects[0] = new System.Drawing.Rectangle(x1, y1, 100, 200);
            rects[1] = new System.Drawing.Rectangle(x2, y2, 100, 200);
            rects[2] = new System.Drawing.Rectangle(x3, y3, 100, 200);
            rects[3] = new System.Drawing.Rectangle(x4, y4, 100, 200);

            gr.DrawRectangles(p, rects);
            if (e.Button == MouseButtons.Right)
                this.Invalidate();
```

Graphics Path

This class is used to point to a graphic object with reference to its complete region. The region includes the complete height and width of the complete height and width of the complete graphic object. Graphics path is a class available under **System.Drawing.Drawing2D** namespace. It can be used to dynamically change the space of the form, any control that supports graphics.

Create an interface to change the shape of the form when the respected shape is selected.
Step 1: create a form



```
//Graphic Path Demo
            // Create a Graphics object
            Graphics g = this.CreateGraphics();
            g.Clear(this.BackColor);

            // Create a GrphicsPath object
            GraphicsPath path = new GraphicsPath();

            // Create an array of points
            Point[] pts =
            {
                        new Point (40, 80),
                        new Point (50, 70),
                        new Point (70, 90),
                        new Point (100, 120),
                        new Point (80, 120)
            };

            // Start first figure and add an arc and a line
            path.StartFigure();
            path.AddArc(250, 80, 100, 50, 30, -180);
            path.AddLine(180, 220, 320, 80);

            // Close first figure
            path.CloseFigure();

            // Start second figure and two lines and
            // a curve and close all figures
            path.StartFigure();
            path.AddLine(50, 20, 5, 90);
            path.AddLine(50, 150, 150, 180);
            path.AddCurve(pts, 5);
            path.CloseAllFigures();

            // Create third figure and don't close it
            path.StartFigure();
            path.AddLine(200, 230, 250, 200);
            path.AddLine(200, 230, 250, 270);

            // Draw path
            g.DrawPath(new Pen(Color.FromArgb(255, 255, 0, 0), 2), path);

            // path.Reverse();
            //path.Reset();
            // Dispose of object
            g.Dispose();
```