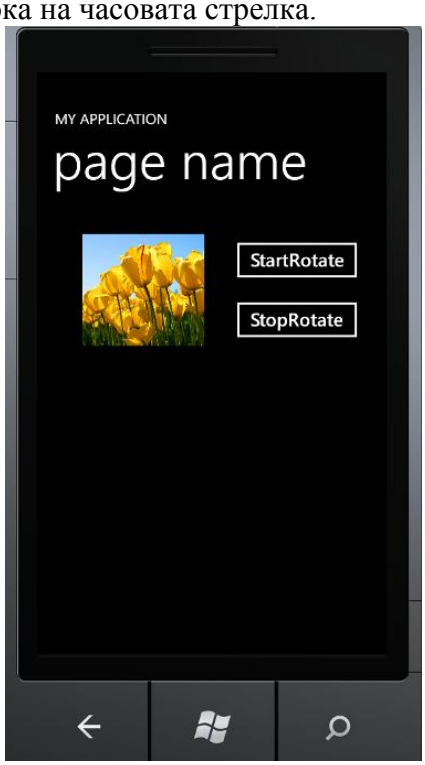
# Упражнение 5

Създаване на **Windows Phone** приложение, което прилага трансформации към елементи на потребителския интерфейс и реализира анимация на свойства на обекти.

Целта на приложението е да илюстрира:

- възможността за прилагане на трансформации (транслация, ротация и мащабиране) към елементи на поребителския интерфейс в **Windows Phone** приложение;
- възможността за създаване на многонишково **Windows Phone** приложение на базата на класа DispatcherTimer;
- възможността за анимиране на свойства на обекти от различен тип (**PointAnimation, ColorAnimation, DoubleAnimation**) чрез обект от клас **Storyboard**.

**Трансформации.**
Всички елементи на потрбителския интерфейс, които наследяват класа **FrameworkElement** притежават свойство **RenderTransform,** което приема като стойност обект от клас **TranslateTransform, RotateTransform** или **ScaleTransform**. Именно свойство **RenderTransform** на контрол **Image** се използва в настоящето упражнение за мащабиране на показаното в контрола изображение **Image**. Същото свойство се използва за завъртане на изображението, показано в контрола **Image,** всяка секунда на 10 градуса в посока на часовата стрелка.




**Фиг.1. Общ вид на екрана на приложението за задача 1**

**Задача 1.** Да се създаде **Windows Phone** приложение, което да включва конторол **Image** и два контрола **Button**. В контрола **Image** да се изведе изображение, което да се мащабира (да се използва класът **ScaleTransform**) при докосване и влачене по екрана

на мобилния телефон (да се използва събитие **ManipulationDelta)**; при влачене в посока нагоре изображението да се увеличава, а при влачене в посока надолу – да се намалява. Да се създаде обект от клас **DispatcherTimer**, като се установи свойство **Interval**, равно на 1 секунда и се създаде метод за обслужване на събитието **Tick**. Методът за обслужване на събитието **Tick** да увеличава ъгъла на завъртане на изображението (свойство **Angle** на обект от клас **RotateTransform**). Първият контрол **Button** да се използва за стартиране на въртенето, а вторият контрол **Button** да се използва за спиране на въртенето.

За създаване на приложението трябва да се изпълнят следните стъпки.

**1.** Създайте нов проект на Visual C# (Silverlight), като използвате шаблона **Windows Phone Application.**

**2.** В режим на проектиране (**MainPage.xaml**) от прозореца **ToolBox** добавете един контрол **Image** и два контрола **Button.**

3. Присъединете към приложението избрано от Вас изображение (.jpg файл), като го направите част от .XAP файла на приложението, следвайки процедурата описана в Упражнение 1. (За целите на настоящето упражнение е използван файлът **Tulips.jpg**);

4. В кода на **XAML** за контрола **Image** добавете атрибут **Source = “Tulips.jpg”**, в резултат на което изображението трябва да се покаже на екрана.

5. За контрола **Image** създайте метод за обслужване на събитието **ManipulationDelta**.

6. В кода на C# за приложението (**MainPage.xaml.cs)** в класа **MainPage** добавете поле данни от тип **ScaleTransform**. В конструктора на класа **MainPage** инициализирайте полето данни с обект от класа **ScaleTransform**. За обекта от класа **ScaleTransform** установете свойства **CenterX, CenterY, ScaleX** и **ScaleY** така, че да зададете център на мащабиране в средата на изображението и мащабиращи фактори по двете координатни оси, равни на 1 (няма мащабиране).
Кодът на **MainPage.xaml.cs** към момента изглежда по следния начин (нововъведеният текст е подчертан):

```
public partial class MainPage : PhoneApplicationPage
    {
        private ScaleTransform s;  // поле данни от тип ScaleTransform
        // Constructor
        public MainPage()
        {
            InitializeComponent();
            s = new ScaleTransform();
            s.CenterX = image1.Width / 2;
            s.CenterY = image1.Height / 2;
            s.ScaleX = 1;
            s.ScaleY = 1;
        }
    }
}
```

7. В кода на метода за обработка на събитието **ManipulationDelta** използвайте параметър **e** от тип **ManipulationDeltaEventArgs** на метода, за да получите отместването по оста **Y**. Анализирайте знака на отместването и ако e > 0 намалете мащабиращия фактор по двете оси с 0.1; в противен случай – увеличете мащабиращия фактор по двете оси с 0.1. Установете свойство **RenderTransform** на контрол **Image** със създадения обект от клас **ScaleTransform**. Кодът на метода за обработка на събитието **ManipulationDelta** би трябва да изглежда по следния начин:

```
private void image1_ManipulationDelta(object sender, ManipulationDeltaEventArgs e)
{
       double deltaY = e.DeltaManipulation.Translation.Y;
       if (deltaY < 0)  //влачението е срещу положителната посока на оста Y
       {
            s.ScaleX += 0.1;
            s.ScaleY += 0.1;
       }
       else //влачението е в положителната посока на оста Y
       {
            s.ScaleX -= 0.1;
            s.ScaleY -= 0.1;
       }
       image1.RenderTransform = s;
}
```

8. В кода на C# за приложението (**MainPage.xaml.cs)** в класа **MainPage** добавете поле данни **r** от тип **RotateTransform.** В конструктора на класа **MainPage** инициализирайте полето данни **r** с обект от класа **RotateTransform**. За обекта от класа **RotateTransform** установете свойства **CenterX, CenterY** и **Angle** така, че да зададете център на ротация в средата на изображението и ъгъл на завъртане, равен на 0 (няма завъртане).

9. В кода на C# за приложението (**MainPage.xaml.cs)** добавете пространство на имената **System.Windows.Threading** (в него е дефиниран класът **DispatcherTimer**). В класа **MainPage** добавете поле данни **timer** от тип **DispatcherTimer.** В конструктора на класа **MainPage** инициализирайте полето данни **timer** с обект от класа **DispatcherTimer**. За обекта **timer** установете свойство **Interval** равно на 1 секунда и прикачете делегат от тип **EventHandler** към списъка от делегати за събитие **Tick**; като параметър на конструктора на делегата посочете името на метода за обслужване на събитието **Tick - OnTimerTick.** Кодът на **MainPage.xaml.cs** към момента би трябвало да изглежда по следния начин (нововъведеният текст е подчертан):

```
using System.Windows.Threading;
namespace Exercise5 {
public partial class MainPage : PhoneApplicationPage
    {
        private ScaleTransform s;
        private RotateTransform r; // поле данни от тип RotateTransform
        DispatcherTimer timer; // поле данни от тип DispatcherTimer

        // Constructor
        public MainPage()
        {
            InitializeComponent();

            s = new ScaleTransform();
```

```
            s.CenterX = image1.Width / 2;
            s.CenterY = image1.Height / 2;
            s.ScaleX = 1;
            s.ScaleY = 1;

            r = new RotateTransform();
            r.CenterX = image1.Width / 2;
            r.CenterY = image1.Height / 2;
            r.Angle = 0;

            timer = new DispatcherTimer();
            timer.Interval = TimeSpan.FromSeconds(1);
            timer.Tick += new EventHandler(OnTimerTick);
        }
}
```

10. Създайте метода **OnTimerTick()** за обслужване на събитието **Tick** на обекта от клас **DispatcherTimer**. В кода на метода **OnTimerTick()** увеличете ъгъла на завъртане – свойство **Angle** на обекта **r** от клас **RotateTransform**. Установете свойство **RenderTransform** на контрол **Image** със създадения обект от клас **RotateTransform**. Кодът на метода би трябвало да изглежда по следния начин:

```
//метод за обслужване на събитието Tick; извиква се всяка секунда
private void OnTimerTick(object sender, EventArgs e)
{
      r.Angle += 10; //увеличава ъгъла на завъртане
      image1.RenderTransform = r;

}
```

11. В кода на **XAML** за първия контрол **Button** променете атрибут **Content = “StartRotate”** и атрибут **Name = “startRotate”.** Създайте метод за обслужване на събитието **Click** за бутона**.**

12. В кода на метода за обслужване на събитието **Click** за бутона **StartRotate** извикайте метод **Start()** на обекта **timer** от клас **DispatcherTimer.** Кодът на метода би трябва да изглежда по следния начин:

```
private void startRotate_Click(object sender, RoutedEventArgs e)
{
       timer.Start();
}
```

13. В кода на **XAML** втория контрол **Button** променете атрибут **Content = “StopRotate”** и атрибут **Name = “stopRotate”.** Създайте метод за обслужване на събитието **Click** за бутона.

14. В кода на метода за обслужване на събитието **Click** за бутона **StopRotate** извикайте метод **Stop()** на обекта **timer** от клас **DispatcherTimer.** Кодът на метода би трябва да изглежда по следния начин:

```
private void stopRotate_Click(object sender, RoutedEventArgs e)
{
       timer.Stop();
```

```
}
```

15. Построете и изпълнете приложението.

Пълният код на C# на приложението изглежда по следния начин:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Animation;
using System.Windows.Shapes;
using Microsoft.Phone.Controls;
using System.Windows.Threading;

namespace Exercise5
{
    public partial class MainPage : PhoneApplicationPage
    {
        private ScaleTransform s;
        private RotateTransform r;
        DispatcherTimer timer;
        // Constructor
        public MainPage()
        {
            InitializeComponent();
            s = new ScaleTransform();
            s.CenterX = image1.Width / 2;
            s.CenterY = image1.Height / 2;
            s.ScaleX = 1;
            s.ScaleY = 1;

            r = new RotateTransform();
            r.CenterX = image1.Width / 2;
            r.CenterY = image1.Height / 2;
            r.Angle = 0;

            timer = new DispatcherTimer();
            timer.Interval = TimeSpan.FromSeconds(1);  //интервал 1 сек.
            timer.Tick += new EventHandler(OnTimerTick);
        }
        //метод за обслужване на събитието Tick
        private void OnTimerTick(object sender, EventArgs e)
        {
            r.Angle += 10; //увеличава ъгъла на завъртане всяка секунда
            image1.RenderTransform = r;
        }

        private void startRotate_Click(object sender, RoutedEventArgs e)
        {
            timer.Start();
        }

        private void stopRotate_Click(object sender, RoutedEventArgs e)
```

```
        {
            timer.Stop();
        }

        private void image1_ManipulationDelta(object sender,
ManipulationDeltaEventArgs e)
        {
            double deltaY = e.DeltaManipulation.Translation.Y;
            if (deltaY < 0)  //влаченето е срещу положителната посока на оста Y
            {
                s.ScaleX += 0.1;
                s.ScaleY += 0.1;
            }
            else
            {
                s.ScaleX -= 0.1;
                s.ScaleY -= 0.1;
            }
            image1.RenderTransform = s;
        }
    }
}
```

Актуализираният код на XAML изглежда по следния начин (добавеният код е подчертан):

```
<Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
            <Image ManipulationDelta="image1_ManipulationDelta" Source="Tulips.jpg"
Height="154" HorizontalAlignment="Left" Margin="50,29,0,0" Name="image1"
Stretch="Fill" VerticalAlignment="Top" Width="167" />
            <Button Content="StartRotate" Height="72" HorizontalAlignment="Left"
Margin="250,29,0,0" Name="startRotate" VerticalAlignment="Top" Width="188"
Click="startRotate_Click" />
            <Button Content="StopRotate" Height="72" HorizontalAlignment="Left"
Margin="250,111,0,0" Name="stopRotate" VerticalAlignment="Top" Width="188"
Click="stopRotate_Click" />
</Grid>
```

**Анимация**

Silverlight за **Windows Mobile** осигурява два начина на анимиране (базово анимиране):

- Промяна на стойносттите на свойства на обекти;
- Ключова анимация (*key frames*).

Анимират се свойства от различен тип, като в зависимост от типа на анимираното свойство се използва различен тип *анимационен обект*:

- **числов тип (**double, int) **(**например: свойство **Width** на обект **Rectangle;** свойство **RadiusX/RadiusY** на обект **Ellipse** и т.н.); използва се анимационен обект от клас **DoubleAnimation;**
- тип **Color** (например свойство **Backround** на обекти **StackPanel**, **Grid, Button** и др.); използва се анимационен обект от клас **ColorAnimation**;
- тип **Point** (например свойство **Center** на обект **Ellipse**); използва се анимационен обект от клас **PointAnimation**.

Анимирането на свойство на обект включва следните задължителни стъпки:

**1. Определяне кой обект и кое неговото свойство ще се анимира.**
Във връзка с тази стъпка обикновено се задава стойност на свойство **Name** на анимирания обект. Допустимо е да се запази стойността по подразбиране на свойство **Name** на анимирания обект. Посочването на нова стойност на свойство **Name** на анимирания обект прави по-лесно посочването на този обект като цел (target) на анимацията.

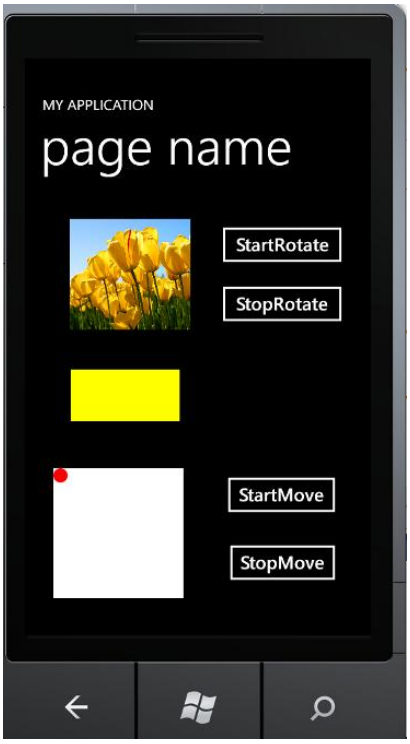
**2. Създаване на обект от клас Storyboard**
Обектът от клас **Storyboard** е задължителен за анимацията. Задължително условие е обектът от клас **Storyboard** да се обяви за ресурс на така наречения базов (root) обект. Например, ако както обикновено ролята на базов обект се изпълнява от обект от клас **StackPanel** или **Grid**, обектът от клас **Storyboard** трябва да се направи ресурс на обекта от клас **StackPanel** или на обекта от клас **Grid**.

**3. Свързване на анимирания обект към обекта от клас Storyboard.**
Свързването на анимирания обект към обекта **Storyboard** става с *анимационен обект*.

*Анимационният обект* определя какво точно се анимира (кой обект и по-конкретно кое негово свойство) и особеностите на изпълнението на анимацията (начална и крайна стойност на свойството, продължителност на анимацията в hh:mm:ss и др). Основните свойства на *анимационния обект*, които трябва да бъдат установени включват:

- Обект, който се анимира: **Storyboard.TargetName**="обект";
- Свойство на обекта, което се анимира (**Storyboard.TargetProperty** = „свойство”);
- Начална и крайна стойност на анимираното свойство (**From**="стойност" **To**="стойност").
- Продължителност на анимацията (**Duration**="часове:минути:секунди")
- Обръщане на анимацията (при достигане края на анимацията тя се повтаря, но в обратен ред (**AutoReverse**="True/False").
- Режим на повторение на анимацията (**RepeatBehavior**="Forever")

**4. Стартиране на анимацията**
Стартирането на анимацията става обикновено в резултат на събитие. Вътре в метода за обслужване на събитието трябва просто да се извика метод **Begin**() за обекта от класа **Storyboard**. Освен метод **Begin()** обектът от клас **Storyboard** предоставя още методи **Stop()**, **Pause()** и **Resume()** за управление на анимацията.




**Фиг.2. Общ вид на екрана на приложението за задача 2, надграждаща задача 1**

**Задача 2.** Да се допълни **Windows Phone** приложението, създадено в **Задача 1,** като се добави контрол **Rectangle**. Да се анимира свойство **Width** на контрола **Rectangle**, като при кликване с левия бутон на мишката върху контрола да започва плавно изменение на неговата ширина (свойство **Width**) от посочена начална до посочена крайна стойност; при достигане на крайната стойност, анимацията да продължава в обратна посока - от посочената крайна до посочената начална стойност на свойство **Width**; анимацията да се повтаря непрекъснато до затваряне на приложението. Към **Windows Phone** приложението да се добави контрол **Canvas** и два контрола **Button**. В контрол **Canvas** да се разположи графичен обект **EllipseGeometry**. При кликване върху първия от двойката контроли **Button** да се стартира преместване на обекта **EllipseGeometry** от посочена начална точка до посочена крайна точка. При кликване върху втория бутон от двойката контроли **Button**, движението на обекта **EllipseGeometry** да спре.

1. В режим на проектиране (**MainPage.xaml**) към проекта от задача 1 добавете контрол **Rectangle** като го привлачите от прозореца **ToolBox.**

2. В режим на проектиране (**MainPage.xaml**) директно в кода на **XAML** за обекта **Rectangle** установете атрибут **Fill=”Yellow”** и създайте метод за обслужване на събитието **MouseLeftButtonUp**.

3. В режим на проектиране (**MainPage.xaml**) директно в кода на **XAML** създайте обект от клас **Storyboard** като ресурс на контрола **Grid;** установете атрибут **Name = “myFirstStoryboard”**.

4. В режим на проектиране (**MainPage.xaml**) директно в кода на **XAML** създайте анимационен обект от клас **DoubleAnimation** (Тъй като типа на анимираното свойство **Width** на анимирания обект **Rectangle** е от числов тип трябва да се използва анимационен обект от клас **DoubleAnimation**). Свойствата на обекта от клас **DoubleAnimation**, с които се управлява анимацията установете както следва:

- Анимиран обект **Storyboard.TargetName="rectangle1";**
- Анимирано свойство на анимирания обект **Storyboard.TargetProperty="Width"**
- Начална стойност на анимираното свойство **From="20"**
- Крайна стойност на анимираното свойство **To="200"**
- Продължителност на анимацията 5 секунди **Duration="0:0:5"**
- Обръщане на анимацията **AutoReverse="True"**
- Повторение на анимацията **RepeatBehavior="Forever"**

Актуализираният код на XAML изглежда по следния начин (подчертан е добавеният код за задача 2):

```
<!--ContentPanel - place additional content here-->
        <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
            <Grid.Resources>
                <Storyboard x:Name="myFirstStoryboard">
                    <DoubleAnimation Storyboard.TargetName="rectangle1"
Storyboard.TargetProperty="Width" From="20" To="200" Duration="0:0:5"
AutoReverse="True" RepeatBehavior="Forever"/>
                </Storyboard>
            </Grid.Resources>
            <Image ManipulationDelta="image1_ManipulationDelta"  Source="Tulips.jpg"
Height="154" HorizontalAlignment="Left" Margin="50,29,0,0" Name="image1"
Stretch="Fill" VerticalAlignment="Top" Width="167" />
            <Button Content="StartRotate" Height="72" HorizontalAlignment="Left"
Margin="250,29,0,0" Name="startRotate" VerticalAlignment="Top" Width="188"
Click="startRotate_Click" />
            <Button Content="StopRotate" Height="72" HorizontalAlignment="Left"
Margin="250,111,0,0" Name="stopRotate" VerticalAlignment="Top" Width="188"
Click="stopRotate_Click" />
            <Rectangle Fill="Yellow" MouseLeftButtonUp="rectangle1_MouseLeftButtonUp"
HorizontalAlignment="Left" Margin="50,237,0,0" Name="rectangle1" Stroke="Black"
StrokeThickness="1" Width="153" Height="74" VerticalAlignment="Top" />
        </Grid>
```

5. В кода на метода за обслужване на събитието **MouseLeftButtonUp** на обекта от клас **Rectangle** извикайте метод **Begin**() за обекта **myFirstStoryboard** от клас **Storyboard**. Кодът на метода би трябва да изглежда по следния начин:

```
private void rectangle1_MouseLeftButtonUp(object sender, MouseButtonEventArgs e)
{
      myFirstStoryboard.Begin();
}
```

6. В режим на проектиране (**MainPage.xaml**) добавете към приложението контрол **Canvas** и два контрола **Button** като ги привлачите от прозореца **ToolBox.**

**7.** В кода на **XAML** за контрола **Canvas** установете атрибут **Background = “White”,** атрибут **Width = “180”** и атрибут **Height = “180”.**

7. В кода на **XAML** за първия контрол **Button** установете атрибут **Content = “StartMove”** и атрибут **Name = “startMove”.** Създайте метод за обслужване на събитието **Click** за бутона**.**

8. В кода на **XAML** за втория контрол **Button** установете атрибут **Content = “StopMove”** и атрибут **Name="stopMove".** Създайте метод за обслужване на събитието **Click** за бутона**.**

9. В кода на **XAML** вътре в контрола **Canvas** създайте елемент **Path,** за който установете атрибут **Fill = “Red”.**

10. В кода на **XAML** за елемента **Path** установете атрибут **Data**, като го инициализирате с елемент от клас **EllipseGeometry.** За елемента **EllipseGeometry** установете атрубути **x:Name = „animatedEllipse”**, **Center = “10,10”, RadiusX = “10”** и **RadiusY = “10”.**

11. В кода на **XAML** създайте елемент от клас **Storyboard**, като го присъедините към ресурсите на елемента **Canvas**. За обекта **Storyboard** установете свойство **x:Name = “mySecondStoryboard”.**

11. В кода на **XAML** в елемента **Storyboard** създайте *анимационен обект* **PointAnimation** (тъй като анимираното свойство **Center** на обекта **EllipseGeometry** е от тип **Point**, *анимационният обект* трябва да е от тип **PointAnimation)**. За анимационния обект установете неговите свойства, както следва:

- Анимиран обект **Storyboard.TargetName="animatedEllipse"**
- Анимирано свойство на анимирания обект **Storyboard.TargetProperty="Center"**
- Начална стойност на анимираното свойство **From="10,10"**
- Крайна стойност на анимираното свойство **To="170,170"**
- Продължителност на анимацията 5 секунди **Duration="0:0:5"**
- Повторение на анимацията **RepeatBehavior="Forever"**

Актуализираният код на XAML изглежда по следния начин (подчертан е новият добавен код от задача 2):

```
<!--ContentPanel - place additional content here-->
        <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
            <Grid.Resources>
                <Storyboard x:Name="myFirstStoryboard">
                    <DoubleAnimation Storyboard.TargetName="rectangle1"
Storyboard.TargetProperty="Width" From="20" To="200" Duration="0:0:5"
AutoReverse="True" RepeatBehavior="Forever"/>
                </Storyboard>
            </Grid.Resources>
            <Image ManipulationDelta="image1_ManipulationDelta"  Source="Tulips.jpg"
Height="154" HorizontalAlignment="Left" Margin="50,29,0,0" Name="image1"
Stretch="Fill" VerticalAlignment="Top" Width="167" />
            <Button Content="StartRotate" Height="72" HorizontalAlignment="Left"
Margin="250,29,0,0" Name="startRotate" VerticalAlignment="Top" Width="188"
Click="startRotate_Click" />
```

```
            <Button Content="StopRotate" Height="72" HorizontalAlignment="Left"
Margin="250,111,0,0" Name="stopRotate" VerticalAlignment="Top" Width="188"
Click="stopRotate_Click" />
            <Rectangle Fill="Yellow" MouseLeftButtonUp="rectangle1_MouseLeftButtonUp"
HorizontalAlignment="Left" Margin="50,237,0,0" Name="rectangle1" Stroke="Black"
StrokeThickness="1" Width="153" Height="74" VerticalAlignment="Top" />
            <Canvas Background="White" Height="180" HorizontalAlignment="Left"
Margin="27,375,0,0" Name="canvas1" VerticalAlignment="Top" Width="180">
                <Path Fill="Red">
                    <Path.Data>
                        <EllipseGeometry x:Name="animatedEllipse" Center="10,10"
RadiusX="10" RadiusY="10" />
                    </Path.Data>
                </Path>
                <Canvas.Resources>
                    <Storyboard x:Name="mySecondStoryboard">
                        <PointAnimation Storyboard.TargetName="animatedEllipse"
Storyboard.TargetProperty="Center" From="10,10" To="170, 170" Duration="0:0:5"
RepeatBehavior="Forever" />
                    </Storyboard>
                </Canvas.Resources>
            </Canvas>
            <Button Content="StartMove" Height="72" HorizontalAlignment="Left"
Margin="278,375,0,0" Name="startMove" VerticalAlignment="Top" Width="172"
Click="startMove_Click"/>
            <Button Content="StopMove" Height="72" HorizontalAlignment="Left"
Margin="281,479,0,0" Name="stopMove" VerticalAlignment="Top" Width="169"
Click="stopMove_Click"/>
        </Grid>
    </Grid>
```

12. В кода на метода за обслужване на събитието **Click** на бутона **StartMove** извикайте метод **Begin**() за обекта **mySecondStoryboard** от клас **Storyboard**. Кодът на метода би трябвало да изглежда по следния начин:

```
private void startMove_Click(object sender, RoutedEventArgs e)
{
      mySecondStoryboard.Begin();
}
```

13. В кода на метода за обслужване на събитието **Click** на бутона **StopMove** извикайте метод **Stop()** за обекта **mySecondStoryboard** от клас **Storyboard**.

```
private void stopMove_Click(object sender, RoutedEventArgs e)
{
      mySecondStoryboard.Stop();
}
```

14. Построете и изпълнете приложението.