AMBIENTES VISUAIS

Manipular eventos de mouse

Objetivo: Demonstrar a manipulação de eventos de mouse, como cliques, pressionamentos e movimentações

Os eventos de mouse são gerados quando o mouse interage com um controle tal como um Form, Panel, Label, entre outros. Cada método de manipulação de eventos de mouse recebe um objeto **MouseEventArgs** como argumento.

A classe **MouseEventArgs** contém informações sobre o evento de mouse, como as coordenadas x e y do ponteiro do mouse, o botão de mouse pressionado, o número de cliques e o número de passos percorridos pelo mouse.

As tabelas abaixo apresenta os principais eventos e propriedades da classe **MouseEventArgs**

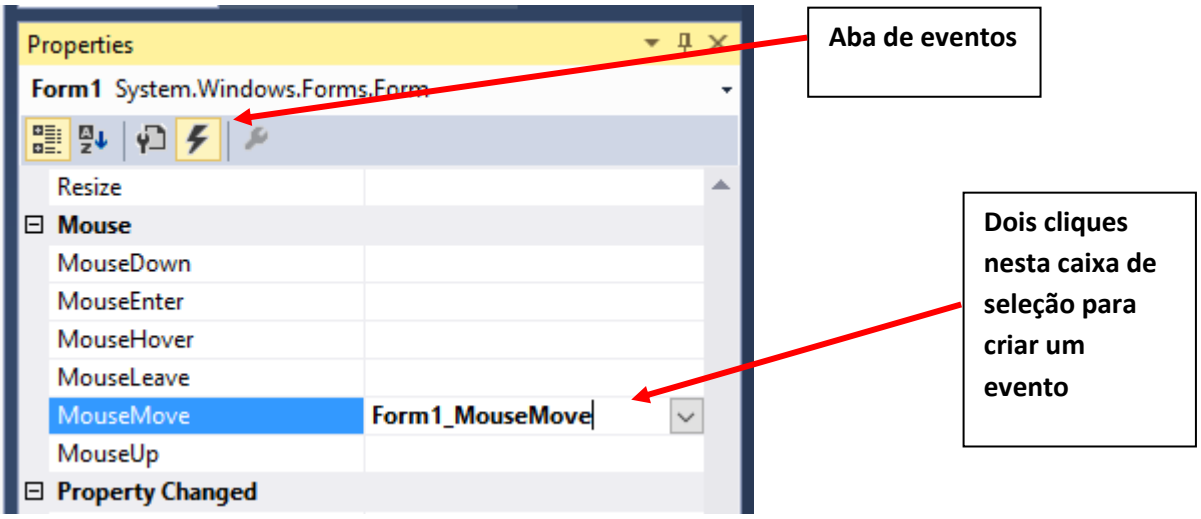
Eventos de mouse, delegados e argumentos de evento	
Eventos de mouse (Delegado ***EventHandler****, argumentos de evento* ***EventArgs****)*	
`MouseEnter`	Disparado se o ponteiro do mouse entra na área do controle.
`MouseLeave`	Disparado se o ponteiro do mouse deixa a área do controle.

Eventos de mouse, delegados e argumentos de evento	
Eventos de mouse (Delegado ***MouseEventHandler****, argumentos de evento* ***MouseEventArgs****)*	
`MouseDown`	Disparado se o botão do mouse é pressionado enquanto seu ponteiro está sobre a área do controle.
`MouseHover`	Disparado se o ponteiro do mouse paira sobre a área do controle.
`MouseMove`	Disparado se o ponteiro do mouse é movido enquanto está na área do controle.
`MouseUp`	Disparado se o botão do mouse é liberado quando o ponteiro está sobre a área do controle.
Propriedades da classe ***MouseEventArgs***	
`Button`	O botão do mouse que foi pressionado (`left`, `right`, `middle` ou `none`).
`Clicks`	O número de vezes que o botão do mouse foi clicado.
`X`	A coordenada *x* do evento, relativa ao controle.
`Y`	A coordenada *y* do evento, relativa ao controle.

Fonte: Deitel, C# Como Programar.

Criaremos uma pequena aplicação que utilize os eventos do mouse para desenhar uma linha sobre um formulário, assim que o usuário pressionar o botão.

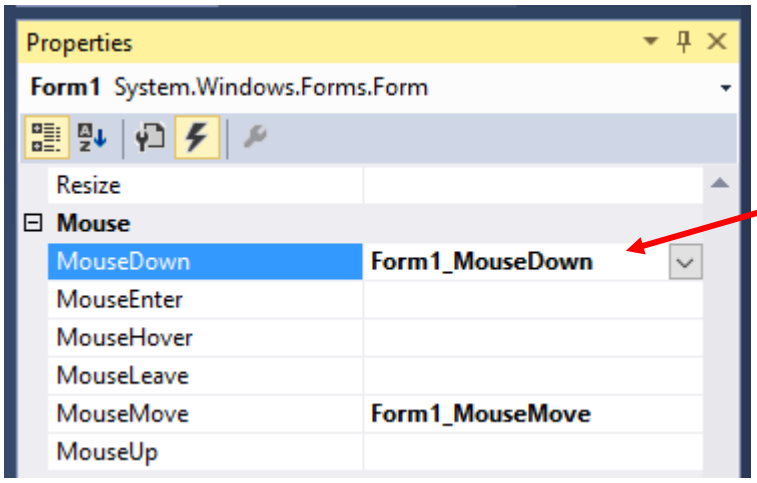
1. Crie um novo projeto no VisualStudio, chamado EventoMouse
2. Altere a propriedade Text do Form1 para se chamar “Quadro”
3. Selecione o Form1 e crie um evento MouseMove para este form.



4. Declare uma variável do tipo bool como atributo do form, por meio dela vamos controla quando pintar ou não o formulário. Implemente também o evento gerado Form1_ MouseMove como o exemplo abaixo.

```
13      public partial class Form1 : Form
14      {
15          /*
16              atributo do formulário
17              utilizado como flag para controla
18              quando deve pintar
19          */
20          bool devePintar = false;
21
22
23          public Form1()
24          {
25              InitializeComponent();
26          }
27
28          private void Form1_MouseMove(object sender, MouseEventArgs e)
29          {
30
31              if(devePintar)
32              {
33                  //cria um do graphics do formulario
34                  Graphics graphics = CreateGraphics();
35                  graphics.FillEllipse(new SolidBrush(Color.BlueViolet),
36                                       e.X, e.Y, 4,4);
37
38              }
39
40          }//fim do evento
```

5. Volte a visualização do formulário (modo Design) selecione o Form1 e crie um novo evento MouseDown. Este evento trocará o valor da flag devePintar de false para true.

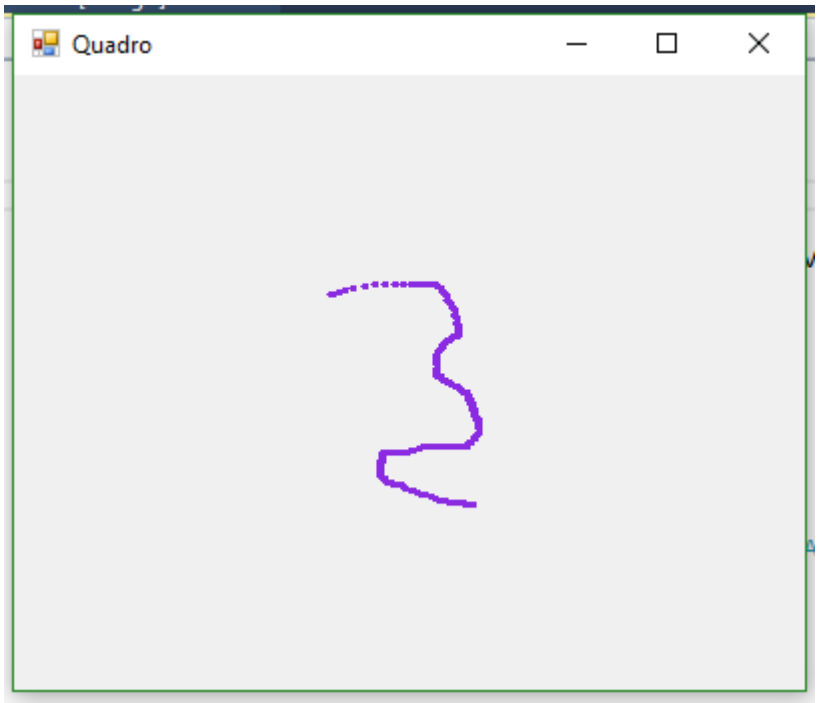


Dois cliques nesta caixa de seleção para criar um evento

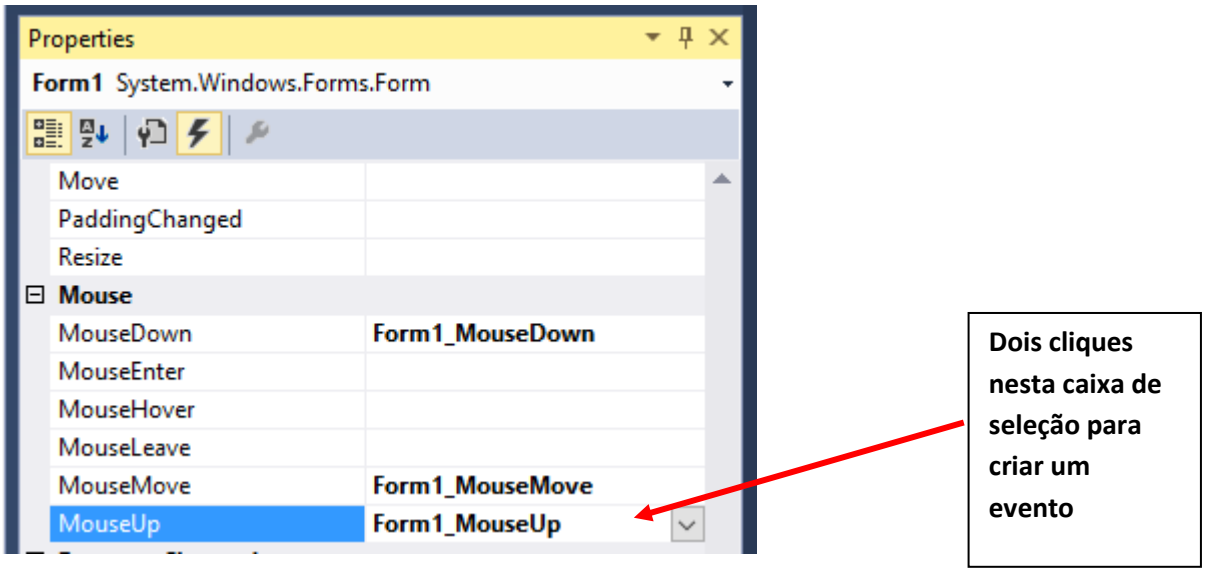
6. Implemente este evento como o exemplo a seguir:

```
44          private void Form1_MouseDown(object sender, MouseEventArgs e)
45          {
46              devePintar = true;
47          }
```

7. Execute a aplicação e pressione o mouse sobre o formulário. Já é possível pintar após o botão do mouse ter sido pressionado , mas após soltar o botão, o ideal é que não pinte mais.



8. Volte a selecionar o Form1 e adicione um novo evento MouseUp.



9. Implemente o evento Form1_MouseUp como o exemplo abaixo:

```
51          private void Form1_MouseUp(object sender, MouseEventArgs e)
52          {
53              devePintar = false;
54          }
55
```

10. Execute e teste aplicação com os eventos de mouse.

Considerações:

Quando o mouse se move, enquanto o botão está pressionado, o vento **MouseMove** é gerado. Dentro do manipulador de evento **Form1_MouseMove**, o programa desenha apenas se a flag **devePintar** for **true** (indicando que o botão de mouse está pressionado).

O objeto **Graphics** do formulário fornece métodos para desenhar várias figuras. O método **FillEllipse** desenha um pequeno círculo em cada ponto sobre o qual o ponteiro do mouse se move. O primeiro paramento do método **FillEllipse** é um objeto **SolidBrush**, o qual determina a cor desenhada. Criamos um novo objeto **SolidBrush**, passando para o construtor um valor de **Color**. A classe **Color** contém muitas constantes de cor predefinidas, selecionamos **BrueViolet**. As coordenas **x** e **y** representam a localização do evento do mouse e são obtidas a partir dos argumentos do evento de mouse (**e.X** e **x.Y**).