Windows Presentation Foundation

IFRN - RN

WPF - Animação

Um dos recursos introduzidos no WPF que mais chamam a atenção são as animações. Antes do WPF, quando queríamos fazer alguma animação em nosso programa, devíamos criar um timer e ir alterando as propriedades dos objetos à medida que o tempo passasse. Mesmo que isto fosse relativamente simples, tínhamos de guardar cada estado do objeto, para que fosse mostrado corretamente.

WPF - Animação

Com o WPF, isto não é mais necessário: podemos especificar o valor inicial e final de uma propriedade, determinar a duração da animação e o WPF se encarrega de animar o objeto, sem que seja necessário escrever código para isso.

Quando queremos fazer um objeto animado devemos alterar alguma propriedade sua em um determinado intervalo de tempo. Por exemplo, se queremos que um retângulo vá de um ponto a outro, devemos alterar sua posição com o tempo. Isto não é feito de uma vez, mas discretamente, a intervalos regulares. Em cada um destes intervalos, devemos especificar a nova posição do retângulo e desenhá-lo lá. Cada um destes desenhos é chamado de “frame”.

WPF - Animação

Quando queremos mostrar a animação, mostramos os “frames” em sequência e eles dão a impressão que o retângulo está se movendo. O intervalo de apresentação entre os frames determina a qualidade da animação: se mostrarmos mais frames por segundo, o espaço entre dois movimentos é menor e a animação é mais uniforme. Se mostrarmos menos frames por segundo, o movimento parece ser feito aos pulos. Por outro lado, um maior número de frames representa maior quantidade de dados a ser representada, e isto requer maior espaço para armazenamento e hardware mais potente para mostrar mais informações. Uma velocidade razoável para a exibição de animações é de 30 frames por segundo, taxa usada nos filmes, embora possamos usar até 15 frames por segundo, com qualidade aceitável. O WPF regula suas animações, por padrão, em 24 frames por segundo.

WPF - Animação

O WPF facilita bastante a criação de animações. Não precisamos mais criar código para gerar os frames, basta apenas dizer o valor inicial e final da propriedade que se quer alterar e a duração da animação. Ele se encarrega de gerar toda a animação. O seguinte código XAML movimenta um retângulo com bordas arredondadas na horizontal indefinidamente:

```
<Canvas>
    <Rectangle Canvas.Top="100" Width="200" Height="100" RadiusX="10" RadiusY="10" Fill="Red">
      <Rectangle.Triggers>
        <EventTrigger RoutedEvent="Rectangle.Loaded">
          <BeginStoryboard>
            <Storyboard TargetProperty="(Canvas.Left)">
              <DoubleAnimation From="0" To="800" Duration="0:0:5"   RepeatBehavior="Forever" AutoReverse="True"/>
            </Storyboard>
          </BeginStoryboard>
        </EventTrigger>
      </Rectangle.Triggers>
    </Rectangle>
  </Canvas>
```

WPF - Animação

As animações em WPF normalmente são associadas a *Triggers*. Quando um gatilho é disparado, ele pode ativar uma animação chamando a ação**BeginStoryboard**. No exemplo anterior, a animação é disparada quando o carregamento do retângulo está completo (quando **Rectangle.Loaded** for disparado). Na ação que é executada quando o gatilho é disparado, colocamos um elemento do tipo **BeginStoryboard**.

Este elemento tem uma propriedade chamada **Storyboard**, que contém nossa animação. Indicamos aí qual a propriedade que será afetada na animação (**Canvas.Left –** a posição horizontal do elemento em relação ao Canvas) e definimos a animação, usando a classe **DoubleAnimation**. Esta classe anima um valor de tipo **Double**. Poderíamos ter outros tipos de animação, como **PointAnimation** ou **ColorAnimation** (por exemplo, se quisermos fazer que a cor de preenchimento do retângulo mudasse de vermelho para azul, usaríamos o **ColorAnimation**).

WPF - Animação

Os valores iniciais e finais são definidos com as propriedades **From e To**. Assim, a posição do retângulo irá variar de 0 a 800. A duração da animação é determinada pela propriedade **Duration**. Ela é indicada no formato *dias.horas:minutos:segundos*. Com apenas estes três atributos, a animação é executada apenas uma vez e pára. Quando queremos que ela execute mais de uma vez, usamos o atributo **RepeatBehavior**. O valor Forever faz que ela seja repetida indefinidamente. Caso quiséssemos que ela fosse repetida por três vezes apenas, colocaríamos **RepeatBehavior=”3x”**. Podemos também fazer que ela se repita por um intervalo de tempo, como 10 segundos, usando **RepeatBehavior=”0:0:10”**.

Quando uma animação é repetida mais de uma vez, ela reinicia do valor que definimos no atributo **From**. Se quisermos que ela seja executada na repetição de maneira inversa, usamos **AutoReverse=”True”**. Outra propriedade que determina a maneira que a animação é finalizada é a propriedade **FillBehavior**. Se configurarmos seu valor para **Stop**, a propriedade do objeto volta ao valor inicial. Se configurarmos para **HoldEnd**, a propriedade fica com seu valor final.

WPF - Animação

Caso quiséssemos fazer mais de uma animação simultaneamente, poderíamos colocá-las no elemento **Storyboard**. Por exemplo, para fazer que nosso retângulo fosse movimentado tanto na posição horizontal quanto na vertical, poderíamos fazer algo como:

```
<BeginStoryboard>
        <Storyboard TargetProperty="(Canvas.Left)">
                <DoubleAnimation From="0" To="800" Duration="0:0:5"  RepeatBehavior="Forever" AutoReverse="True"/>
        </Storyboard>
</BeginStoryboard>
<BeginStoryboard>
      <Storyboard TargetProperty="(Canvas.Top)">
            <DoubleAnimation From="0" To="800" Duration="0:0:5"  RepeatBehavior="Forever" AutoReverse="True"/>
      </Storyboard>
</BeginStoryboard>
```

WPF - Animação

Podemos também animar transformações do elemento. Por exemplo, para rodar o retângulo, poderíamos aplicar a seguinte animação:

```
<Canvas>
    <Canvas.Resources>
      <Style TargetType="{x:Type Rectangle}">
        <Setter Property="RenderTransform">
          <Setter.Value>
            <RotateTransform />
          </Setter.Value>
        </Setter>
      </Style>
    </Canvas.Resources>
    <Rectangle Canvas.Left="200" Canvas.Top="100" Width="200" Height="100" RadiusX="10" RadiusY="10" Fill="Red">
      <Rectangle.Triggers>
        <EventTrigger RoutedEvent="Rectangle.Loaded">
          <BeginStoryboard>
            <Storyboard >
              <DoubleAnimation  Storyboard.TargetProperty="RenderTransform.Angle"  From="0" To="360" Duration="0:0:15" />
            </Storyboard>
          </BeginStoryboard>
        </EventTrigger>
      </Rectangle.Triggers>
    </Rectangle>
  </Canvas>
```

WPF - Animação

Animação com Keyframes - Até agora, vimos como animar propriedades, com os valores variando de maneira contínua.

Porém, este nem sempre é o que queremos. Por exemplo, podemos querer que um objeto se movimente em ziguezague pela tela. Neste caso a maneira mais simples de fazer a animação é usando *keyframes*.

Este tipo de animação permite especificar os valores de uma propriedade a um determinado tempo. O exemplo seguinte mostra o retângulo andando em zigue-zague. Temos duas animações simultâneas, a primeira anima o movimento no eixo X e a segunda usa keyframes para animar o movimento no eixo Y: no instante 0, o valor de Y é 0, no instante 5, o valor de Y é 200, voltando a 0 e 200 nos instantes 10 e 15:

WPF - Animação

```
<BeginStoryboard>
    <Storyboard>
        <DoubleAnimation Storyboard.TargetProperty="(Canvas.Left)" Duration="0:0:15" From="0" To="600" />
        <DoubleAnimationUsingKeyFrames Storyboard.TargetProperty="(Canvas.Top)" Duration="0:0:15">
            <LinearDoubleKeyFrame Value="0"  KeyTime="0:0:0" />
            <LinearDoubleKeyFrame Value="200" KeyTime="0:0:5" />
            <LinearDoubleKeyFrame Value="0" KeyTime="0:0:10" />
            <LinearDoubleKeyFrame Value="200" KeyTime="0:0:15" />
        </DoubleAnimationUsingKeyFrames>
    </Storyboard>
</BeginStoryboard>
```

WPF - Animação

Quando queremos que a propriedade varie linearmente, usamos **LinearKeyframes**. Isto faz com que a propriedade desejada mude continuamente do valor atual até o final, no intervalo indicado pela keyframe. Se quisermos que esta variação não seja linear, mas siga uma sequência determinada por uma curva, podemos usar **SplineKeyframes**. Neste caso, os valores são determinados por uma curva de Bezier cúbica. Quando usamos **SplineKeyFrames** indicar os dois pontos de controle que irão determinar a curva, usando o atributo **KeySpline:**

WPF - Animação

```
<Canvas>
    <Rectangle Canvas.Top="100" Width="200" Height="100" RadiusX="10" RadiusY="10" Fill="Red">
      <Rectangle.Triggers>
        <EventTrigger RoutedEvent="Rectangle.Loaded">
          <BeginStoryboard>
            <Storyboard>
              <DoubleAnimation Storyboard.TargetProperty="(Canvas.Left)" Duration="0:0:15" From="0" To="600" />
              <DoubleAnimationUsingKeyFrames Storyboard.TargetProperty="(Canvas.Top)" Duration="0:0:15">
                <SplineDoubleKeyFrame Value="0" KeySpline="0,0.5 1,0.5"   KeyTime="0:0:0" />
                <SplineDoubleKeyFrame Value="200" KeySpline="0,0.5 1,0.5" KeyTime="0:0:5" />
                <SplineDoubleKeyFrame Value="0" KeySpline="0,0.5 1,0.5"   KeyTime="0:0:10" />
                <SplineDoubleKeyFrame Value="200" KeySpline="0,0.5 1,0.5" KeyTime="0:0:15" />
              </DoubleAnimationUsingKeyFrames>
            </Storyboard>
          </BeginStoryboard>
        </EventTrigger>
      </Rectangle.Triggers>
    </Rectangle>
  </Canvas>
```

WPF - Animação

Uma terceira maneira de variação da propriedade é a variação discreta, isto é, o valor da propriedade não varia no intervalo da keyframe. Isto é particularmente interessante como quando queremos animar uma propriedade que não tem estados intermediários, como **Visibility**. Neste caso, ela só tem 2 estados, ligada ou desligada. Podemos ligar e desligar a visibilidade do retângulo de 5 em 5 segundos com o seguinte código:

WPF - Animação

```
<BeginStoryboard>
     <Storyboard>
         <DoubleAnimation Storyboard.TargetProperty="(Canvas.Left)" Duration="0:0:15" From="0" To="600" />
         <ObjectAnimationUsingKeyFrames Storyboard.TargetProperty="(Visibility)" Duration="0:0:15">
              <DiscreteObjectKeyFrame Value="{x:Static Visibility.Visible}"   KeyTime="0:0:0" />
              <DiscreteObjectKeyFrame Value="{x:Static Visibility.Collapsed}" KeyTime="0:0:5" />
              <DiscreteObjectKeyFrame Value="{x:Static Visibility.Visible}"   KeyTime="0:0:10" />
              <DiscreteObjectKeyFrame Value="{x:Static Visibility.Collapsed}" KeyTime="0:0:15" />
         </ObjectAnimationUsingKeyFrames>
    </Storyboard>
</BeginStoryboard>
```

Neste caso, usamos uma **ObjectAnimationUsingKeyFrames**, que permite animar a propriedade de um objeto, alterando-a em cada keyframe. Ao executar este código, vemos que o retângulo se move por 5 segundos, desaparece, reaparecendo mais adiante e desaparecendo após 15 segundos.

WPF - Animação

Aceleração e desaceleração

Todas as animações vistas até agora têm uma velocidade constante. Porém, não é isso que vemos na vida real. Quando queremos criar uma animação semelhante à realidade, devemos considerar que os objetos não se movem com velocidade constante. Por exemplo, uma bola atirada para cima inicia com uma velocidade inicial, que é diminuída até ela atingir sua altura máxima, quando sua velocidade é zero e ela passa a cair e sua velocidade aumenta. Para atingir este efeito, temos as propriedades **AccelerationRatio e DecelerationRatio**, que irão determinar o tempo que a propriedade sofrerá aceleração ou desaceleração.

WPF - Animação

Estas propriedades têm valores entre 0 e 1, indicando o percentual de tempo do movimento que sofrerá ação da aceleração. No exemplo da bola, podemos criar uma animação em que a posição do objeto varia de 0 a 100 e que a propriedade **DecelerationRatio** seja 1 (isto indica que o movimento sofre ação da desaceleração todo o tempo do movimento). Com isso, a bola sobe até atingir seu ponto máximo. Se configurarmos a propriedade **AutoReverse** para True, a bola irá descer, com movimento acelerado (quando o movimento é invertido, a aceleração também o é) . O código a seguir movimenta a bola para cima com uma desaceleração igual a 1, dando a impressão da ação da gravidade sobre ela:

WPF - Animação

```
<Canvas>
      <Ellipse Canvas.Top="300" Width="100" Height="100" Fill="Red">
        <Ellipse.Triggers>
          <EventTrigger RoutedEvent="Rectangle.Loaded">
            <BeginStoryboard>
              <Storyboard >
                <DoubleAnimation From="300" To="0" Storyboard.TargetProperty="(Canvas.Top)" Duration="0:0:0.5"
  RepeatBehavior="Forever" AutoReverse="True" DecelerationRatio="1"/>
                <DoubleAnimation From="0" To="600" Storyboard.TargetProperty="(Canvas.Left)" Duration="0:0:3"
  RepeatBehavior="Forever" AutoReverse="True"/>
              </Storyboard>
            </BeginStoryboard>
          </EventTrigger>
        </Ellipse.Triggers>
      </Ellipse>
   </Canvas>
```