

Aula 1 - Introdução ao WPF

Resumo:

- Programação por eventos
- Ferramentas de desenvolvimento
- Conceitos básicos
- Introdução ao XAML

Programação sequencial:

A programação sequencial refere-se aos programas compostos por linhas de código que são executadas uma a seguir a outra do princípio até ao fim. Quando o programa é executado esse corre sem nenhum atraso de uma forma sequencial.

```
Main()  
{  
    Linha de código  
    Linha de código  
    Linha de código  
}
```

Pseudo código: programação sequencial

Programação por eventos:

Um paradigma de programação diferente é usado para a programação de interfaces, conhecido como programação por eventos (*event-driven programming*). Neste tipo de programas, existem uma série de controlos ou objectos (por exemplo, botões, caixas de texto, listas, etc.) e o programa está a espera da ocorrência de um evento nesses controlos, tal como um clique num botão ou premir uma tecla. Depois da ocorrência de um evento, é executada o procedimento associado ao evento que ocorreu. O WPF, como quase todas as linguagens desenvolvidas para criar interfaces gráficas é baseado nesse paradigma de programação.

Neste tipo de programação o programa tem que responder aos vários eventos através do chamando *event handling*. O sistema tem que associar aos vários eventos de interesse (*event source*) um observador (*event listener* ou *observer*) que indica qual o método que será executado em resposta ao evento gerado (*callback*).

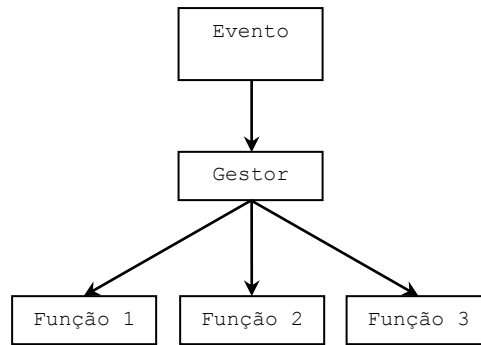


Figura 1-1: Event Handling

```

Main()
{
    Código de inicialização
    Registo das callbacks
    Ciclo principal
}

Callback_1()
{
    Linha de código
    Linha de código
    Linha de código
}

Callback_2()
{
    ...
}
    
```

Figura 1-2: Pseudo-código - programação por Evento

1.1. Ferramentas de desenvolvimento / WPF

O *Windows Presentation Foundation* (WPF) é a *framework* da Microsoft para o desenvolvimento de aplicações interativas combinando interfaces, gráficos 2D e 3D, documentos e multimédia numa só ferramenta. As WPFs separam as camadas de interface e de lógica dos programas. Para tal recorrem a linguagem XAML (*Extensible Application Markup Language*) uma linguagem baseada em XML que permite criar e inicializar componentes. NET.

Para desenvolver aplicações em WPF a ferramenta principal é o Visual Studio que está mais vocacionada para programadores permitindo criar e editar código em XAML.


1.2. Primeiro exemplo

No visual studio, crie uma nova aplicação WPF (File->New->Project). Escolhe Visual C# - WPF application como tipo de aplicação e escolhe um diretório e um nome para o projeto.

Explore o código criado pela aplicação, note a existência de dois ficheiros de código com extensão .cs e dois ficheiros .xaml que descrevem a interface.

Os eventos já estão pré-definidos em Visual C#, existindo uma grande quantidade de eventos aos quais se pode associar código.

O primeiro evento que vamos explorar é o clique na janela principal.

Na visualização da interface (clcando no ficheiro `xaml` e escolhendo `View Designer`), visualize as propriedades da `Windows` (cuidado de seleccionar a janela-`window` e não a `grelha-grid`) e selecione o icon dos eventos  que permite visualizar os eventos associados ao controlo seleccionado. Na lista de eventos, faça duplo clique na opção `MouseLeftButtonDown` e observe o que acontece ao nível do código.

Acrescente na função criada automaticamente o código seguinte que permite escrever uma linha na janela de saída do Visual C#.

```
Console.WriteLine("Olá Mundo");
```

Execute o código e observe o que ocorre na janela de `output` sempre que clica na janela. Verifique que está a visualizar a janela de `output` do Visual Studio enquanto corre o programa, dado que o comando de escrita na consola escreve nessa janela. Para ativar a janela use a opção `View->Output`.

1.3. Outros eventos em Visual C#

Modifique o código para responder a outros tipos de eventos colocando na janela de saída uma mensagem específica para cada evento. Explore e escolha quatro eventos (associando uma mensagem diferente para cada um deles) entre os seguintes (`Activated`, `MouseMove`, `SizeChanged`, `MouseDoubleClick`, `MouseDown`, `Closed`, `Loaded`, `DragEnter`, `Drop`, etc.). A que eventos correspondem deles? Note que para alguns eventos pode ser necessário modificar atributos da janela (por exemplo `AllowDrop`)

1.4. Associar vários controlos ao mesmo evento

É possível associar vários controlos ao mesmo evento. Se o desejar, pode remover o código acrescentado na última alínea mas note que ao remover o código associado a um evento é necessário remover no ficheiro `xaml` correspondente a linha em que é definido o evento.

Acrescentar na `grid` um botão usando a `toolbox`. Modifique o texto para `button1` alterando no `xaml` ou na interface o atributo `content`. Associe ao evento clique no botão o código seguinte que mostra uma mensagem numa `dialog box`.

```
MessageBox.Show("Olá Mundo")
```

Para definir o evento pode simplesmente fazer duplo clique no botão (o evento clique é definido por omissão). Ao associar o evento clique ao botão 1, o Visual Studio cria automaticamente o código seguinte:

```
private void button1_Click(object sender, RoutedEventArgs e)
{
}
```

O Visual C# criou uma sub-rotina com o nome `Button1_Click` invocada sempre que houver um clique no botão 1.

Coloque um novo botão (`button2`) que deverá estar associado a mesma sub-rotina. Para tal, basta associar ao evento clique do segundo botão a função já criada usando as propriedades do mesmo.

Pode acrescentar um `breakpoint` (`F9` ou `Debug->Toggle Breakpoint`) na função para verificar que a mesma é invocada ao clicar em qualquer um dos botões.

Os dois eventos “clique no botão 1” e “clique no botão2” estão agora associados a mesma sub-rotina. Note contudo que ao clicar no botão 2, o mesmo salta para o código associado ao botão 1 deixando de ser possível ter uma subrotina específica para o botão 2 dado que o mesmo já está associado a outra subrotina.

1.5. Argumentos dos eventos

Os argumentos da função `Button1_click` que vimos anteriormente são: `object sender`, `RoutedEventArgs e`. O `sender` é uma referência ao controlo/objeto que desencadeou o evento (o botão que foi acionado por exemplo). A variável `e` por seu lado contém informação relativa ao evento, por exemplo o botão ou a posição do rato no monitor no caso de clique num botão.

Para ilustrar a utilização dos argumentos, vamos utilizar outro evento: `MouseMove`.

Comece por remover todo o código da alínea anterior ficando só com um botão. Associe ao botão o evento `MouseMove`. Utilize o segundo argumento da função (do tipo `MouseEventArg`) para determinar qual o botão que foi utilizado. Para tal, pode utilizar o código seguinte:

```
if (e.RightButton == MouseButtonState.Pressed)
    Console.WriteLine("Botão Direito");
```

Execute o código e veja o que acontece quando clica nos botões do rato.

Acrescente uma função para detetar um duplo-clique do rato na janela e indicar a posição onde ocorreu o clique. Para tal, pode utilizar o código que segue:

```
Point p = e.GetPosition(this);
double xPos = p.X;
double yPos = p.Y;
Console.WriteLine("The X Position is " + xPos + " The Y Position is " + yPos);
```

Teste novamente o código. O que é que ocorre? Usando este programa, tente determinar quais são as coordenadas que são utilizadas ao nível da janela tentando determinar os limites máximos e mínimos em pixels e verifique onde é definida no xaml essa dimensão.

1.6. Animação

Modifique o código não para mostrar uma mensagem mas para mudar o conteúdo (`content`) do botão ao clicar nele (por exemplo mudo o título de `button1` para `Olá Mundo`).

Inclua agora no ficheiro a biblioteca para animação:

```
using System.Windows.Media.Animation
```

Adicione na função de clique o código seguinte:

```
SolidColorBrush init_color = new SolidColorBrush(Colors.DarkGray);
this.Background = init_color;
ColorAnimation colorAnim = new ColorAnimation();
colorAnim.Duration = new
Duration(TimeSpan.FromMilliseconds(2000));
colorAnim.To = Colors.LightGray;
init_color.BeginAnimation(SolidColorBrush.ColorProperty,
colorAnim);
```

Modifique o programa para que o fundo da janela corresponde a cor do início da animação. Modifique ainda alguns parâmetros da animação (cor inicial, final, tempo, etc...).

1.7. Utilização de XAML

Utilize o código seguinte para adicionar um novo botão diretamente no ficheiro XAML em vez de utilizar o Designer do Visual Studio.

```
<Button Content="XAML button" Margin="24,195,22,20" Name="button2"
FontSize="25"/>
```

Execute o código e veja como o mesmo se comporta se modificar o tamanho da janela. No visual *design* do *Visual Studio* utilize os *rulers* para alinhar o botão na parte inferior da janela. Confirme o que ocorre ao nível do XAML em termos de alinhamento.

1.8. Exemplo simples de data binding

Uma das grandes vantagens do XAML é a possibilidade de associar informação aos controlos (data binding). Acrescente na interface um *slider* e uma *textbox*. No xaml associado ao *slider* e a *textbox*, associe o código seguinte, mantendo inalterado o resto do código.

```
<Slider x:Name ="SliderName"/>
<TextBox Text="{Binding Value, ElementName= SliderName}"/>
```

Este código associa um nome único ao *slider* (diretiva *x:name*) para depois poder referir o mesmo no código. O conteúdo da *textbox* (*text*) é então associado a esse controlo. Modifique o código para que o valor devolvido seja entre os valores 0 e 100. Investigue a propriedade *StringFormat*¹ que permite formatar o aspeto dos dados. Veja a formatação de dados numéricos para *string*² e formate o exemplo para mostrar só 2 casa decimais.

1.9. Um programa para monitorizar o rato na janela

Utilizando todo o código e os exemplos que foram introduzidos nesta aula, Crie um pequeno programa que permita monitorizar o rato numa janela. O programa deve criar uma janela vazia e realizar as seguintes operações:

- Contar o nº de cliques do rato, distinguindo botão direito e botão esquerdo.
- Indicar na consola a posição x,y onde ocorre cada clique do rato.
- Somar o movimento todo que o rato realiza dentro da janela.
- Ao fechar, visualizar os resultados da monitorização usando o código:

```
MessageBox.Show("Texto a visualizar").
```

Uma vez concluído este programa, modifique o mesmo adicionando os controlos necessários para visualizar os valores na janela a medida que se interage com a mesma.

¹<https://msdn.microsoft.com/en-us/library/system.windows.data.bindingbase.stringformat%28v=vs.110%29.aspx>

²[https://msdn.microsoft.com/en-us/library/0c899ak8\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/0c899ak8(v=vs.110).aspx)