

Silverlight 4

فهرست مطالب

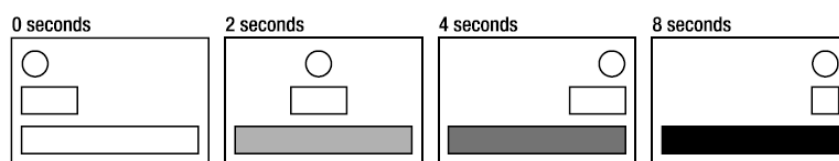
| | |
|--|-----|
| فصل ۲۳ – آشنایی با پویا نمایی در Silverlight | ۴۹۶ |
| مقدمه | ۴۹۶ |
| معرفی شیء Storyboard | ۴۹۶ |
| انواع پویانمایی در Silverlight | ۴۹۷ |
| مدیریت پویانمایی توسط برنامه نویسی | ۵۰۱ |
| آغاز یک پویانمایی به صورت خودکار و بدون استفاده از کد نویسی | ۵۰۴ |
| پویانمایی تغییر شکل اشیاء | ۵۰۵ |
| چرخش سه بعدی اشیاء به کمک پویانمایی و Perspective Transforms | ۵۰۶ |
| پویانمایی ظاهر نمایشی اشیاء Brush | ۵۰۸ |
| استفاده از شتاب دهنده‌های سخت افزاری | ۵۰۹ |
| معرفی متدهای Easing در پویانمایی | ۵۱۰ |

چاپ عمومی غیر رایگان این مطالب بدون مجوز کتبی از طرف نویسنده به هر نحوی غیرمجاز است.
انتشار این مطالب بر روی اینترنت و یا استفاده از آن به صورت مستقیم و یا غیر مستقیم در نشریات الکترونیکی با ذکر مأخذ بلا مانع است.

فصل ۲۳ - آشنایی با پویا نمایی در Silverlight

مقدمه

در گذشته برای ایجاد پویانمایی (Animation) در یک سایت عموماً به Adobe Flash رجوع می‌شد؛ اما اکنون با استفاده از امکانات Silverlight ایجاد پویانمایی با ابزارهایی که آن‌ها را می‌شناسید و همچنین توانایی کنترل آن‌ها را با برنامه نویسی دارید، بسیار ساده‌تر شده‌است. در Silverlight پیاده سازی پویانمایی بر اساس تغییر خاصیت یا خواصی از یک شیء در طول زمان است؛ برای مثال انتقال یک شیء در طول زمان از مکانی به مکان دیگر.



شکل ۱- مثالی از یک شیء Storyboard.

معرفی شیء Storyboard

در فیلم‌های کارتونی storyboard متشکل است از مجموعه‌ای از تصاویر که در طول زمان فیلم، پخش شده و حس حرکت و پویایی را القاء می‌کنند (بنابراین به آن timeline هم می‌توان گفت). مثالی از این مفهوم را در شکل ۱ می‌توانید ملاحظه نمایید. در اینجا در طی ۸ ثانیه تغییر شکل‌هایی به اشیاء موجود در صفحه اعمال شده‌است. اگر این پویانمایی را بخواهیم توسط Silverlight ارائه دهیم مراحل کار به نحو بعد خواهد بود:

- دو پویانمایی برای انتقال دایره و مستطیل کوچکتر از سمت چپ به سمت راست صفحه نیاز می‌باشند.
- یک پویانمایی برای تغییر رنگ پس زمینه مستطیل بزرگتر از سفید به مشکی نیاز خواهد بود.
- یک پویانمایی دیگر برای تغییر شکل مستطیل کوچکتر به مربع نیاز می‌باشد.

انواع پویانمایی در Silverlight

دو نوع پایه‌ای از پویانمایی در Silverlight موجود است:

۱. Linear interpolation animation : در این روش خواص اشیاء به صورت منظم و یکنواخت در طول زمان تغییر خواهند کرد.

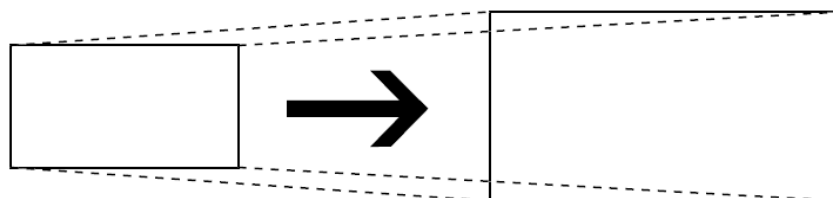
۲. Keyframe animation : در این روش تغییرات مقادیر بر اساس keyframes (فریم‌های کلیدی و اصلی) اضافه شده به یک نقطه‌ی مفروض در Storyboard است.

عموما این دو روش با هم ترکیب شده و پویانمایی یکنواخت و مطلوبی ارائه می‌گردد.

کلیه انواع پویانمایی در Silverlight از کلاس Timeline تعریف شده در فضای نام System.Windows.Media.Animation مشتق شده‌اند و شامل موارد زیر می‌باشند:

- ColorAnimation
- ColorAnimationUsingKeyFrames
- DoubleAnimation
- DoubleAnimationUsingKeyFrames
- ObjectAnimationUsingKeyFrames
- PointAnimation
- PointAnimationUsingKeyFrames

هر کدام از این انواع، جهت اعمال پویانمایی به مقادیر خاصی طراحی شده‌اند. برای مثال ColorAnimation سبب پویانمایی خاصیت Color بین دو شیء مفروض می‌گردد. به همین صورت DoubleAnimation برای پویانمایی خاصیت‌هایی از نوع عددی Double، PointAnimation برای پویانمایی خاصیتی از نوع Point و ObjectAnimation برای پویانمایی خاصیتی از نوع Object بکار می‌روند.



شکل ۲- پویانمایی ابعاد یک مستطیل

برای مثال به پویانمایی یک ابعاد یک مستطیل در طول زمان دقت بفرمائید (شکل ۲). برای پیاده سازی آن نیاز است از DoubleAnimationUsingKeyFrames استفاده گردد زیرا نوع‌های خواص طول و عرض یک مستطیل، از نوع Double می‌باشند. کدهای XAML این مثال را در ادامه ملاحظه خواهید نمود:

MainPage.xaml

```
<UserControl x:Class="SilverlightApplication89.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="
        http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d" d:DesignHeight="300" d:DesignWidth="400">
    <UserControl.Resources>
        <Storyboard x:Name="Storyboard1">
            <DoubleAnimationUsingKeyFrames
                BeginTime="00:00:00"
                Storyboard.TargetName="rectangle"
                Storyboard.TargetProperty="Width">
                <SplineDoubleKeyFrame
                    KeyTime="00:00:02"
                    Value="400"/>
            </DoubleAnimationUsingKeyFrames>
            <DoubleAnimationUsingKeyFrames
                BeginTime="00:00:00"
                Storyboard.TargetName="rectangle"
                Storyboard.TargetProperty="Height">
                <SplineDoubleKeyFrame
                    KeyTime="00:00:02"
                    Value="240"/>
            </DoubleAnimationUsingKeyFrames>
        </Storyboard>
    </UserControl.Resources>
    <Grid x:Name="LayoutRoot" Background="White" >
        <Rectangle
            Height="120"
            Width="200"
            HorizontalAlignment="Left"
            VerticalAlignment="Top"
            Stroke="#FF000000"
            x:Name="rectangle"/>
    </Grid>
</UserControl>
```

در این مثال ابتدا نام مستطیل صریحا مشخص شده است؛ این مورد ضروری است زیرا از این نام جهت پویا نمایی آن استفاده خواهیم کرد.

سپس در داخل یک شیء Storyboard دو پویانمایی تعریف شده است. مورد اول سبب تغییر عرض خواهد شد و مورد دوم ارتفاع شیء مستطیل را تغییر می‌دهد.

توسط خاصیت BeginTime مشخص خواهیم نمود که هر کدام از پویانمایی‌ها در طی چه زمانی از شروع Storyboard باید آغاز شوند که در اینجا همان زمان آغاز به کار Storyboard در نظر گرفته شده است.

خاصیت TargetName مشخص می‌سازد که قصد داریم پویانمایی را به کدام شیء تعریف شده در صفحه اعمال نمائیم که در اینجا هر دو مورد به شیء مستطیل تنظیم شده‌اند.

خاصیت TargetProperty به خاصیتی از شیء مقصد که قرار است مورد تغییر واقع شود اشاره می‌کند. در مثال فوق طول و عرض مستطیل قید شده‌اند.

در ادامه به کمک شیء SplineDoubleKeyFrame مشخص خواهیم ساخت که در طی چه مدت زمانی خاصیت ذکر شده در TargetProperty باید به مقداری که مشخص می‌نمائیم، تنظیم شود. برای مثال در حالت دوم طول مستطیل در طی ثانیه دوم به ۲۴۰ تنظیم خواهد شد :

```
<SplineDoubleKeyFrame KeyTime="00:00:02" Value="240"/>
```

برای مشاهده‌ی این پویانمایی کدهای صفحه را به صورت ذیل تغییر دهید:

MainPage.xaml.cs

```
using System.Windows;
namespace SilverlightApplication89
{
    public partial class MainPage
    {
        public MainPage()
        {
            InitializeComponent();
            this.Loaded += MainPage_Loaded;
        }
        void MainPage_Loaded(object sender, RoutedEventArgs e)
        {
            Storyboard1.Begin();
        }
    }
}
```

به مثالی دیگر در این زمینه دقت بفرمائید. در اینجا قصد داریم خاصیت شفافیت یک مستطیل را به وسیله‌ی پویانمایی تنظیم کنیم. کدهای XAML این مثال در آمده ذکر شده است و فراخوانی متد Begin آن همانند مثال قبل است.

MainPage.xaml

```
<UserControl x:Class="SilverlightApplication91.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">
```

```

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
xmlns:mc=
"http://schemas.openxmlformats.org/markup-compatibility/2006"
mc:Ignorable="d" d:DesignHeight="300" d:DesignWidth="400">
<UserControl.Resources>
    <Storyboard x:Name="Storyboard1">
        <DoubleAnimation
            Storyboard.TargetName="MyAnimatedRectangle"
            Storyboard.TargetProperty="Opacity"
            From="1.0"
            To="0.0"
            Duration="0:0:1"
            AutoReverse="True"
            RepeatBehavior="Forever" />
        </Storyboard>
    </UserControl.Resources>
    <Grid x:Name="LayoutRoot" Background="White">
        <Rectangle Height="100"
            Width="200"
            Fill="Bisque"
            Stroke="Gray"
            HorizontalAlignment="Center"
            StrokeThickness="3"
            x:Name="MyAnimatedRectangle"
            />
    </Grid>
</UserControl>

```

توضیحات:

توسط خاصیت TargetName نام شیء مورد نظر در عملیات پویانمایی ذکر شده است. سپس خاصیت Opacity این شیء به عنوان خاصیت تنظیم شونده در حین عملیات پویانمایی توسط ویژگی TargetProperty ذکر گردیده است.

با استفاده از ویژگی‌های From و To مشخص می‌نمائیم که در این حین، مقادیر شفافیت از چه عددی به عدد دیگر به شکلی یکنواخت و منظم تغییر خواهند کرد.

به کمک خاصیت Duration تعیین خواهیم نمود که تغییر مقادیر خواص شفافیت باید در طی چه مدت زمانی رخ دهد.

خاصیت RepeatBehavior به Forever تنظیم شده است تا این عملیات به صورت مداوم رخ دهد. با تنظیم AutoReverse به True سبب خواهیم شد تا عملیات مقدار دهی در جهت معکوس، پس از یکبار نمایش Storyboard رخ دهد و عملیات محو شدن زیبایی را شاهد باشیم. در این مثال بجای استفاده از خاصیت To می‌توان از خاصیت By نیز استفاده کرد. در این حالت مقادیر شفافیت هر بار به این اندازه افزایش خواهند یافت.

یکی از خواص Storyboard، FillBehavior نام دارد و دو مقدار HoldEnd و Stop را می‌پذیرد. در حالت HoldEnd، مقدار نهایی خاصیت تنظیم شده، ثابت نگه داشته خواهد شد (حالت پیش فرض) و در حالت Stop، مقدار نهایی حاصل مجدداً به همان مقدار اولیه بازگشت داده می‌شود.

مدیریت پویانمایی توسط برنامه نویسی

هنگامیکه پویانمایی شما تعریف شد، Silverlight نیاز دارد تا بداند چه زمانی باید آن را اجرا نماید. برای این منظور متدهای چندی برای کنترل پویانمایی تعریف شده‌اند که لیست آن‌ها را در جدول بعد ملاحظه خواهید نمود.

جدول ۱- متدهای متداول کار با Storyboard

| متد | توضیحات |
|----------|---|
| Begin() | سبب آغاز به کار Storyboard می‌گردد. |
| Pause() | سبب بروز مکث در عملیات Storyboard خواهد شد. |
| Resume() | کار Storyboard را مجدداً از سر خواهد گرفت. |
| Stop() | سبب توقف عملیات Storyboard می‌گردد. |
| Seek() | توسط آن می‌توان به قسمتی خاص از یک Storyboard رجوع کرد. |

لطفاً به مثال دیگری در این زمینه دقت بفرومائید. در این مثال یک مستطیل به صورت متناوب بزرگ و کوچک خواهد شد. به همین منظور دکمه‌هایی را جهت آغاز و متوقف سازی آن تعریف خواهیم کرد. کدهای XAML این مثال را در ادامه ملاحظه می‌نمائید:

MainPage.xaml

```
<UserControl x:Class="SilverlightApplication90.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="
        http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d" d:DesignHeight="300" d:DesignWidth="400">
    <UserControl.Resources>
        <Storyboard x:Name="MoveRect" RepeatBehavior="Forever">
            <DoubleAnimationUsingKeyFrames BeginTime="00:00:00"
                Storyboard.TargetName="rectangle"
                Storyboard.TargetProperty="Width">
```



```

        <SplineDoubleKeyFrame
            KeyTime="00:00:00"
            Value="200"/>
        <SplineDoubleKeyFrame
            KeyTime="00:00:03"
            Value="600"/>
        <SplineDoubleKeyFrame
            KeyTime="00:00:06"
            Value="200"/>
    </DoubleAnimationUsingKeyFrames>
    <DoubleAnimationUsingKeyFrames
        BeginTime="00:00:00"
        Storyboard.TargetName="rectangle"
        Storyboard.TargetProperty="Height">
        <SplineDoubleKeyFrame
            KeyTime="00:00:00"
            Value="100"/>
        <SplineDoubleKeyFrame
            KeyTime="00:00:03" Value="300"/>
        <SplineDoubleKeyFrame
            KeyTime="00:00:06" Value="100"/>
    </DoubleAnimationUsingKeyFrames>
</Storyboard>
</UserControl.Resources>
<StackPanel>
    <Rectangle Height="100"
        Width="200"
        Fill="Bisque"
        Stroke="Gray"
        HorizontalAlignment="Center"
        StrokeThickness="3"
        x:Name="rectangle"
        MouseLeftButtonUp="rectangle_MouseLeftButtonUp" />
    <StackPanel
        HorizontalAlignment="Center"
        Orientation="Horizontal">
        <Button Height="24" Margin="15"
            Content="Start"
            Width="100" x:Name="btnStart"
            Click="btnStart_Click" />
        <Button Height="24" Margin="15"
            Content="Stop"
            Width="100" x:Name="btnStop"
            Click="btnStop_Click" />
    </StackPanel>
</StackPanel>
</UserControl>

```

کدهای متناظر با این صفحه در ادامه ذکر شده‌اند:

MainPage.xaml.cs

```
using System.Windows;
using System.Windows.Input;

namespace SilverlightApplication90
{
    public partial class MainPage
    {
        private bool _paused;

        public MainPage()
        {
            InitializeComponent();
        }

        private void btnStart_Click(object sender,
            RoutedEventArgs e)
        {
            MoveRect.Begin();
        }

        private void btnStop_Click(object sender,
            RoutedEventArgs e)
        {
            MoveRect.Stop();
        }

        private void rectangle_MouseLeftButtonUp(object sender,
            MouseButtonEventArgs e)
        {
            if (_paused)
            {
                MoveRect.Resume();
                _paused = false;
            }
            else
            {
                MoveRect.Pause();
                _paused = true;
            }
        }
    }
}
```

توضیحات:

برای شروع به کار پویانمایی، کاربر باید بر روی دکمه‌ی Start کلیک نماید که سبب فراخوانی متد Begin مرتبط با Storyboard برنامه به نام MoveRect می‌شود. با توجه به تنظیم خاصیت RepeatBehavior به Forever، این پویانمایی تا زمانیکه مرورگر بسته نشده یا کاربر به صفحه‌ای دیگر رجوع نکند، ادامه خواهد یافت.

توسط دکمه‌ی Stop و فراخوانی متد Stop شیء Storyboard می‌توان این عملیات را متوقف کرد. همچنین اگر کاربر بر روی مستطیل در حال پویانمایی کلیک کند، در بار اول سبب مکث عملیات و در بار بعد سبب از سرگیری مجدد پویانمایی خواهد شد.

آغاز یک پویانمایی به صورت خودکار و بدون استفاده از کد نویسی

می‌توان زمان بروز رخدادی را به کمک EventTriggers تحت نظر قرار داده و عملی را توسط آن آغاز کرد. برای مثال می‌توان عملیات استاندارد BeginStoryboard را جهت آغاز خودکار پویانمایی بکار برد؛ که کدهای XAML آن را در ادامه ملاحظه می‌نمائید:

MainPage.xaml

```
<UserControl x:Class="SilverlightApplication92.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="
        http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d" d:DesignHeight="300" d:DesignWidth="400">
    <UserControl.Triggers>
        <EventTrigger>
            <EventTrigger.Actions>
                <BeginStoryboard>
                    <Storyboard x:Name="colorStoryboard">
                        <ColorAnimation
                            BeginTime="00:00:00"
                            Storyboard.TargetName="Rectangle1"
                            Storyboard.TargetProperty=
                                "(Rectangle.Fill).(SolidColorBrush.Color)"
                            From="Red"
                            To="Green"
                            AutoReverse="True"
                            RepeatBehavior="Forever"
                            Duration="0:0:4" />
                    </Storyboard>
                </BeginStoryboard>
            </EventTrigger.Actions>
        </EventTrigger>
    </UserControl.Triggers>
    <Grid x:Name="LayoutRoot" Background="White">
        <Rectangle
            Width="100"
            Height="100"
            Fill="Bisque">
```

```

        Name="Rectangle1"
    />
</Grid>
</UserControl>

```

EventTriggers در Silverlight برخلاف WPF بسیار محدود بوده و برای مثال تنها عملیات BeginStoryboard در اینجا پشتیبانی می‌گردد.

پویانمایی تغییر شکل اشیاء

در فصل برنامه نویسی گرافیکی با Silverlight، با المان‌های Transform جهت تغییر شکل اشیاء گرافیکی آشنا شدیم. در مثال بعد قصد داریم به کمک امکانات پویانمایی و المان‌های تغییر شکل، سبب چرخیدن مداوم یک مستطیل شویم:

MainPage.xaml

```

<UserControl x:Class="SilverlightApplication93.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="
        http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d" d:DesignHeight="300" d:DesignWidth="400">
    <UserControl.Triggers>
        <EventTrigger>
            <BeginStoryboard>
                <Storyboard x:Name="myStoryboard">
                    <DoubleAnimation
                        Storyboard.TargetName="myTransform"
                        Storyboard.TargetProperty="Angle"
                        From="0"
                        To="360"
                        Duration="0:0:5"
                        RepeatBehavior="Forever" />
                </Storyboard>
            </BeginStoryboard>
        </EventTrigger>
    </UserControl.Triggers>
    <StackPanel Margin="15">
        <Rectangle
            Width="50" Height="50"
            Fill="RoyalBlue">
            <Rectangle.RenderTransform>
                <RotateTransform
                    x:Name="myTransform"

```

```

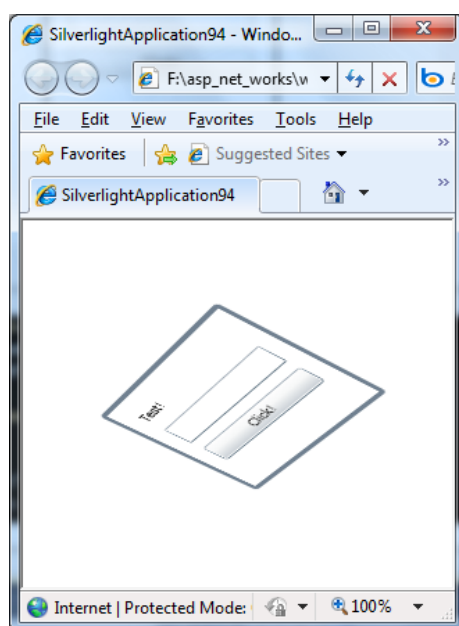
        Angle="45"
        CenterX="25"
        CenterY="25" />
    </Rectangle.RenderTransform>
</Rectangle>
</StackPanel>
</UserControl>

```

در این مثال خاصیت زاویه‌ی یک المان تغییرشکل چرخشی را به کمک پویانمایی از ۰ تا ۳۶۰ درجه تنظیم کرده‌ایم. این تغییرات در مقادیر خاصیت زاویه در طی ۵ ثانیه رخ داده و به صورت مداوم تکرار خواهد شد. همچنین به کمک EventTrigger تعریف شده، آغاز این پویا نمایی بدون نیاز به کدنویسی و خودکار خواهد بود.

چرخش سه بعدی اشیاء به کمک پویانمایی و Perspective Transforms

Silverlight از گرافیک سه بعدی به صورت توکار پشتیبانی نمی‌کند اما به کمک مفهومی به نام perspective transforms می‌توان یک سطح سه بعدی را در آن شبیه سازی کرد. در اینجا می‌توان یک صفحه‌ی مفروض را حول محورهای X، Y و یا Z چرخاند. این ویژگی به کلیه اشیاء Silverlight که دارای خاصیت Projection باشند قابل اعمال است. بنابراین کلیه عناصر UI شامل این لیست خواهند بود. در ادامه با تنظیم این زوایای چرخشی حول محورهای سه گانه، سبب چرخش مداوم یک فرم خواهیم شد (شکل ۳).



شکل ۳- چرخش سه بعدی یک فرم.

کدهای XAML این مثال را در ادامه ملاحظه خواهید نمود:

MainPage.xaml

```
<UserControl x:Class="SilverlightApplication94.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="
        http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d" d:DesignHeight="300" d:DesignWidth="400">
    <UserControl.Triggers>
        <EventTrigger>
            <BeginStoryboard>
                <Storyboard x:Name="spinStoryboard">
                    <DoubleAnimation
                        Storyboard.TargetName="projection1"
                        RepeatBehavior="Forever"
                        Storyboard.TargetProperty="RotationY"
                        From="0" To="360" Duration="0:0:3">
                    </DoubleAnimation>
                    <DoubleAnimation
                        Storyboard.TargetName="projection1"
                        RepeatBehavior="Forever"
                        Storyboard.TargetProperty="RotationZ"
                        From="0" To="360" Duration="0:0:30">
                    </DoubleAnimation>
                    <DoubleAnimation
                        Storyboard.TargetName="projection1"
                        RepeatBehavior="Forever"
                        Storyboard.TargetProperty="RotationX"
                        From="0" To="360" Duration="0:0:40">
                    </DoubleAnimation>
                </Storyboard>
            </BeginStoryboard>
        </EventTrigger>
    </UserControl.Triggers>
    <Border CornerRadius="2" Padding="10"
        Height="140" Width="170"
        BorderBrush="SlateGray"
        BorderThickness="4">
        <Border.Projection>
            <PlaneProjection x:Name="projection1"/>
        </Border.Projection>
        <StackPanel>
            <TextBlock Text="Test!" Margin="5"/>
            <TextBox Margin="5" />
            <Button Content="Click!" Margin="5"/>
        </StackPanel>
    </Border>
</UserControl>
```

```
</Border>
</UserControl>
```

در این مثال یک PlaneProjection با کمک خاصیت Projection شیء Border تعریف شده است. از این صفحه برای چرخش سه بعدی محتوای قرار گرفته شده‌ی داخل Border فوق کمک خواهیم گرفت؛ نحوه‌ی تغییر مقادیر مختلف آن‌را در Storyboard تعریف شده ملاحظه می‌نمائید.

پویانمایی ظاهر نمایشی اشیاء Brush

اعمال پویانمایی به اشیاء Brush نیز همانند سایر مثال‌هایی است که تاکنون مرور کرده‌ایم. یک شیء مقصد مشخص شده و سپس مقادیر خاصیتی مشخص را در طول زمان تغییر خواهیم داد. در مثال بعد پویانمایی از نوع PointAnimation را علاوه بر سایر حالات پیشین، بررسی خواهیم نمود:

MainPage.xaml

```
<UserControl x:Class="SilverlightApplication95.MainPage"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
  xmlns:mc="
    http://schemas.openxmlformats.org/markup-compatibility/2006"
  mc:Ignorable="d" d:DesignHeight="300" d:DesignWidth="400">
  <UserControl.Triggers>
    <EventTrigger>
      <BeginStoryboard>
        <Storyboard x:Name="ellipseStoryboard">
          <PointAnimation Storyboard.TargetName="ellipseBrush"
            Storyboard.TargetProperty="GradientOrigin"
            From="0.7,0.3" To="0.3,0.7"
            Duration="0:0:10" AutoReverse="True"
            RepeatBehavior="Forever">
          </PointAnimation>
          <ColorAnimation
            Storyboard.TargetName="ellipseBrushStop"
            Storyboard.TargetProperty="Color"
            To="Black" Duration="0:0:10"
            AutoReverse="True"
            RepeatBehavior="Forever">
          </ColorAnimation>
        </Storyboard>
      </BeginStoryboard>
    </EventTrigger>
  </UserControl.Triggers>
```

```

<Grid x:Name="LayoutRoot" Background="White">
  <Ellipse x:Name="ellipse" Margin="5"
    Grid.Row="1" Stretch="Uniform">
    <Ellipse.Fill>
      <RadialGradientBrush
        x:Name="ellipseBrush"
        RadiusX="1" RadiusY="1"
        GradientOrigin="0.7,0.3">
        <GradientStop
          x:Name="ellipseBrushStop"
          Color="White"
          Offset="0"/>
        <GradientStop
          Color="Blue" Offset="1"/>
      </RadialGradientBrush>
    </Ellipse.Fill>
  </Ellipse>
</Grid>
</UserControl>

```

در این مثال در حین پویانمایی، نقطه‌ی مرکزی گرادیان حلقوی تغییر مکان داده و همچنین رنگ گرادیان آن نیز از آبی به مشکی تغییر خواهد کرد.

اگر به این نوع پویانمایی علاقمند شده باشید، تیم WPF ابزاری را برای خلق ساده‌تر آن‌ها پدید آورده است که از آدرس ذیل قابل دریافت است:

<http://windowsclient.net/downloads/folders/controlgallery/entry2336.aspx>

استفاده از شتاب دهنده‌های سخت افزاری

استفاده از GPU (graphics processing unit) کارت‌های گرافیکی با توجه به امکان پردازش سریع‌تر یک سری از امور خاص گرافیکی توسط آن مانند تغییر ابعاد یک تصویر، سبب نمایش روان‌تر پویانمایی‌های ایجاد شده و همچنین کاهش مصرف CPU می‌گردد. تحت ویندوز کارت‌های گرافیکی سازگار با DirectX 9 به بعد برای این منظور مفید می‌باشند و یا در Mac OSX، کارت گرافیکی سیستم باید حداقل با OpenGL2 سازگار باشد. برای فعال سازی استفاده از امکانات GPU در Silverlight نیاز است تا پارامترهای شیء معرفی کننده‌ی افزونه‌ی Silverlight را اندکی ویرایش نمائیم. لطفاً به کدهای HTML بعد در این زمینه دقت بفرمائید.

HTML

```

...
<div id="silverlightControlHost">
  <object data="data:application/x-silverlight-2,"
    type="application/x-silverlight-2" width="100%" height="100%">

```



```
<param name="enableGPUAcceleration" value="true" />
<param name="enableCacheVisualization" value="true" />
<param name="enableFrameRateCounter" value="true" />
...
```

دو سطر اضافی `enableCacheVisualization` و `enableFrameRateCounter` تنها جهت Debug عملیات و مشاهده‌ی وضعیت فعلی و مقایسه‌ی آن با حالتی که استفاده از GPU فعال نگردیده است می‌باشند و لزومی به ارائه‌ی آن‌ها در محصول نهایی نیست.

معرفی متدهای Easing در پویانمایی

روش‌های پویانمایی که تاکنون معرفی شدند زیبا هستند اما عموماً غیر واقعی به نظر می‌رسند؛ زیرا در دنیای واقعی نیروهای بسیاری بر حرکت اجسام تاثیر گذارند مانند اصطکاک، جاذبه، نیروهای گریز از مرکز و غیره که سبب حرکت غیر خطی اجسام می‌گردند. ما در دنیایی با این واقعیات زندگی می‌کنیم و هر شبیه سازی که این مسایل فیزیکی را مد نظر قرار ندهد، غیرواقعی به نظر خواهد رسید. برای حل این مسایل تیم Silverlight متدهای Easing را معرفی کرده‌اند.

استفاده از متدهای Easing بسیار ساده است. تنها کافی است یکی از آن‌ها را در قسمت منابع صفحه تعریف نموده و سپس با کمک خاصیت `EasingFuncion`، این متد را به پویانمایی خود اضافه نمایید. لطفاً به مثال بعد در این زمینه دقت بفرمائید:

MainPage.xaml

```
<UserControl x:Class="SilverlightApplication96.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="
        http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d" d:DesignHeight="300" d:DesignWidth="400">
    <UserControl.Resources>
        <SineEase x:Key="easeOut" EasingMode="EaseOut" />
    </UserControl.Resources>
    <UserControl.Triggers>
        <EventTrigger>
            <BeginStoryboard>
                <Storyboard x:Name="animRXOut">
                    <DoubleAnimation
                        To="360" Duration="00:00:03"
                        EasingFunction="{StaticResource easeOut}"
```

```

        Storyboard.TargetName="rt1"
        Storyboard.TargetProperty="Angle" />
    </Storyboard>
</BeginStoryboard>
</EventTrigger>
</UserControl.Triggers>
<Grid x:Name="LayoutRoot" Background="White">
    <Ellipse Name="ball1"
        Stroke="Black"
        Width="100" Height="50" >
        <Ellipse.RenderTransform>
            <RotateTransform Angle="0" x:Name="rt1" />
        </Ellipse.RenderTransform>
    </Ellipse>
</Grid>
</UserControl>

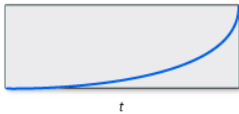
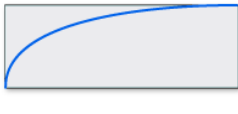
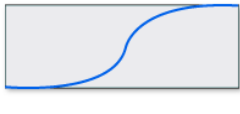
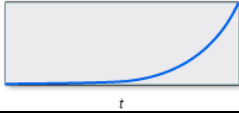
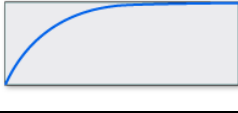
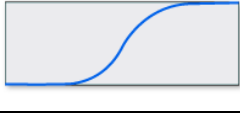

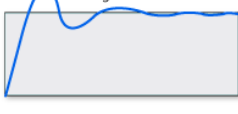
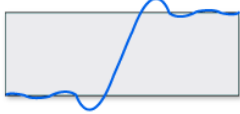
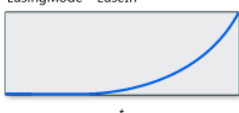

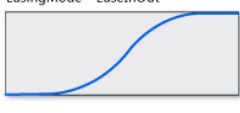
```

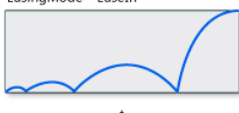


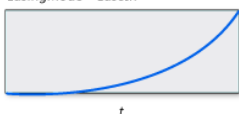

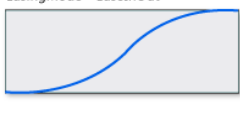
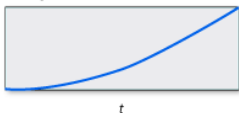
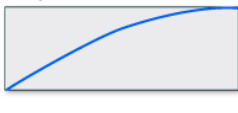
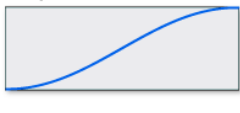
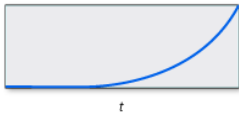
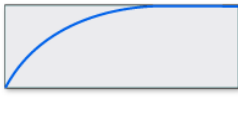
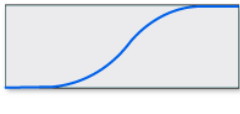


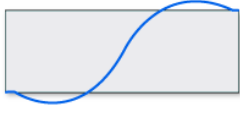
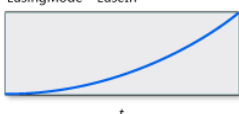
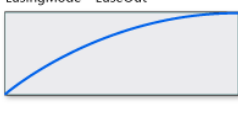

توسط EasingMode مشخص می‌شود که چه زمانی باید تابع Easing اعمال گردد و سه مقدار EaseIn، EaseOut و EaseInOut را می‌تواند بپذیرد (در زمان شروع پویا نمایی اعمال شود، در زمان پایان کار اعمال شود یا در هر دو زمان آغاز و پایان اعمال گردد). خلاصه‌ی متدهای Easing به همراه اثرات آن‌ها بر پویانمایی را در جدول بعد ملاحظه خواهید نمود.

لطفا جهت مشاهده‌ی مثالی جامع تهیه شده توسط تیم Silverlight در این زمینه به آدرس زیر مراجعه نمائید:

<http://tinyurl.com/animationeasing>

جدول ۲- متدهای گوناگون Easing و اثرات آن‌ها بر پویا نمایی

| اثرات آن بر پویا نمایی | | | متد Easing |
|---|---|--|-------------|
|  |  |  | CircleEase |
|  |  |  | QuinticEase |
|  |  |  | ElasticEase |
|  |  |  | QuarticEase |

| | | | |
|--|---|--|-----------------|
|  EasingMode="EaseIn" |  EasingMode="EaseOut" |  EasingMode="EaseInOut" | BounceEase |
|  EasingMode="EaseIn" |  EasingMode="EaseOut" |  EasingMode="EaseInOut" | CubicEase |
|  EasingMode="EaseIn" |  EasingMode="EaseOut" |  EasingMode="EaseInOut" | ExponentialEase |
|  EasingMode="EaseIn" |  EasingMode="EaseOut" |  EasingMode="EaseInOut" | PowerEase |
|  EasingMode="EaseIn" |  EasingMode="EaseOut" |  EasingMode="EaseInOut" | BackEase |
|  EasingMode="EaseIn" |  EasingMode="EaseOut" |  EasingMode="EaseInOut" | SineEase |