Упражнение 4

Създаване на **Windows Phone** приложение, илюстриращо възможности за работа с изображения и за създаване на векторна графика.

Целта на приложението е да илюстрира:

- възможността за изрязване (crop) на изображение, показано в контрол **Image** и за налагане на карта маска (opacity mask) с цел управление на прозрачността на изображението;
- възможността за използване на изображение за създаване на обект от клас **ImageBrush** и неговото използване за установяване на свойство **Foreground** на **TextBlock** контрол с цел текстуриране на текст;
- възможността за създаване на векторна графика чрез изчертаване на графични примитиви на екрана;
- използването на различните видове обекти-четка за запълване на графичен примитив – правоъгълник;
- възможността за създаване на векторна графика чрез рисуване с *четка*.

За създаване на векторна графика – изчертаване на графични примитиви на екрана, Silverlight за Windows Phone осигурява базов клас **Brush**. По-конкретно, използват се обекти не от базовия клас **Brush**, а от класовете-наследници: **SolidColorBrush**, **LinearGradientBrush**, **RadialGradientBrush** и **ImageBrush**. Обектите от тези класове представляват *четки*, използвани за запълване на графични примитиви, със следните характеристики:

- Четка в един избран цвят (клас **SolidColorBrush**);
- Четка с преливаща се палитра от цветове (клас **LinearGradientBrush** и клас **RadialGradientBrush**);
- Четка, рисуваща с изображение (клас **ImageBrush**).

Задаване на цвят в **Windows Phone** приложение – три начина.
Първият начин е най-лесен и се състои в използване на предварително дефинирани имена на цветове. XAML преобразува избраното име на цвят (Red, Green, Yellow и т.н.) в структура **Color** с точно установени стойности на ARGB. В код на XAML задаването на запълване на област с избран цвят изглежда по следния начин (правоъгълник запълнен с червен цвят):

```
<Rectangle Width="100" Height="100" Fill="Red" />
```

На езика C# посочването на цвят за едноцветна четка чрез предварително дефинирани имена на цветове (изброени в **enum Colors**) би изглеждало по следния начин:

```
SolidColorBrush brush = new SolodColorBrush(Colors.Red);
```

Третият начин е като се използват шестнадесетични стойности за цвета. Структурата на шестнадесетичната стойност за цвят включва:

- две шестнадесетични цифри за alpha канал (opacity),
- две шестнадесетични цифри за канал red
- две шестнадесетични цифри за канал green
- две шестнадесетични цифри за канал blue.

Например червен цвят като шестнадесетична стойност е #FFFF0000, където: alpha=FF, red=FF, green=00, blue=00. На XAML посочването на червен цвят за запълване на правоъгълник чрез шестнадесетична стойност изглежда по следния начин:
<Rectangle Width="100" Height="100" Fill="#FFFF0000" />

В **Windows Phone** приложения могат да бъдат изчертавани *форми* (2D графични примитиви, **Shape**) като: линия (**Line**), полилиния (**Polyline**), елипса (**Ellipse**), правоъгълник (**Rectangle**), полигон (**Polygon**) и път (**Path**).
Обектите от всички изброени по-горе класове имат свойство **Stroke** и свойство **StrokeThickness**. Свойство **Stroke** трябва да се инициализира с обект-четка, за да се очертае контура на формата. Свойство **StrokeThickness** задава дебелината на контура в пиксели.

Задача 1. Да се създаде **Windows Phone** приложение за обработка на изображение, показано в контрол **Image**. Приложението да използва свойство **Clip** на контрола **Image** за изрязване на изображението с контур на елипса (използвайте обект от клас **EllipseGeometry**). За управление на прозрачността на изображението да се използва свойство **OpacityMask** (за карта-маска да се използва обект от клас **RadialGradientBrush)**.

За да създадете приложението следвайте описаната по-долу процедура.

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

2. В режим на проектиране (**MainPage.xaml**) към съществуващия контрол **Grid**, с подразбиращо се име **ContentPanel**, добавете чрез влачене от **ToolBox** един контрол **Image** и два контрола **Button**. В XAML кода за контрола **Image** установете свойство **Source** да сочи файл с изображение, вече присъединено към XAP файла на приложението (за конкретния пример файлът Tulips.jpg)**:**
<Image Source="Tulips.jpg" />
Забележка. За присъединяване на изображение към XAP файла на приложението използвайте процедурата, описана в Упражнение 1.
За контрола **Image** създайте метод за обработка на събитието **ManipulationStarted.**

Фиг. 1. Общ вид на прозореца на приложението за задачи 1, 2, 3 и 4

За първия от бутоните установете атрибут **Content = “Mask”**, атрибут **Name = “mask”** и създайте метод за обработка на събитието **Click**.
За втория от бутоните установете атрибут **Content = “Reset”,** атрибут **Name = “reset”** и създайте метод за обработка на събитието **Click** .
Примерният вид на XAML кода на приложението към момента изглежда по следния начин (подчертани са резултатите от извършените допълнителни действия):

```
 <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
      <Image Height="116" ManipulationStarted="image1_ManipulationStarted"
HorizontalAlignment="Left" Margin="21,21,0,0" Name="image1" Stretch="Fill"
VerticalAlignment="Top" Width="186" Source="Tulips.jpg"/>
     <Button Content="Mask" Height="80" HorizontalAlignment="Left"
Margin="160,198,0,0" Name="mask" VerticalAlignment="Top" Width="137"
Click="mask_Click" />
     <Button Content="Reset" Height="81" HorizontalAlignment="Left"
Margin="303,197,0,0" Name="reset" VerticalAlignment="Top" Width="140"
Click="reset_Click" />
</Grid>
```

3. Напишете кода на C# на метода за обработка на събитието **ManipulationStarted** на контрола **Image** за иницииране на изрязване на изображението, показано в него при докосване на екрана на мобилния телефон в точка, попадаща вътре в изображението. За целта е необходимо да се установи свойство **Clip** на контрола **Image,** така че да указва геометрична форма (например елипса, правоъгълник, линия, път), която ще се използва

за изрязването. В конкретния случай използвайте обект от клас **EllipseGeometry**. В резултат, от показаното в контрол **Image** изображение се изрязва само частта, която попада в геометричната форма елипса. Създайте обект от клас **EllipseGeometry** и установете свойства **RadiusX**, **RadiusY** и **Center**. За стойност на свойство **Center** задайте обект от клас **Point** с параметри на конструктора на класа **Point** – координатите на точката, в която е докоснат екрана на мобилния телефон (използвайте свойство **ManipulationOrigin** на параметъра **ManipulationStartedEventArgs** на метода за обработка на събитието **ManipulationStarted**). За стойности на свойствата **RadiusX** и **RadiusY** задайте стойност една трета от свойства **Width** и **Height** на контрола **Image**, съответно. Методът за обработка на събитието **ManipulationStarted** на контрола **Image** трябва да изглежда по следния начин:

```
private void image1_ManipulationStarted(object sender, ManipulationStartedEventArgs e)
{
        EllipseGeometry elip = new EllipseGeometry();
        elip.Center = new Point(e.ManipulationOrigin.X, e.ManipulationOrigin.Y);
        elip.RadiusX = image1.Width / 3;
        elip.RadiusY = image1.Height / 3;
        image1.Clip = elip;
        e.Complete();
        e.Handled = true;
}
```

4. Напишете кода на C# на метода за обработка на събитието **Click** на бутона **Mask.**
Бутонът с название **Mask** се използва за управление на прозрачността на изображението, вече показано в контрола **Image**. За целта е необходимо да се установи свойство **OpacityMask** на контрола **Image** да сочи карта-маска. В конкретния пример се използва карта *radial gradient* за да се постигне ефект на избледняване на изображението по краищата. За създаването на карта-маска се използва обект от клас **RadialGradientBrush**. За обекта от клас **RadialGradientBrush** установете неговите свойства **GradientOrigin** (стойности 0.5,0.5 означават средата на изображението), **Center** (стойности 0.5,0.5 означават средата на изображението), **RadiusX** (стойност 0.1 е пълна непрозрачност в центъра) и **RadiusY** (стойност 0.5 е отслабване до пълно изчезване на изображението близо до горната и долната граница на контрол **Image**). Трябва също да се създадат начален и краен цвят, като и двата са обекти от клас **GradientStop**. Тъй като в случая става въпрос за черно-бяла карта-маска, то от значение е само алфа компонентата на началния и крайния цвят - ARGB). Методът за обработка на събитието **Click** за бутона **Mask** трябва да изглежда по следния начин:

```
private void mask_Click(object sender, RoutedEventArgs e)
{
            RadialGradientBrush brush = new RadialGradientBrush();
            GradientStop gr1 = new GradientStop();
            gr1.Color = Colors.White;
            gr1.Offset = 0.1;
            GradientStop gr2 = new GradientStop();
            gr2.Color = Colors.Transparent;
            gr2.Offset = 0.9;
            brush.GradientStops.Add(gr1);
            brush.GradientStops.Add(gr2);
            brush.Center = new Point(0.5, 0.5);
            brush.GradientOrigin = new Point(0.5, 0.5);
            brush.RadiusX = 0.5;
            brush.RadiusY = 0.5;
            image1.OpacityMask = brush;
```

```
}
```

Забележка. Вместо в код написан на езика C#, свойство **Clip** на контрол Image може да се установи чрез промяна в XAML кода на контрола **Image**, както следва:

```
<Image Height="116" HorizontalAlignment="Left" Margin="124,54,0,0" Name="image1"
Stretch="Fill" VerticalAlignment="Top" Width="186" Source="Tulips.jpg">
    <Image.OpacityMask>
       <RadialGradientBrush Center="0.5,0.5" GradientOrigin="0.5, 0.5" RadiusX="0.5"
RadiusY="0.8">
           <GradientStop Color="#ffffffff" Offset="0.1"/>
           <GradientStop Color="#00ffffff" Offset="0.9"/>
       </RadialGradientBrush>
    </Image.OpacityMask>
</Image>
```

5. Напишете кода на C# на метода за обработка на събитието **Click** на бутона **Reset.** Бутонът с название **Reset** се използва за възстановяване на изображението, което е било или изрязано или обработено с карта-маска. За целта в кода за обработка на събитието **Click** за бутона **Reset** установете в **null** свойство **Clip** и свойство **OpacityMask** на контрол **Image**. Методът за обработка на събитието **Click** за бутона **Reset** трябва да изглежда по следния начин:

```
private void reset_Click(object sender, RoutedEventArgs e)
{
              image1.Clip = null;
              image1.OpacityMask = null;
}
```

6. Построете и изпълнете приложението.

Задача 2. Към **MainPage.xaml** от **Windows Phone** приложението от Задача 1 добавете контрол **TextBlock**, с изведен в него примерен текст. Създайте обект от клас **ImageBrush** и го използвайте за текстуриране на текста чрез установяване на атрибут **Foreground** на контрола **TextBlock**.

Към приложението от Задача 1 в режим на проектиране добавете контрол **TextBlock** като го привлачите от прозореца **ToolBox.** За контрола **TextBlock** установете свойство **Text = “MY TEXT”** и добавете свойства: **FontFamily = “Arial”, FontSize = “48”** и **FontWeight = “Bold”**. Установете също свойство **Foreground** на контрола **TextBlock** като създадете обект **ImageBrush**, за който установите свойството **ImageSource = “Tulips.jpg**”.

Примерният вид на XAML кода на приложението към момента изглежда по следния начин (подчертани са резултатите от извършените в рамките на Задача 2 действия):

```
<Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
      <Image Height="116" ManipulationStarted="image1_ManipulationStarted"
HorizontalAlignment="Left" Margin="21,21,0,0" Name="image1" Stretch="Fill"
VerticalAlignment="Top" Width="186" Source="Tulips.jpg"/>
     <Button Content="Mask" Height="80" HorizontalAlignment="Left"
Margin="160,143,0,0" Name="mask" VerticalAlignment="Top" Width="137"
Click="mask_Click" />
     <Button Content="Reset" Height="81" HorizontalAlignment="Left"
Margin="310,142,0,0" Name="reset" VerticalAlignment="Top" Width="140"
Click="reset_Click" />
```

```
    <TextBlock Height="116" HorizontalAlignment="Right" Margin="213,21,0,0"
Name="textBlock1" Text="MY TEXT" VerticalAlignment="Top" Width="216"
FontFamily="Arial" FontSize="48" FontWeight="Bold" >
        <TextBlock.Foreground>
            <ImageBrush ImageSource="Tulips.jpg"/>
        </TextBlock.Foreground>
    </TextBlock>
</Grid>
```

Задача 3. Към **MainPage.xaml** от **Windows Phone** приложението от Задачи 1 и 2 добавете контрол **Rectangle** и три контрола **Button**. Използвайте контролите **Button** за запълване на графичния примитив **Rectangle** с различни четки: **SolidColorBrush, LinearGradientBrush, RadialGradientBrush.**

1. За първия от новите добавени бутони установете атрибут **Content = “Solid”,** атрибут **Name = “solid”** и създайте метод за обработка на събитието **Click**. За втория от новите добавени бутони установете атрибут **Content = “Linear”,** атрибут **Name = “linear”** и създайте метод за обработка на събитието **Click**. За третия от новите добавени бутони установете атрибут **Content = “Radial”,** атрибут **Name = “radial”** и създайте метод за обработка на събитието **Click** .

Примерният вид на XAML кода на приложението към момента изглежда по следния начин (подчертани са резултатите от извършените в рамките на задача 3 действия):

```
<Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
    <Image Height="116" ManipulationStarted="image1_ManipulationStarted"
HorizontalAlignment="Left" Margin="21,21,0,0" Name="image1" Stretch="Fill"
VerticalAlignment="Top" Width="186" Source="Tulips.jpg"/>
    <Button Content="Mask" Height="80" HorizontalAlignment="Left" Margin="160,143,0,0"
Name="mask" VerticalAlignment="Top" Width="137" Click="mask_Click" />
    <Button Content="Reset" Height="81" HorizontalAlignment="Left"
Margin="310,142,0,0" Name="reset" VerticalAlignment="Top" Width="140"
Click="reset_Click" />
    <TextBlock Height="116" HorizontalAlignment="Right" Margin="213,21,0,0"
Name="textBlock1" Text="MY TEXT" VerticalAlignment="Top" Width="216"
FontFamily="Arial" FontSize="48" FontWeight="Bold" >
        <TextBlock.Foreground>
            <ImageBrush ImageSource="Tulips.jpg"/>
        </TextBlock.Foreground>
    </TextBlock>
    <Rectangle Height="105" HorizontalAlignment="Left" Margin="106,242,0,0"
Name="rectangle1" Stroke="Black" StrokeThickness="1" VerticalAlignment="Top"
Width="249" />
    <Button Content="Solid" Height="72" HorizontalAlignment="Left" Margin="12,370,0,0"
Name="solid" VerticalAlignment="Top" Width="142" Click="solid_Click" />
    <Button Content="Linear" Height="72" HorizontalAlignment="Left"
Margin="151,370,0,0" Name="linear" VerticalAlignment="Top" Width="160"
Click="linear_Click" />
    <Button Content="Radial" Height="71" HorizontalAlignment="Left"
Margin="310,371,0,0" Name="radial" VerticalAlignment="Top" Width="132"
Click="radial_Click" />
</Grid>
```

2. В тялото на метода за обработка на събитието **Click** за бутон **Solid** разположете оператори на C# за създаване на обект от клас **SolidColorBrush** и за използване на този

обект за установяване на свойство **Fill** на контрола **Rectangle**. Кодът на метода трябва да изглежда по следния начин:

```
private void solid_Click(object sender, RoutedEventArgs e)
{
      SolidColorBrush brush = new SolidColorBrush(Colors.Red);
      rectangle1.Fill = brush;
}
```

3. В тялото на метода за обработка на събитието **Click** за бутон **Linear** разположете оператори на C# за създаване на обект от клас **LinearGradientBrush,** за установяване на свойствата на обекта (преливаща между черно и бяло четка) и за използване на обекта за установяване на свойство **Fill** на контрола **Rectangle**. Кодът на метода трябва да изглежда по следния начин:

```
private void linear_Click(object sender, RoutedEventArgs e)
{
      LinearGradientBrush brush = new LinearGradientBrush();
      GradientStop gr1 = new GradientStop();
      gr1.Color = Colors.Black;
      gr1.Offset = 0.0;
      GradientStop gr2 = new GradientStop();
      gr2.Color = Colors.White;
      gr2.Offset = 1.0;
      brush.GradientStops.Add(gr1);
      brush.GradientStops.Add(gr2);
      brush.StartPoint = new Point(0, 0);
      brush.EndPoint = new Point(1, 1);
      rectangle1.Fill = brush;
}
```

4. В тялото на метода за обработка на събитието **Click** за бутон **Radial** разположете оператори на C# за създаване на обект от клас **RadialGradientBrush,** за установяване на свойствата на обекта и за използване на обекта за установяване на свойство **Fill** на контрола **Rectangle**. Кодът на метода трябва да изглежда по следния начин:

```
private void radial_Click(object sender, RoutedEventArgs e)
{
            RadialGradientBrush brush = new RadialGradientBrush();
            GradientStop gr1 = new GradientStop();
            gr1.Color = Colors.Red;
            gr1.Offset = 0.1;
            GradientStop gr2 = new GradientStop();
            gr2.Color = Colors.Yellow;
            gr2.Offset = 0.9;
            brush.GradientStops.Add(gr1);
            brush.GradientStops.Add(gr2);
            brush.Center = new Point(0.5, 0.5);
            brush.GradientOrigin = new Point(0.5, 0.5);
            brush.RadiusX = 0.5;
            brush.RadiusY = 0.5;
            rectangle1.Fill = brush;
}
```

5. Построете и изпълнете приложението.

Задача 4. Към **MainPage.xaml** от **Windows Phone** приложението от Задачи 1, 2 и 3 добавете още един контрол **Grid,** в който да изчертавате затворен многоъгълник (обект

от клас **Polygon**) чрез докосвания на екрана на мобилния телефон в границите на контрола **Grid.** Полигонът е графичен примитив, дефиниран от определен брой точки (върхове), свързани с прави линии (ребра, сегменти), като последният връх е свързан с първия (затворен контур). Тъй като полигонът е затворен контур, той може да се запълни с цвят, като за целта трябва неговото свойство **Fill** да се инициализира с избрана *четка*. За задаване на върховете да се използва свойство **Points** на обекта от клас **Polygon**, което сочи към колекция от обекти от клас **Point**.

1. Към **MainPage.xaml** добавете като го привлачите от прозорец **ToolBox,** един нов контрол **Grid.** За контрола **Grid** установете свойство **Background=”White”** и създайте метод за обработка на събитието **ManipulationStarted**.

2. В контрола **Grid** директно в кода на XAML добавете контрол **Polygon.** За контрола **Polygon** установете атрибут **Fill = “Red”** и атрибут **x:Name=”p”.** Установяването на свойство **x:Name** е необходимо за да може да се обръщате от код на **C#** към обекта от клас **Polygon.**

3. В метода за обработка на събитието **ManipulationStarted** за контрола **Grid** създайте нов обект от клас **Point,** като използвате координатите, в които е докоснат екрана на мобилния телефон и добавете новата точка към колекцията от точки **Points** на обекта от клас **Polygon**. Кодът на метода трябва да изглежда по следния начин:

```
private void grid1_ManipulationStarted(object sender, ManipulationStartedEventArgs e)
{
    Point pt = new Point(e.ManipulationOrigin.X, e.ManipulationOrigin.Y);
    p.Points.Add(pt);
}
```

Примерният вид на XAML кода на приложението към момента изглежда по следния начин (подчертани са резултатите от извършените в рамките на задача 4 действия):

```
<Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
            <Image Height="116" ManipulationStarted="image1_ManipulationStarted"
HorizontalAlignment="Left" Margin="21,21,0,0" Name="image1" Stretch="Fill"
VerticalAlignment="Top" Width="186" Source="Tulips.jpg"/>
            <Button Content="Mask" Height="80" HorizontalAlignment="Left"
Margin="21,143,0,0" Name="mask" VerticalAlignment="Top" Width="137" Click="mask_Click"
/>
            <Button Content="Reset" Height="81" HorizontalAlignment="Left"
Margin="171,143,0,0" Name="reset" VerticalAlignment="Top" Width="140"
Click="reset_Click" />
            <TextBlock Height="116" HorizontalAlignment="Right" Margin="213,21,0,0"
Name="textBlock1" Text="MY TEXT" VerticalAlignment="Top" Width="216"
FontFamily="Arial" FontSize="48" FontWeight="Bold" >
                <TextBlock.Foreground>
                    <ImageBrush ImageSource="Tulips.jpg"/>
                </TextBlock.Foreground>
            </TextBlock>
            <Rectangle Height="105" HorizontalAlignment="Left" Margin="106,242,0,0"
Name="rectangle1" Stroke="Black" StrokeThickness="1" VerticalAlignment="Top"
Width="249" />
            <Button Content="Solid" Height="72" HorizontalAlignment="Left"
Margin="12,370,0,0" Name="solid" VerticalAlignment="Top" Width="142"
Click="solid_Click" />
```

```
            <Button Content="Linear" Height="72" HorizontalAlignment="Left"
Margin="151,370,0,0" Name="linear" VerticalAlignment="Top" Width="160"
Click="linear_Click" />
            <Button Content="Radial" Height="71" HorizontalAlignment="Left"
Margin="310,371,0,0" Name="radial" VerticalAlignment="Top" Width="132"
Click="radial_Click" />
            <Grid Background="White" Height="100"
ManipulationStarted="grid1_ManipulationStarted" HorizontalAlignment="Left"
Margin="21,473,0,0" Name="grid1" VerticalAlignment="Top" Width="400" >
                <Polygon Fill="Red" x:Name="p" />
            </Grid>
        </Grid>
```

4. Построете и изпълнете приложението.