Анимация в WinForms

Необходимо заставить кусок текста вращаться в окне About приложения Windows Forms.

- Создайте таймер, который срабатывает периодически
- Когда таймер сработает, воспользуйтесь обработчиком событий для вычисления некоторых деталей анимации, таких как новый угол поворота. Затем сделайте недействительным все окно либо его часть.
- Сразу после этого Windows попросит окно перерисовать свое содержимое, запустив специальный код рисования.
- В коде рисования визуализируйте повернутый текст.

Возникающие проблемы

- Рисуются пиксели, а не элементы управления
- Подразумевается единственная анимация
- Частота кадров анимации фиксирована
- Сложность кода анимации растет в геометрической прогрессии

Анимация на основе свойств

Часто анимацию воспринимают как последовательность кадров. Чтобы выполнить анимацию, эти кадры отображаются друг за другом, подобно мультипликации. В WPF используется совершенно другая модель. По сути, анимация WPF — это просто способ модифицировать значение свойства зависимости через интервалы времени.

Классы анимации

Линейная интерполяция – последовательное изменение свойств от начального до конечного значения. Например, DoubleAnimation и ColorAnimation, иными словами классы линейно интерполяции носят имя ИмяТипаAnimation

Анимация ключевого кадра – применяется для изменение определенных типов данных, таких как строки и объекты, при этом, изменение происходит скачкообразно. Все классы ключевого кадра носят имя в формате ИмяТипаAnimationUsingKeyFrames — например, StringAnimationUsingKeyFrames и ObjectAnimationUsingKeyFrames.

Анимация на основе пути – модифицирует значение в соответствии с фигурой, описанной в объекте PathGeometry, и в первую очередь применяется для перемещения элемента по некоторому пути. Классы для анимации на основе пути имеют имена в стиле ИмяТипаAnimationUsingPath, например, DoubleAnimationUsingPath или PointAnimationUsingPath.

Классы анимации

В конечном итоге, вот что можно обнаружить в пространстве имен System.Windows.Media.Animation:

- 17 классов использующих анимацию методом интерполяции;
- 22 класса использующих анимацию ключевого кадра;
- 3 класса использующих анимацию на основе пути.

Анимация в коде

```
private void btn_Click(object sender, RoutedEventArgs e)
{
      DoubleAnimation da = new DoubleAnimation();
      da.From = 150;
      da.To = this.Width - 20;
      da.Duration = TimeSpan.FromSeconds(5);
      btn.BeginAnimation(Button.WidthProperty, da);
}
```

Время жизни анимации

Формально анимации WPF являются временными, а это означает, что они в действительности не изменяют значения лежащего в основе свойства. Пока анимация активна, она просто переопределяет значение свойства.

Однонаправленная анимация (как анимация роста кнопки) остается активной и после завершения ее работы. Это объясняется тем, что анимация должна удерживать ширину кнопки в новом размере. Это может привести к неожиданной проблеме, а именно: попытка модифицировать значение свойства в коде после завершения анимации никакого эффекта не дает. Причина в том, что код просто присваивает свойству новое локальное значение, но анимированное значение имеет приоритет перед ним.

Решение проблем

- Создать анимацию, которая сбрасывает элементы в их исходное состояние, либо создание обратной анимации посредством установки свойства AutoReverse в true.
- Изменить свойство **FillBehavior**. Изначально FillBehavior установлено в HoldEnd, а это означает, что когда анимация завершится, ее финальное значение будет применено к целевому свойству. Если изменить FillBehavior на Stop, то по завершении анимации свойство вернется к своему исходному значению.
- Удалить объект анимации по ее завершении, обработав событие **Completed** объекта анимации.(widthAnimation.Completed += animation_Completed;)

Класс TimeLine

- BeginTime - устанавливает задержку перед запуском анимации (как TimeSpan). Эта задержка добавляется к общему времени, так что пятисекундная анимация с пятисекундной задержкой займет в сумме десять секунд. Свойство BeginTime удобно для синхронизации разных анимаций, которые запускаются в одно и то же время, но должны выполнять свои действия последовательно.
- Duration - устанавливает длительность времени выполнения анимации, от старта до финиша

Класс TimeLine

SpeedRatio - увеличивает или уменьшает скорость анимации. Изначально SpeedRatio равно 1. Если его увеличить, то анимация завершится быстрее (например, SpeedRatio, равное 5, выполнит анимацию впятеро быстрее). Если уменьшить значение этого свойства, анимация замедлится (например, установка SpeedRatio в 0.5 приводит к получению анимации, выполняющейся вдвое дольше). Для получения того же результата можно также изменить свойство Duration анимации. Когда применяется задержка BeginTime, свойство SpeedRatio во внимание не принимается

Класс TimeLine

AccelerationRatio, DecelerationRatio - делает анимацию нелинейной, так что она запускается медленно, затем происходит ускорение (за счет увеличения AccelerationRatio) либо замедление (при увеличении DecelerationRatio). Оба значения находятся в промежутке от 0 до 1 и начинаются с 0. Кроме того, сумма обоих величин не может превышать 1.

AutoReverse - если это свойство равно true, то анимация будет запущена в обратном порядке, как только завершится. Если увеличить SpeedRatio, оно будет применено как к прямому воспроизведению анимации, так и к обратному. Свойство BeginTime применяется только в самом начале анимации — задержки при запуске в обратном направлении не происходит.

Класс TimeLine

FillBehavior - определяет то, что произойдет по завершении анимации. Обычно свойство остается зафиксированным в конечном значении (FillBehavior.HoldEnd), но можно также выбрать возврат к исходному значению (FillBehavior.Stop)

RepeatBehavior - позволяет повторить анимацию заданное количество раз либо в течение указанного интервала времени.

Раскадровка

Усовершенствованная временная шкала. Ее можно применять для группирования множества анимаций и, кроме того, она позволяет управлять воспроизведением анимации — приостанавливать ее, прекращать и изменять текущую позицию. Однако самым базовым средством, предлагаемым классом StoryBoard, является способность указывать на определенное свойство и определенный элемент, используя свойства TargetProperty и TargetName. Другими словами, раскадровка заполняет пробел между анимацией и свойством, для которого должна осуществляться анимация.

Пример

```
<Button MaxHeight="50" Width="160" Padding="5" Name="btn1"
                        Click="btn_Click" Margin="0,10,0,10" Content="Щелкни и я расширюсь ;)">
        <Button.Triggers>
                <EventTrigger RoutedEvent="Button.Click">
                        <EventTrigger.Actions>
                            <BeginStoryboard>
                                <Storyboard>
                                    <DoubleAnimation Storyboard.TargetProperty="Width" To="300"
                                                                        Duration="0:0:2"></DoubleAnimation>
                                </Storyboard>
                            </BeginStoryboard>
                        </EventTrigger.Actions>
                </EventTrigger>
        </Button.Triggers>
</Button>
```

Перекрывающиеся и синхронизированные анимации

Раскадровка предоставляет возможность изменять способ работы с перекрывающимися анимациями — другими словами, когда вторая анимация применяется к свойству, которое уже анимируется. Это делается через свойство BeginStoryboard.HandoffBehavior.

Обычно, когда анимации перекрываются, то вторая переопределяет первую немедленно.

Единственным альтернативным значением перечисления HandoffBehavior является Compose, которое приводит к объединению второй анимации с временной шкалой первой анимации.

Пример

```
<EventTrigger RoutedEvent="ListBoxItem.MouseLeave">
    <EventTrigger.Actions>
        <BeginStoryboard HandoffBehavior="Compose">
            <Storyboard>
                <DoubleAnimation Storyboard.TargetProperty="Fontslze"
                    BeginTime="0:0:0.5" Duration="0:0:0.2"></DoubleAnimation>
            </Storyboard>
        </BeginStoryboard>
    </EventTrigger.Actions>
</EventTrigger>
```

Синхронизированные анимации

Класс Storyboard непрямо унаследован от TimeLineGroup, что дает ему возможность поддерживать более одной анимации. Лучше всего то, что эти анимации управляются как единая группа — в том смысле, что запускаются одновременно.

Пример

```
<EventTrigger RoutedEvent="Button.Click">
    <EventTrigger.Actions>
        <BeginStoryboard>
            <Storyboard>
                <DoubleAnimation Storyboard.TargetProperty="Width"
                    To="300" Duration="0:0:5"></DoubleAnimation>
                <DoubleAnimation Storyboard.TargetProperty="Height"
                    To="300" Duration="0:0:5"></DoubleAnimation>
            </Storyboard>
        </BeginStoryboard>
    </EventTrigger.Actions>
</EventTrigger>
```

Управление воспроизведением

PauseStoryboard - приостанавливает воспроизведение анимации и сохраняет ее в текущей позиции

Resumestoryboard - возобновляет воспроизведение приостановленной анимации

StopStoryboard - останавливает воспроизведение анимации и сбрасывает ее часы в начало

Управление воспроизведением

SeekStoryboard - перепрыгивает в определенную точку временной шкалы анимации. Если анимация в данный момент воспроизводится, то воспроизведение продолжается с новой позиции. Если же анимация приостановлена, она остается приостановленной.

SetStoryboardSpeedRatio - Изменяет SpeedRatio всей раскадровки (а не только одной анимации внутри нее)

Управление воспроизведением

SkipStoryboardToFill - перемещает раскадровку в конец ее временной шкалы. Этот период известен как область заполнения (fill region). Для стандартной анимации, у которой свойство FillBehavior установлено в HoldEnd, анимация продолжается для удержания финального значения.

Removestoryboard - Удаляет раскадровку, прерывая все текущие выполняющиеся анимации и возвращая свойства в исходные, установленные последний раз значения. Это дает тот же эффект, что и вызов BeginAnimation() для соответствующего элемента с null-объектом анимации

Пример

```
<Grid>
            <Grid.RowDefinitions>
                <RowDefinition></RowDefinition>
                <RowDefinition Height="auto"></RowDefinition>
            </Grid.RowDefinitions>
            <Image Source="night.jpg"></Image>
            <Image Source="day.jpg" Name="imgDay"></Image>
            <StackPanel Grid.Row="1" Orientation="Horizontal" HorizontalAlignment="Center"
                        Margin="5">
                <StackPanel.Resources>
                    <Style TargetType="Button">
                        <Setter Property="Padding" Value="5"></Setter>
                        <Setter Property="Margin" Value="0,0,3,0"></Setter>
                    </Style>
                </StackPanel.Resources>
                <Button Name="cmd_Start">Старт</Button>
                <Button Name="cmd_Pause">Пауза</Button>
                <Button Name="cmd_Resume">Продолжить</Button>
                <Button Name="cmd_Stop">Стоп</Button>
                <Button Name="cmd_Middle">В середину</Button>
            </StackPanel>
</Grid>
```

Пример

```
<Window.Triggers>
        <EventTrigger SourceName="cmd_Start" RoutedEvent="Button.Click">
            <BeginStoryboard Name="fadeStoryBoardBegin">
                <Storyboard  Name="MyStoryboard">
                    <DoubleAnimation Storyboard.TargetProperty="Opacity" From="1"
                                     To="0" Storyboard.TargetName="imgDay" Duration="0:0:3"></DoubleAnimation>
                </Storyboard>
            </BeginStoryboard>
        </EventTrigger>
        <EventTrigger SourceName="cmd_Pause" RoutedEvent="Button.Click">
            <PauseStoryboard BeginStoryboardName="fadeStoryBoardBegin"></PauseStoryboard>
        </EventTrigger>
        <EventTrigger SourceName="cmd_Resume" RoutedEvent="Button.Click">
            <ResumeStoryboard BeginStoryboardName="fadeStoryBoardBegin"></ResumeStoryboard>
        </EventTrigger>
        <EventTrigger SourceName="cmd_Stop" RoutedEvent="Button.Click">
            <StopStoryboard BeginStoryboardName="fadeStoryBoardBegin"></StopStoryboard>
        </EventTrigger>
        <EventTrigger SourceName="cmd_Middle" RoutedEvent="Button.Click">
            <SeekStoryboard BeginStoryboardName="fadeStoryBoardBegin"
                            Offset="0:0:1.5"></SeekStoryboard>
        </EventTrigger>
</Window.Triggers>
```