

Анимация в WPF-приложениях обеспечивается постепенным изменением свойств элементов.

Типы анимации

Все классы анимации объявлены в пространстве имен `System.Windows.Media.Animation`. Имена классов анимации начинаются с имени типа свойства, для которого предназначена данная анимация. Примеры:

Имя класса анимации	Описание
<code>ByteAnimation...</code>	Анимация, предназначенная для изменения свойства типа <code>Byte</code>
<code>Int32Animation...</code>	Анимация, предназначенная для изменения свойства типа <code>Int32</code>
<code>DoubleAnimation...</code>	Анимация, предназначенная для изменения свойства типа <code>Double</code>
<code>BooleanAnimation...</code>	Анимация, предназначенная для изменения свойства типа <code>Boolean</code>
<code>ColorAnimation...</code>	Анимация, предназначенная для изменения свойства типа <code>Color</code>
<code>SizeAnimation...</code>	Анимация, предназначенная для изменения свойства типа <code>Size</code>
<code>StringAnimation...</code>	Анимация, предназначенная для изменения свойства типа <code>String</code>
<code>ThicknessAnimation...</code>	Анимация, предназначенная для изменения свойства типа <code>Thickness</code> (например, для свойств <code>Margin</code> , <code>Padding</code> , <code>BorderThickness</code>); примеры значений: «10» - задание значения для всех сторон; «5,10,5,20» - задание значений для разных сторон: 5 для левой стороны, 10 – для верхней, 5 – для правой, 20 – для нижней
<code>VectorAnimation...</code>	Анимация, предназначенная для изменения свойства типа <code>Vector</code>

Набор классов **<Тип>Animation** предназначен для линейного изменения значения свойства от начального (или текущего) до конечного за указанный промежуток времени. Конечное значение свойства может быть задано явно, либо как приращение к начальному (или текущему) значению свойства.

Набор классов **<Тип>AnimationUsingKeyFrames** предназначен для анимации с использованием ключевых кадров. Разработчик задает набор значений для изменяемого свойства, временные характеристики и способ перехода между этими значениями.

Набор классов **<Тип>AnimationUsingPath** предназначен для изменения значения свойства в соответствии с геометрическим путем.

Класс <Тип>Animation

Класс `<Тип>Animation` включает следующие основные свойства:

- **From** – начальное значение анимируемого свойства. Если свойство не задано, то в качестве начального значения используется текущее значение свойства.
- **To** – конечное значение анимируемого свойства. Может быть не определено, если задано свойство `By`.
- **By** – смещение, прибавляемое к начальному значению свойства для получения конечного значения свойства.

После завершения действия анимации измененные свойства **не** возвращаются к первоначальным значениям. Вернуть первоначальное значение свойства можно с помощью дополнительного объекта анимации, для которого не заданы свойства `To` и `By`. Т.е. если не заданы свойства `To` и `By`, то целевым значением свойства является начальное значение, заданное до начала серии анимаций.

Примеры:

```
<DoubleAnimation To="10" /> – изменение свойства от текущего значения до значения 10;  
<DoubleAnimation From="1" By="4" /> – изменение свойства от значения 1 до значения 5;
```

`<DoubleAnimation By="4" />` – увеличение значения свойства на 4 от текущего значения (если вызвать анимацию несколько раз, то каждый раз значение свойства будет увеличиваться);
`<DoubleAnimation />` – возвращение к первоначальному значению свойства, заданному до начала действия серии анимаций;

- **Duration** – временной интервал, за который осуществляется анимация в формате «часы:минуты:секунды».

Пример:

`<DoubleAnimation Duration="0:1:5.5" ... >` – анимация осуществляется за 1 минуту и 5.5 секунд;

- **BeginTime** – временной интервал задержки перед началом анимации в формате «часы:минуты:секунды».
- **AutoReverse** – если true, то по завершении анимации начнется обратная анимация к первоначальному состоянию; значение по умолчанию – false.
- **RepeatBehavior** – способ повторения анимации; если указано значение в формате «ЧИСЛОх» (например, «5х»), то анимация будет повторена указанное число раз; если указано значение «Forever» то анимация будет повторяться неограниченное количество раз.

Класс Storyboard – раскадровка

Элементы анимации объединяются в родительском элементе Storyboard (раскадровка), которому может быть присвоено некоторое имя:

```
<Storyboard x:Name="ButtonAnimation">
    <DoubleAnimation ... />
    <ColorAnimation ... />
    ...
</Storyboard>
```

Анимлируемое свойство определяется в присоединяемых свойствах Storyboard.TargetName и Storyboard.TargetProperty:

```
<Storyboard>
    <DoubleAnimation Storyboard.TargetName="ИМЯ_ЭЛЕМЕНТА" Storyboard.TargetProperty="ИМЯ_СВОЙСТВА" ... />
    <ColorAnimation Storyboard.TargetName="ИМЯ_ЭЛЕМЕНТА" Storyboard.TargetProperty="ИМЯ_СВОЙСТВА" ... />
    ...
</Storyboard>
```

Присоединяемые свойства пишутся в скобках:

```
Storyboard.TargetProperty="(Canvas.Left)"
Storyboard.TargetProperty="Background.(SolidColorBrush.Color)"
Storyboard.TargetProperty="Background.(RadialGradientBrush.GradientStops)[0].Color"
Storyboard.TargetProperty="Background.(LinearGradientBrush.GradientStops)[1].Offset"
```

Для всей раскадровки можно задать свойства AutoReverse и RepeatBehavior.

Запуск раскадровки осуществляется с помощью элемента BeginStoryboard, который соответствует не объекту, а действию. Раскадровка объявляется либо в самом элементе BeginStoryboard:

```
<BeginStoryboard>
    <Storyboard>
        ...
    </Storyboard>
</BeginStoryboard>
```

либо имя раскадровки указывается в атрибуте Storyboard элемента BeginStoryboard (для этого раскадровка должна быть определена ранее как ресурс с именем):

```
<BeginStoryboard Storyboard="{StaticResource ButtonStoryboard}" />
```

Анимацию можно не только запустить, но и остановить, приостановить и возобновить с помощью следующих элементов:

```
<StopStoryboard BeginStoryboardName="{StaticResource PanelStoryboard}" />  
<PauseStoryboard BeginStoryboardName="{StaticResource PanelStoryboard}" />  
<ResumeStoryboard BeginStoryboardName="{StaticResource PanelStoryboard}" />
```

Действия с раскадровками можно выполнять:

1) в триггере события `EventTrigger` при возникновении какого-либо события:

```
<EventTrigger RoutedEvent="СОБЫТИЕ">  
    Элементы BeginStoryboard, StopStoryboard, PauseStoryboard, ResumeStoryboard  
</EventTrigger>
```

2) в других типах триггеров (`Trigger`, `MultiTrigger`, `DataTrigger`, `MultiDataTrigger`) в коллекциях `EnterActions` (при срабатывании триггера) и `ExitActions` (при прекращении действия триггера):

```
<MultiDataTrigger ...>  
    ...  
    <MultiDataTrigger.EnterActions >
```

Элементы `BeginStoryboard`, `StopStoryboard`, `PauseStoryboard`, `ResumeStoryboard`, запускаемые при срабатывании триггера

```
</MultiDataTrigger.EnterActions>  
<MultiDataTrigger.ExitActions >
```

Элементы `BeginStoryboard`, `StopStoryboard`, `PauseStoryboard`, `ResumeStoryboard`, запускаемые при прекращении действия триггера

```
</MultiDataTrigger.ExitActions>  
</MultiDataTrigger>
```

Задание 1

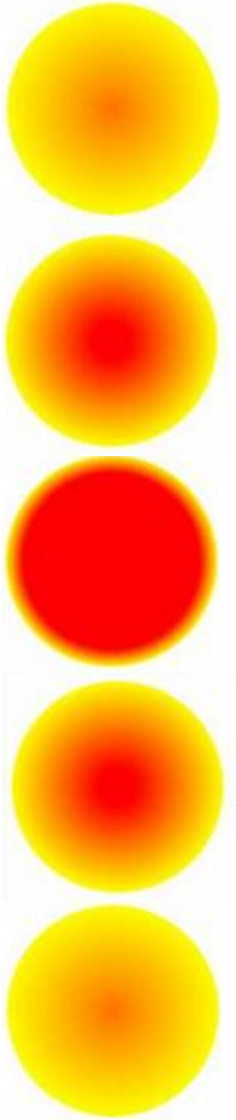
Добавьте в WPF-приложение `MultiEdit`, разработанное в лабораторной работе №5, эффекты анимации.

Задание 2

Разработайте WPF-приложение «Убегающая кнопка»: при наведении курсора мыши на кнопку она смещается на некоторое расстояние от курсора. Событие наведения курсора мыши – `MouseEnter`.

Задание 3

Разработайте WPF-приложение «Пульсар», изображающее круг, плавно меняющий свое состояние по следующей схеме:



и т.д.

Используйте элемент «Эллипс» с радиальной градиентной заливкой `RadialGradientBrush` для свойства `Fill` (заливка).