FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Media Programming on mobile devices (Windows Phone)

Bachelorarbeit / 188.939

zur Erlangung des akademischen Grades

Bachelor of Science

im Rahmen des Studiums

Medieninformatik und Visual Computing / 033 532

eingereicht von

Markus Besau
Matrikelnummer 0425311

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuer/in: Ao.Univ.Prof.Mag.Dr Horst Eidenberger

Wien, 30.09.1013 _____ _____

(Unterschrift Verfasser/in) (Unterschrift Betreuer/in)

Table of Contents

1 Project Description

Purpose of this project was to provide specific easy to use and understand examples which should ease the first steps in Windows Phone application development.

A list of Android-based examples has been ported and adapted to the Windows Phone platform and additional ones, which show distinctive Windows Phone features, have been added.

Although all kinds of features are being demonstrated, the focus lies on Media Programming. Therefore a lot of Media Programming applications are found in the examples.

2 Time Table

Date	from	until	delta	Activity
04.01.2013	16:00	19:00	3:00	neu aufsetzen des Laptops, Installation VS 2012 mit SP 1, Windows Phone 8 SDK und Konfiguration
05.01.2013	10:30	14:45	4:15	TFS Projekt erstellt und in die IDE eingebunden, Installation Eclipse und Android SDK
12.01.2013	11:30	15:00	3:30	Einlesen in den Android Code
13.01.2013	13:00	15:00	2:00	Anlegen des ersten Android Demo Controls
13.01.2013	15:00	20:00	5:00	Install & Config Android Emulator & Phone
13.01.2013	21:00	23:30	2:30	Canvas-Based Android Application
14.01.2013	18:30	20:00	1:30	Generic Connection Framework Android
15.01.2013	18:45	20:30	1:45	Recherche Generic Connection Framework WP
16.01.2013	18:00	20:00	2:00	Generic Connection Framework Windows Phone
03.02.2013	14:00	15:00	1:00	XAMLBased UI
03.02.2013	15:00	17:00	2:00	Eventbased Interaction WP
18.03.2013	18:15	21:00	2:45	Personal Information Management Android
19.03.2013	19:00	21:45	2:45	Recherche Personal Information Management WP
20.03.2013	18:00	20:00	2:00	Personal Information Management WP
21.03.2013	19:00	21:30	2:30	Location API Android
23.03.2013	18:00	20:45	2:45	Recherche Location API WP
24.03.2013	19:45	22:00	2:15	Location API WP
26.03.2013	18:00	19:15	1:15	SMS Android
27.03.2013	18:45	21:30	2:45	Recherche SMS WP

28.03.2013	18:15	22:00	3:45	SMS WP
02.04.2013	18:00	21:30	3:30	MMS Android
03.04.2013	18:15	22:00	3:45	Recherche MMS WP
04.04.2013	19:00	21:15	2:15	Recherche MMS WP (leider nicht möglich)
05.07.2013	15:00	18:00	3:00	Accelerator Sensor Android
06.07.2013	09:30	13:00	3:30	Recherche Accelerator Sensor WP
06.07.2013	15:00	18:00	3:00	Recherche Accelerator Sensor WP
07.07.2013	18:30	22:00	3:30	Accelerator Sensor WP
08.07.2013	18:00	20:00	2:00	Accelerator Sensor WP
15.07.2013	08:00	11:00	3:00	MediaPlayback Android
15.07.2013	14:00	17:00	3:00	Recherche MediaPlayback WP
16.07.2013	09:00	12:00	3:00	MediaPlayback WP
16.07.2013	15:00	17:45	2:45	VideoRecorder Android
17.07.2013	09:00	12:30	3:30	Recherche VideoRecorder WP
17.07.2013	17:00	18:45	1:45	Recherche VideoRecorder WP
18.07.2013	07:30	12:00	4:30	VideoRecorder WP
18.07.2013	13:45	16:00	2:15	AudioRecorder Android
18.07.2013	18:00	20:15	2:15	Snapshot Android
19.07.2013	08:45	11:15	2:30	Recherche AudioRecorder WP
19.07.2013	18:00	20:00	2:00	AudioRecorder WP
19.07.2013	21:00	23:30	2:30	AudioRecorder WP v2
20.07.2013	08:15	11:00	2:45	Recherche LiveTile WP
20.07.2013	11:00	13:00	2:00	Recherche Snapshot WP
20.07.2013	15:00	18:00	3:00	LiveTile WP
20.07.2013	18:00	19:00	1:00	Snapshot WP
20.07.2013	19:30	22:15	2:45	Application Menu WP
21.07.2013	07:15	13:00	5:45	Process Image WP
21.07.2013	13:00	15:00	2:00	Touch Input WP
21.07.2013	16:00	21:00	5:00	Bluetooth WP
22.07.2013	08:00	11:15	3:15	Process Image Android
22.07.2013	13:30	15:45	2:15	Bluetooth Android
29.07.2013	19:00	21:30	2:30	Databinding WP

30.07.2013	08:00	15:00	7:00	clean up code
01.08.2013	09:00	17:00	8:00	create documentation
02.08.2013	18:45	21:30	2:45	documentation
03.08.2013	07:15	11:00	3:45	3D graphics Android
03.08.2013	13:00	17:00	4:00	Recherche 3D Silverlight/XNA WP
04.08.2013	13:00	22:00	9:00	3D Graphics XNA WP
25.09.2013	12:00	14:00	2:00	3D Graphics XNA WP
25.09.2013	16:00	19:00	3:00	rewrite Bluetooth WP
26.09.2013	09:30	19:00	9:30	clean up documentation
27.09.2013	16:00	18:30	2:30	clean up documentation
		Total:	**192:30**	

3 General Information

Visual Studio Project Templates

All projects were created using the default Windows Phone Visual Studio project templates. Project settings have not been tempered with unless stated otherwise. Most of the added references, classes and other resources (e.g. images) are therefore been added by the template.

Windows Phone Developer License

A developer license is needed to publish a Windows Phone application on the Marketplace[1]. Costs for the license are 100€ per year. You can also get the license for free if you have access to DreamSpark for Academic Institutions.

Testing Applications on a device

Only 3 applications can be installed simultaneously on a developer unlocked windows phone device.

To publish an application onto the device, the device must be connected to the PC running Visual Studio, the device's screen must be unlocked and if running Windows Phone 7.8 or lower, Zune must be installed and running on the PC.

1 http://create.msdn.com/

4 Software

All used applications and additional libraries are listed below. Each application is free to use or offers a free alternative.

Visual Studio Professional 2012

Visual Studio is necessary to develop all kind of .NET applications. Since 2 examples run only on Windows Phone 8, it is advised to use a Visual Studio 2012 or later version, because the Windows Phone 8 SDK is known to be a little difficult to fully configure with Visual Studio 2010.

The free version Visual Studio Express 2012[2] is sufficient for these applications. Some schools and university (like TU Wien[3]) offer access to Microsoft DreamSpark for Academic Institutions[4] (former known as MSDN AA) which offers students all kind of software for free. (e.g. Visual Studio Professional 2012)

Team Foundation Server 2012

A Team Foundation Server 2012 was used as repository. A free to use light version can be found on http://tfs.visualstudio.com/

Windows Phone 8 SDK

The Windows Phone SDK[5] must be installed alongside with Visual Studio to be able to develop Windows Phone applications. It comes with all necessary APIs, Visual Studio project templates and built-in emulators.

Windows Phone SDK 8 supports both Windows Phone 7.8 and lower and Windows Phone 8 applications, whereas Windows Phone SDK 7.8 only supports Windows Phone 7.8 and lower applications.

Note: Not all APIs are accessible on Windows Phone SDK 7.8 and therefore not all applications can run on Windows Phone 7.8 and lower. (e.g. the Bluetooth API has only been added in Windows Phone SDK 8)

Windows Phone Toolkit

The Windows Phone Toolkit adds additional controls to Silverlight for Windows Phone applications. (e.g. DateTimePicker, AutoCompleteBox,...) It is free to use and can be downloaded from Codeplex[6] or installed directly out of Visual Studio using the NuGet package manager.

---

2 http://www.microsoft.com/visualstudio/eng/products/visual-studio-express-products

3 https://www.informatik.tuwien.ac.at/msdnaa

4 https://www.dreamspark.com/institution/subscription.aspx

5 https://dev.windowsphone.com/en-us/downloadsdk

6 http://phone.codeplex.com/

Blend for Visual Studio 2012

Blend for Visual Studio 2012 comes with Visual Studio and helps the developer design an application by providing additional design tools. It supports most project types as Visual Studio and can therefore be used for most .NET user interfaces. (e.g. Silverlight for Windows Phone, ASP.NET, WPF,...)

The same project can be opened at the same time in both Visual Studio and Blend.

Windows Phone Developer Registration

If you successfully registered as a Windows Phone Developer on http://create.msdn.com and installed the Windows Phone SDK you can now unlock up to 3 devices for developer purposes using the Windows Phone Developer Registration tool which comes with the Windows Phone SDK.

5 Code Examples

5.1 Event-based User Interaction

This application show how to use Event-based User Interaction by adding a Handler to the *ValueChanged* event of a simple *DatePicker*. This way the application can react to any kind of events, may they be initiated by the user or the system.

In this example the new *DatePickers* value is being displayed in a separate *TextBox*.

The DatePicker is part of the Windows Phone Toolkit[7] which must be installed separately. It contains a lot of useful advanced controls.

References added:

Windows.Phone.Controls.Toolkit (used for DatePicker control)

XAML:

```
<phone:PhoneApplicationPage
    x:Class="EventBasedInteraction.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
```

7 http://phone.codeplex.com/

```
    xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
    xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d" d:DesignWidth="480" d:DesignHeight="768"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
    SupportedOrientations="Portrait" Orientation="Portrait"
    shell:SystemTray.IsVisible="True"

    xmlns:toolkit="clr-
      namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone.Controls.Toolkit">

    <!--LayoutRoot is the root grid where all page content is placed-->
    <Grid x:Name="LayoutRoot" Background="Transparent">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>

        <!--TitlePanel contains the name of the application and page title-->
        <StackPanel x:Name="TitlePanel" Grid.Row="0" Margin="12,17,0,28">
            <TextBlock x:Name="ApplicationTitle" Text="MediaUnderstanding"
                    Style="{StaticResource PhoneTextNormalStyle}"/>
                  <TextBlock x:Name="PageTitle" Text="EventBased" Margin="9,-7,0,0"
                    Style="{StaticResource PhoneTextTitle1Style}"/>
        </StackPanel>

        <!--ContentPanel - place additional content here-->
        <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
            <StackPanel Orientation="Vertical">
                <TextBlock x:Name="tbDate" Text="Date"/>
                <toolkit:DatePicker x:Name="dpDate"
                    ValueChanged="dpDate_ValueChanged" />
            </StackPanel>
        </Grid>
    </Grid>

</phone:PhoneApplicationPage>
```

Code Behind:

```
using Microsoft.Phone.Controls;

/// <summary>
/// This application shows how to use Event-Based User Interaction.
/// It uses a simple <see cref="Microsoft.Phone.Controls.DatePicker"/>
/// and adds a handler to it's ValueChanged event. That way the
/// application can react to any kind of input, may it be
/// initiated by the user or the system.
/// </summary>
/// <remarks>
/// Bachelorthesis - "Media Programming on Mobile Devices (Windows Phone)"
/// by Markus Besau
/// July, 2013
/// </remarks>
namespace EventBasedInteraction
{
    public partial class MainPage : PhoneApplicationPage
```

```
    {
        public MainPage()
        {
            InitializeComponent();
        }

        private void dpDate_ValueChanged(object sender,
             DateTimeValueChangedEventArgs e)
        {
             // the DatePickers value has changed,
             // display it additionally in the TextBox.
            tbDate.Text = e.NewDateTime.Value.Date.ToShortDateString();
        }
    }
}
```

5.2 Touch Input

This example shows how to use touch inputs in your applications. A *Rectangle* is drawn within a simple *Grid*. Depending on the coordinates of the touch input, the rectangles width and height will increase or decrease. On the left side of the rectangle its width will decrease, on the right side its width will increase, on the top side its height will decrease and on the bottom side its height will increase.

To handle touch input on non-interactive controls (e.g. a *Grid* or *Rectangle*) a handler must be added to the controls *Tap* event. The touch coordinates are accessible by calling *GetPosition(UIElement)* on the passed parameter of type *GestureEventArgs*. This method will return the coordinates relative to the passed element.

References added:
None

XAML:

```
<phone:PhoneApplicationPage
    x:Class="TouchInput.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
    xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d" d:DesignWidth="480" d:DesignHeight="768"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
```

```
    SupportedOrientations="Portrait" Orientation="Portrait"
    shell:SystemTray.IsVisible="True">

    <!--LayoutRoot is the root grid where all page content is placed-->
    <Grid x:Name="LayoutRoot" Background="Transparent">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>

        <!--TitlePanel contains the name of the application and page title-->
        <StackPanel x:Name="TitlePanel" Grid.Row="0" Margin="12,17,0,28">
            <TextBlock x:Name="ApplicationTitle" Text="MediaUnderstanding"
                    Style="{StaticResource PhoneTextNormalStyle}"/>
            <TextBlock x:Name="PageTitle" Text="TouchInput" Margin="9,-7,0,0"
                    Style="{StaticResource PhoneTextTitle1Style}"/>
        </StackPanel>

        <!--ContentPanel - place additional content here-->
        <Grid x:Name="ContentPanel" Grid.Row="1" Width="400" Height="500"
                    Tap="ContentPanel_Tap" Background="Green">
            <Rectangle x:Name="rect" Fill="Aquamarine" Height="100" Width="100"
                    VerticalAlignment="Center" HorizontalAlignment="Center"/>
        </Grid>
        <TextBlock HorizontalAlignment="Left" Margin="413,299,0,0"
             Grid.Row="1" Text="+" VerticalAlignment="Top"/>
        <TextBlock HorizontalAlignment="Left" Margin="226,527,0,0"
             Grid.Row="1" Text="+" VerticalAlignment="Top"/>
        <TextBlock HorizontalAlignment="Left" Margin="50,299,0,0"
             Grid.Row="1" Text="-" VerticalAlignment="Top"/>
        <TextBlock HorizontalAlignment="Left" Margin="232,54,0,0"
             Grid.Row="1" Text="-" VerticalAlignment="Top"/>
    </Grid>

</phone:PhoneApplicationPage>
```

Code Behind:

```
using Microsoft.Phone.Controls;
using System.Windows.Input;

namespace TouchInput
{
    /// <summary>
    /// This applications demonstrates how to get touch input
    /// coordinates and how to react to them.
    /// Depending on the coordinates it will increase or decrease
    /// the rectangles width and height.
    /// </summary>
    /// <remarks>
    /// Bachelorthesis - "Media Programming on Mobile Devices (Windows Phone)"
    /// by Markus Besau
    /// July, 2013
    /// </remarks>
    public partial class MainPage : PhoneApplicationPage
    {
        private double m_ContentWidth;
        private double m_ContentHeight;

        public MainPage()
```

```
        {
            InitializeComponent();

            // get ContentPanels current width and height
            m_ContentWidth = ContentPanel.Width;
            m_ContentHeight = ContentPanel.Height;
        }

        private void ContentPanel_Tap(object sender, GestureEventArgs e)
        {
            // get the touch coordinates
            var touch = e.GetPosition(ContentPanel);

            // get the rectangles current width and height
            var rectWidth = rect.Width;
            var rectHeight = rect.Height;

            // calculate rectangles left, top, right, bottom
            var rectTop = m_ContentHeight / 2 - rectHeight / 2;
            var rectBottom = m_ContentHeight / 2 + rectHeight / 2;
            var rectLeft = m_ContentWidth / 2 - rectWidth / 2;
            var rectRight = m_ContentWidth / 2 + rectWidth / 2;

            // resize the rectangles width
            if (touch.X < rectLeft && rect.Width > 20)
                rect.Width -= 20; // decrease width
            else if (touch.X > rectRight && rect.Width < m_ContentWidth - 40)
                rect.Width += 20; // increase width

            // resize the rectangles height
            if (touch.Y < rectTop && rect.Height > 20)
                rect.Height -= 20; // decrease height
            else if (touch.Y > rectBottom && rect.Height < m_ContentHeight - 40)
                rect.Height += 20; // increase width
        }
    }
}
```

5.3 Generic Connection Framework

This example shows how to retrieve some data from the Web by using a *HttpWebRequest*[8]. The connection is completely handled by the OS. The OS decides which channel to use based on the available options. If the device is connected to a WiFi network with internet access, the OS will run the request through the WiFi network. If the WiFi is disabled the OS will automatically choose the mobile data connection, if available. So there is no need for the developer to do that manually.

8 http://msdn.microsoft.com/en-US/library/windowsphone/develop/system.net.httpwebrequest(v=vs.105).aspx

After the button has been clicked the application will call http://www.tuwien.ac.at to get the left menue entries and list them using a *ListBox*[9]. For better visualization a *ListBox.ItemTemplate* is *used* to set some design properties. (e.g. Margin, Font, Border) The *ListBox.ItemTemplate* will be applied to every *ListBoxItem*. The *ListBox.ItemTemplate* contains a single *TextBox* whose Text-Property uses DataBinding. After setting the source to any *IEnumerable* container, in this example a *List<string>*, the Text will be automatically displayed.

References added:

None

XAML:

```
<phone:PhoneApplicationPage
    x:Class="GenericConnectionFramework.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
    xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d" d:DesignWidth="480" d:DesignHeight="768"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
    SupportedOrientations="Portrait" Orientation="Portrait"
    shell:SystemTray.IsVisible="True">

    <!--LayoutRoot is the root grid where all page content is placed-->
    <Grid x:Name="LayoutRoot" Background="Transparent">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>

        <!--TitlePanel contains the name of the application and page title-->
        <StackPanel x:Name="TitlePanel" Grid.Row="0" Margin="12,17,0,28">
            <TextBlock x:Name="ApplicationTitle" Text="MediaUnderstanding"
                    Style="{StaticResource PhoneTextNormalStyle}"/>
            <TextBlock x:Name="PageTitle" Text="Generic Connection Framework"
                    Margin="9,7,0,0" Style="{StaticResource PhoneTextTitle1Style}"/>
        </StackPanel>

        <!--ContentPanel - place additional content here-->
        <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
            <Grid.RowDefinitions>
                <RowDefinition Height="100" />
                <RowDefinition />
            </Grid.RowDefinitions>
```

9 http://msdn.microsoft.com/en-us/library/system.windows.controls.listbox(v=vs.95).aspx

```
            <Button x:Name="btMakeHttpRequest" Content="make HttpWebRequest"
             Grid.Row="0" Height="100" HorizontalAlignment="Center"
             Click="btMakeHttpRequest_Click" />
            <ListBox x:Name="lbMenue" Grid.Row="1" >
                <ListBox.ItemTemplate>
                    <DataTemplate>
                        <Border BorderBrush="White" BorderThickness="2" Margin="10"
                           Padding="5" Width="400">
                            <TextBlock Text="{Binding}" FontSize="20"/>
                        </Border>
                    </DataTemplate>
                </ListBox.ItemTemplate>
            </ListBox>
        </Grid>
    </Grid>

</phone:PhoneApplicationPage>
```

Code Behind:

```
using Microsoft.Phone.Controls;
using System;
using System.Collections.Generic;
using System.IO;
using System.Net;
using System.Windows;

namespace GenericConnectionFramework
{
    /// <summary>
    /// This application uses a <see cref="System.Net.HttpWebRequest"/> to read a
       website.
    /// The OS will handle the connection based on the available channels on the
       device automatically.
    /// There is no need to specify which channel to use.
    /// e.g. if the device is connected to a WiFi network with internet access,
    /// the OS will run the request through WiFi
    /// </summary>
    /// <remarks>
    /// Bachelorthesis - "Media Programming on Mobile Devices (Windows Phone)"
    /// by Markus Besau
    /// July, 2013
    /// </remarks>
    public partial class MainPage : PhoneApplicationPage
    {
        public MainPage()
        {
            InitializeComponent();
        }

        private void btMakeHttpRequest_Click(object sender, RoutedEventArgs e)
        {
            // create the webRequest an add the asynchronous callback
            HttpWebRequest webRequest = (HttpWebRequest)HttpWebRequest.Create(new
                    Uri("http://www.tuwien.ac.at"));
            webRequest.BeginGetResponse(new AsyncCallback(httpWebRequest_Callback),
                    webRequest);
        }
```

```
        private void httpWebRequest_Callback(IAsyncResult _callbackResult)
        {
            // get request and response
            HttpWebRequest webRequest = (HttpWebRequest)_callbackResult.AsyncState;
            HttpWebResponse webResponse =
                    (HttpWebResponse)webRequest.EndGetResponse(_callbackResult);

            // get the response stream
            using (StreamReader streamReader = new
                    StreamReader(webResponse.GetResponseStream()))
           {
                // process website, in our case get the left menu entries
                string siteContent = streamReader.ReadToEnd();
                string[] splitStrings = {"<li class=\"nav1\">"};
                string[] menue = siteContent.Split(splitStrings,
                           StringSplitOptions.RemoveEmptyEntries);
                var filteredItems = new List<string>();
                for (int i = 1; i < menue.Length - 1; i++)
               {
                    var titleStart = menue[i].IndexOf("title=\"") + 6;
                    var titleEnd = menue[i].IndexOf(">");
                   filteredItems.Add(menue[i].Substring(titleStart, titleEnd-titleStart));
               }

                // since it's an asynchronous callback, we need to use
                // a dispatcher to update the UI
               Dispatcher.BeginInvoke(() => lbMenue.ItemsSource = filteredItems);
           }

            webResponse.Close();
        }
    }
}
```

5.4 Location API

This application shows how to use a *GeoCoordinateWatcher*[10] to get the current location. The *GeoCoordinateWatcher* can use GPS, WiFi- and GSM triangulation. A threshold can be set to disable location updates based on static noise.

The *GeoCoordinateWatcher* checks if it has access to the phones location service and if it can retrieve data. If so, an event will be raised every time the location changes for more than the defined threshold.

References added:

None

10 http://msdn.microsoft.com/EN-US/library/windowsphone/develop/system.device.location.geocoordinatewatcher(v=vs.105).aspx

XAML:

```
<phone:PhoneApplicationPage
    x:Class="LocationAPI.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
    xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d" d:DesignWidth="480" d:DesignHeight="768"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
    SupportedOrientations="Portrait" Orientation="Portrait"
    shell:SystemTray.IsVisible="True">

    <!--LayoutRoot is the root grid where all page content is placed-->
    <Grid x:Name="LayoutRoot" Background="Transparent">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>

        <!--TitlePanel contains the name of the application and page title-->
        <StackPanel x:Name="TitlePanel" Grid.Row="0" Margin="12,17,0,28">
            <TextBlock x:Name="ApplicationTitle" Text="MediaUnderstanding"
             Style="{StaticResource PhoneTextNormalStyle}"/>
            <TextBlock x:Name="PageTitle" Text="Location API" Margin="9,-7,0,0"
             Style="{StaticResource PhoneTextTitle1Style}"/>
        </StackPanel>

        <!--ContentPanel - place additional content here-->
        <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
            <TextBlock HorizontalAlignment="Left" Margin="10,10,0,0"
             TextWrapping="Wrap" Text="Latitude:" VerticalAlignment="Top"/>
            <TextBox x:Name="tbLatitude" HorizontalAlignment="Left" Height="72"
             Margin="0,37,0,0" TextWrapping="Wrap" VerticalAlignment="Top"
             Width="456" IsReadOnly="True"/>
            <TextBlock HorizontalAlignment="Left" Margin="10,109,0,0"
             TextWrapping="Wrap" Text="Longitude:" VerticalAlignment="Top"/>
            <TextBox x:Name="tbLongitude" HorizontalAlignment="Left" Height="72"
             Margin="0,131,0,0" TextWrapping="Wrap" VerticalAlignment="Top"
             Width="456" IsReadOnly="True"/>
        </Grid>
    </Grid>

</phone:PhoneApplicationPage>
```

Code Behind:

```
using Microsoft.Phone.Controls;
using System;
using System.Device.Location;
using System.Windows;

namespace LocationAPI
{
    /// <summary>
```

```
/// This application uses a <see
   cref="System.Device.Location.GeoCoordinateWatcher"/>
/// to get the current location. The GeoCoordinateWatcher can use GPS,
/// WiFi- and GSM triangulation. A threshold can be set to disable
/// location updates based on static noise.
/// The GeoCoordinateWatcher checks if it has access to the
/// location service and if it can retrieve data. If so,
/// an event will be raised every time the location changes
/// for more than the defined threshold.
/// </summary>
/// <remarks>
/// Bachelorthesis - "Media Programming on Mobile Devices (Windows Phone)"
/// by Markus Besau
/// July, 2013
/// </remarks>
public partial class MainPage : PhoneApplicationPage
{
    GeoCoordinateWatcher m_GeoWatcher;

    public MainPage()
    {
        InitializeComponent();
        /*
         * Windows Phone Location API uses
         * GPS
         * WiFi-triangulation
         * GSM-triangulation
         */

        // High uses GPS too, Default only WiFi and GSM
        m_GeoWatcher = new GeoCoordinateWatcher(GeoPositionAccuracy.High);
        // set min distance to move before triggering an update
        m_GeoWatcher.MovementThreshold = 10;

        // add eventhandlers
        m_GeoWatcher.StatusChanged += new
         EventHandler<GeoPositionStatusChangedEventArgs>(status_Changed);
        m_GeoWatcher.PositionChanged += new
         EventHandler<GeoPositionChangedEventArgs<GeoCoordinate>>
         (position_Changed);

        // start the watcher
        m_GeoWatcher.Start();
    }

    private void status_Changed(object sender,
         GeoPositionStatusChangedEventArgs e)
    {
        switch (e.Status)
        {
            case GeoPositionStatus.Disabled:
                {
                    //location service disabled or unsupported

                    if (m_GeoWatcher.Permission == GeoPositionPermission.Denied)
                        MessageBox.Show("location service has been disabled on
                             this device");
                    else
                        MessageBox.Show("location service is not supported on this
                             device");
                    break;
```

```
                    }

                case GeoPositionStatus.Initializing:
                    {
                        // still initializing
                        break;
                    }
                case GeoPositionStatus.NoData:
                    {
                        // location service is working but can't get location data
                        break;
                    }
                case GeoPositionStatus.Ready:
                    {
                        // location service is ready
                        break;
                    }
            }
        }

        private void position_Changed(object sender,
             GeoPositionChangedEventArgs<GeoCoordinate> e)
        {
            tbLatitude.Text = e.Position.Location.Latitude.ToString();
            tbLongitude.Text = e.Position.Location.Longitude.ToString();
        }
    }
}
```

5.5 SMS

This example shows how to use a *SmsComposeTask*[11] to enable users to send an SMS from your application. The application uses two *TextBoxes* to get the recipients number and the message body. After reading the values, a *SmsComposeTask* is used to launch the native Messaging application with the pre-populated recipient and message. The message will be sent after the user presses the send button. If the message shall be sent to more than one recipient, the recipients numbers must be separated by a semicolon.

References added:
None

11 http://msdn.microsoft.com/en-US/library/windowsphone/develop/microsoft.phone.tasks.smscomposetask(v=vs.105).aspx

XAML:

```
<phone:PhoneApplicationPage
    x:Class="SMS.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
    xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d" d:DesignWidth="480" d:DesignHeight="768"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
    SupportedOrientations="Portrait" Orientation="Portrait"
    shell:SystemTray.IsVisible="True">

    <!--LayoutRoot is the root grid where all page content is placed-->
    <Grid x:Name="LayoutRoot" Background="Transparent">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>

        <!--TitlePanel contains the name of the application and page title-->
        <StackPanel x:Name="TitlePanel" Grid.Row="0" Margin="12,17,0,28">
            <TextBlock x:Name="ApplicationTitle" Text="MediaUnderstanding"

                    Style="{StaticResource PhoneTextNormalStyle}"/>
            <TextBlock x:Name="SMS" Text="send SMS" Margin="9,-7,0,0"
                    Style="{StaticResource PhoneTextTitle1Style}"/>
        </StackPanel>

        <!--ContentPanel - place additional content here-->
        <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
            <Grid.RowDefinitions>
                <RowDefinition Height="70"/>
                <RowDefinition Height="70"/>
                <RowDefinition Height="70"/>
                <RowDefinition Height="*"/>
            </Grid.RowDefinitions>
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="100"/>
                <ColumnDefinition/>
            </Grid.ColumnDefinitions>

            <TextBox Name="tbText" Text="yippie kai-yay!" Grid.Row="0"
             Grid.Column="1" />
            <TextBlock Text="Text:" VerticalAlignment="Center"/>

            <TextBlock Name="lbNumber" Text="Number" VerticalAlignment="Center"
                    Grid.Row="1" Grid.Column="0" />
            <TextBox Name="tbNumber" Text="0664 12 34 567"
             Grid.Row="1" Grid.Column="1"/>
            <Button Content="Send SMS" Grid.Row="2" Grid.ColumnSpan="2"
             Name="btSend" Click="btSend_Click"/>
        </Grid>
    </Grid>

</phone:PhoneApplicationPage>
```

Code Behind:

```
using Microsoft.Phone.Controls;
using Microsoft.Phone.Tasks;
using System.Windows;

namespace SMS
{
    /// <summary>
    /// This application uses a <see cref="Microsoft.Phone.Tasks.SmsComposeTask"/>
    /// to enable the user to send a SMS from your application.
    /// </summary>
    /// <remarks>
    /// "Media Programming on Mobile Devices (Windows Phone)"
    /// by Markus Besau
    /// July, 2013
    /// </remarks>
    public partial class MainPage : PhoneApplicationPage
    {
        public MainPage()
        {
            InitializeComponent();
        }

        private void btSend_Click(object sender, RoutedEventArgs e)
        {
            // get the user-submitted values
            string smsText = tbText.Text;
            string smsTo = tbNumber.Text;

            // create a new SmsComposeTask and set number & text
            SmsComposeTask smsComposeTask = new SmsComposeTask();
            smsComposeTask.To = smsTo;
            smsComposeTask.Body = smsText;

            // additional recipients can be added to the SmsComposeTask.To Property
            // smsComposeTask.To += "; 0664 111 22 33";

            // open task
            smsComposeTask.Show();
        }
    }
}
```

5.6 Bluetooth Connections

This example shows how to use the *PeerFinder*[12] to open a Bluetooth connection to another device. After getting all paired devices, the application tries to connect to the first one. If it succeeds in connecting, it opens a stream and sends a message to the other device.

Note: There are 2 different modes of Bluetooth Connections on Windows Phone: "App to App" and "App to Device". Using the *PeerFinder* both types can be used. While this example uses "App to Device" to connect to

12 http://msdn.microsoft.com/en-us/library/windows/apps/windows.networking.proximity.peerfinder

another device, "App to App" can be used to connect to an application running on the other device.

Note: ID_CAP_Proximity must be tagged in WMAppManifest.xml

Note: The Bluetooh API is only available on Windows Phone 8.

References added:

None

XAML:

```
<phone:PhoneApplicationPage
    x:Class="BluetoothConnection.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
    xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
    SupportedOrientations="Portrait" Orientation="Portrait"
    shell:SystemTray.IsVisible="True">

    <!--LayoutRoot is the root grid where all page content is placed-->
    <Grid x:Name="LayoutRoot" Background="Transparent">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>

        <!--TitlePanel contains the name of the application and page title-->
        <StackPanel x:Name="TitlePanel" Grid.Row="0" Margin="12,17,0,28">
            <TextBlock Text="MediaUnderstanding" Style="{StaticResource
                    PhoneTextNormalStyle}" Margin="12,0"/>
            <TextBlock Text="Bluetooth" Margin="9,-7,0,0" Style="{StaticResource
                    PhoneTextTitle1Style}"/>
        </StackPanel>

        <!--ContentPanel - place additional content here-->
        <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">

        </Grid>
    </Grid>

</phone:PhoneApplicationPage>
```

Code Behind:

```
using Microsoft.Phone.Controls;
using System;
using System.Collections.ObjectModel;
using System.Linq;
```

```
using System.Windows;
using Windows.Networking.Proximity;
using Windows.Networking.Sockets;
using Windows.Storage.Streams;

namespace BluetoothConnection
{
    /// <summary>
    /// This application shows how to open a Bluetooth connection
    /// to another device and send a message to it.
    /// The other device must be paired first, otherwise the
    /// application wont work.
    /// Using the <see cref="Windows.Networking.Proximity.PeerFinder"/>
    /// the application can also pair to remote apps of its own kind.
    /// </summary>
    /// <remarks>
    /// Bachelorthesis - "Media Programming on Mobile Devices (Windows Phone)"
    /// by Markus Besau
    /// July, 2013
    /// </remarks>
    public partial class MainPage : PhoneApplicationPage
    {
        public MainPage()
        {
            InitializeComponent();

            Loaded += connectBluetoothDevice;
        }

        private async void connectBluetoothDevice(object sender, RoutedEventArgs e)
        {
            try
            {
                // search for all paired devices
                PeerFinder.AlternateIdentities["Bluetooth:Paired"] = "";

                // get paired devices
                var pairedDevices = await PeerFinder.FindAllPeersAsync();

                if (pairedDevices.Count == 0)
                {
                    // no paired devices
                    MessageBox.Show("found no paired Bluetooth devices");
                    return;
                }

                // get the first paired device
                PeerInformation bluetoothDevice = pairedDevices.Last();

                // try to connect
                StreamSocket socket = new StreamSocket();
                await socket.ConnectAsync(bluetoothDevice.HostName, "1");

                // send message
                DataWriter writer = new DataWriter(socket.OutputStream);
                writer.WriteString("media understanding");
                await writer.FlushAsync();
            }
            catch (Exception ex)
            {
                if ((uint)ex.HResult == 0x8004048F)
```

```
                {
                    // errorcode for deactivated bluetooth
                    MessageBox.Show("Bluetooth is turned off!");
                }
            }
        }

    }
}
```

5.7 3D Graphics Output

This application shows how to use the XNA framework to render 3D output on a Windows Phone device. To keep it simple the application only renders a rotating triangle.

The rotations angle will be incremented in each iteration of the game-loop to be able to apply the world matrix in the Draw method.

Vertices are being created using *Vector3*[13] and *VertexPositionColor*[14] objects to store their color. The *GraphicsDevice*[15] is being used to create the viewport and the projection-matrix. All matrices are being stored in *Matrix*[16] objects which provide functions for some basic matrix calculations.

References added:

None

XAML:

None, for using XNA instead of Silverlight

Code Behind:

```
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;

namespace _3DGraphicsXNA
{
    /// <summary>
    /// This application shows how to use XNA for 3D graphics output
    /// by rendering a simple rotating triangle.
```

13 http://msdn.microsoft.com/en-us/library/microsoft.xna.framework.vector3.aspx

14 http://msdn.microsoft.com/en-us/library/Microsoft.Xna.Framework.Graphics.VertexPositionColor.aspx

15 http://msdn.microsoft.com/en-us/library/microsoft.xna.framework.graphics.graphicsdevice.aspx

16 http://msdn.microsoft.com/en-us/library/microsoft.xna.framework.matrix.aspx

```
    /// In each game-loop an angle is incremented which will be used
    /// to rotate the triangle on the Draw method.
    /// </summary>
    /// <remarks>
    /// Bachelorthesis - "Media Programming on Mobile Devices (Windows Phone)"
    /// by Markus Besau
    /// July, 2013
    /// </remarks>
    public class Game1 : Microsoft.Xna.Framework.Game
    {
        GraphicsDeviceManager m_Graphics;
        SpriteBatch m_SpriteBatch;
        GraphicsDevice m_Device;
        private float m_Angle = 0f;
        BasicEffect m_Effect;
        VertexPositionColor[] m_Vertices;
        Matrix m_ViewMatrix;
        Matrix m_ProjectionMatrix;

        public Game1()
        {
            m_Graphics = new GraphicsDeviceManager(this);
            Content.RootDirectory = "Content";
        }

        /// <summary>
        /// Allows the game to perform any initialization it needs to before starting
             to run.
        /// This is where it can query for any required services and load any non-
             graphic
        /// related content.  Calling base. Initialize will enumerate through any
             components
        /// and initialize them as well.
        /// </summary>
        protected override void Initialize()
        {
            m_Graphics.PreferredBackBufferWidth = 300;
            m_Graphics.PreferredBackBufferHeight = 500;
            m_Graphics.IsFullScreen = false;
            m_Graphics.ApplyChanges();

            base.Initialize();
        }

        /// <summary>
        /// LoadContent will be called once per game and is the place to load
        /// all of your content.
        /// </summary>
        protected override void LoadContent()
        {
            m_SpriteBatch = new SpriteBatch(GraphicsDevice);
            m_Device = m_Graphics.GraphicsDevice;

            m_Effect = new BasicEffect(m_Graphics.GraphicsDevice);
            setUpCamera();
            setupEffect();
            setupVertices();
        }

        /// <summary>
        /// enable effects, e.g. vertex coloring
```

```
        /// </summary>
        private void setupEffect()
        {
            m_Effect.VertexColorEnabled = true;
        }

        /// <summary>
        /// setup the camera position
        /// </summary>
        private void setUpCamera()
        {
            m_ViewMatrix = Matrix.CreateLookAt(new Vector3(0, 0, 2),
             new Vector3(0, 0, 0), new Vector3(0, 1, 0));
            m_ProjectionMatrix =
             Matrix.CreatePerspectiveFieldOfView(MathHelper.PiOver4,
             m_Device.Viewport.AspectRatio, 1.0f, 100);
        }

        /// <summary>
        /// create 3 vertices for the animated triangle
        /// </summary>
        private void setupVertices()
        {
            m_Vertices = new VertexPositionColor[3];

            m_Vertices[0].Position = new Vector3(-0.5f, -0.5f, 0f);
            m_Vertices[0].Color = Color.Red;
            m_Vertices[1].Position = new Vector3(0, 0.5f, 0f);
            m_Vertices[1].Color = Color.Green;
            m_Vertices[2].Position = new Vector3(0.5f, -0.5f, 0f);
            m_Vertices[2].Color = Color.Yellow;
        }

        /// <summary>
        /// UnloadContent will be called once per game and is the place to unload
        /// all content.
        /// </summary>
        protected override void UnloadContent()
        {
            // nothing to do
        }

        /// <summary>
        /// Allows the game to run logic such as updating the world,
        /// checking for collisions, gathering input, and playing audio.
        /// </summary>
        /// <param name="gameTime">Provides a snapshot of timing values.</param>
        protected override void Update(GameTime gameTime)
        {
            // Allows the game to exit
            if (GamePad.GetState(PlayerIndex.One).Buttons.Back == ButtonState.Pressed)
                this.Exit();

            // increment angle for triangle rotation
            m_Angle += 0.05f;

            base.Update(gameTime);
        }

        /// <summary>
        /// This is called when the game should draw itself.
```

```
        /// </summary>
        /// <param name="gameTime">Provides a snapshot of timing values.</param>
        protected override void Draw(GameTime gameTime)
        {
            m_Device.Clear(Color.CornflowerBlue);

            RasterizerState rs = new RasterizerState();
            rs.CullMode = CullMode.None;
            m_Device.RasterizerState = rs;

            m_Effect.View = m_ViewMatrix;
            m_Effect.Projection = m_ProjectionMatrix;

            // apply rotation
            Vector3 rotAxis = new Vector3(0, m_Angle, 0);
            rotAxis.Normalize();
            Matrix worldMatrix = Matrix.CreateFromAxisAngle(rotAxis, m_Angle);
            m_Effect.World = worldMatrix;

            // apply all effects on each graphic primitive
            foreach (EffectPass pass in m_Effect.CurrentTechnique.Passes)
            {
                pass.Apply();
                m_Device.DrawUserPrimitives(PrimitiveType.TriangleList,
                    m_Vertices, 0, 1,
                    VertexPositionColor.VertexDeclaration);
            }

            // draw the current frame
            base.Draw(gameTime);
        }
    }
}
```

5.8 Personal Information Management

This example shows how to use a *SaveContactTask*[17] to enable the user to save a new contact from within an application. Similar to the SMS example, an application is not able to save a Contact directly, but instead uses a Task to launch the native Contact application with prepopulated data.

This example uses 2 *TextBoxes* to get the new contacts last name and mobile phone number. After reading the values, the *SaveContactTask* will be started. When the user submits the new contact or cancels the task, the application will continue by calling the *SaveContactTask.Completed* EventHandler, where the application can react to the result.

17 http://msdn.microsoft.com/en-US/library/windowsphone/develop/microsoft.phone.tasks.savecontacttask(v=vs.105).aspx

References added:

None

XAML:

```
<phone:PhoneApplicationPage
    x:Class="PersonalInformationManagement.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
    xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d" d:DesignWidth="480" d:DesignHeight="768"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
    SupportedOrientations="Portrait" Orientation="Portrait"
    shell:SystemTray.IsVisible="True">

    <!--LayoutRoot is the root grid where all page content is placed-->
    <Grid x:Name="LayoutRoot" Background="Transparent">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>

        <!--TitlePanel contains the name of the application and page title-->
        <StackPanel x:Name="TitlePanel" Grid.Row="0" Margin="12,17,0,28">
            <TextBlock x:Name="ApplicationTitle" Text="MediaUnderstanding"

                    Style="{StaticResource PhoneTextNormalStyle}"/>
            <TextBlock x:Name="PageTitle" Text="Personal Information Management"
                    Margin="9,7,0,0" Style="{StaticResource PhoneTextTitle1Style}"/>
        </StackPanel>
        <!--ContentPanel - place additional content here-->
        <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
            <TextBlock HorizontalAlignment="Left" Margin="10,15,0,0"
             TextWrapping="Wrap" Text="Name" VerticalAlignment="Top"/>
            <TextBox x:Name="tbName" HorizontalAlignment="Left" Height="72"
                    Margin="0,42,0,0" TextWrapping="Wrap" Text="TestContact"
                    VerticalAlignment="Top" Width="456"/>
            <TextBlock HorizontalAlignment="Left" Margin="10,114,0,0"
             TextWrapping="Wrap" Text="Phone Number" VerticalAlignment="Top"
                    RenderTransformOrigin="2.171,7.148"/>
            <TextBox x:Name="tbNumber" HorizontalAlignment="Left" Height="72"
                    Margin="0,146,0,0" TextWrapping="Wrap" Text="01 234 45 678"
                    VerticalAlignment="Top" Width="456"/>
            <Button x:Name="btCreate" Content="Create" HorizontalAlignment="Left"
                    Margin="224,218,0,0" VerticalAlignment="Top"
                    Click="btCreate_Click" Width="232"/>
        </Grid>
    </Grid>

</phone:PhoneApplicationPage>
```

Code Behind:

```
using Microsoft.Phone.Controls;
using Microsoft.Phone.Tasks;
using System;
using System.Windows;

namespace PersonalInformationManagement
{
    /// <summary>
    /// This application uses a <see cref="Microsoft.Phone.Tasks.SaveContactTask"/>
    /// to enable the user to add a new contact
    /// </summary>
    /// <remarks>
    /// Bachelorthesis - "Media Programming on Mobile Devices (Windows Phone)"
    /// by Markus Besau
    /// July, 2013
    /// </remarks>
    public partial class MainPage : PhoneApplicationPage
    {
        private SaveContactTask m_SavingTask;

        public MainPage()
        {
            InitializeComponent();

            // create new SaveContactTask and add EventHandler
            m_SavingTask = new SaveContactTask();
            m_SavingTask.Completed += new
                    EventHandler<SaveContactResult>(saveContactTask_Completed);
        }

        private void saveContactTask_Completed(object _sender,
             SaveContactResult _result)
        {
            switch (_result.TaskResult)
            {
                // saving Contact was successful
                case TaskResult.OK:
                   {
                        MessageBox.Show("Contact saved");
                        break;
                   }
                // Task was canceled
                case TaskResult.Cancel:
                   {
                        MessageBox.Show("Saving contact cancelled");
                        break;
                   }
                // Task could not be completed
                case TaskResult.None:
                   {
                        MessageBox.Show("Contact could not be saved");
                        break;
                   }
            }

        }

        private void btCreate_Click(object sender, RoutedEventArgs e)
        {
            // create new Contact
```

```
            m_SavingTask.LastName = tbName.Text.Trim();
            m_SavingTask.MobilePhone = tbNumber.Text.Trim();

            // many other properties can be set, e.g.:
            // m_SavingTask.HomeAddressCity = "Vienna";

            // save new Contact
            m_SavingTask.Show();
        }
    }
}
```

5.9 Multimedia: Check Capabilities and Media Playback

This example shows how to play a video using the *MediaElement*[18] control. A short video of the movie Big Buck Bunny[19] (under the Creative Commence license) has been added to the application for demonstration purposes.

After setting the media source and starting the *Play( )* function on the *MediaElement*, the playback starts.

If the device isn't able to play the media (e.g. because it's coded in an unsupported way) the *MediaElement.MediaFailed* event will be fired which allows developers to check for media playback capabilities.

References added:

None

XAML:

```
<phone:PhoneApplicationPage
    x:Class="MediaPlayback.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
    xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d" d:DesignWidth="480" d:DesignHeight="768"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
    SupportedOrientations="Portrait" Orientation="Portrait"
    shell:SystemTray.IsVisible="True">

    <!--LayoutRoot is the root grid where all page content is placed-->
```

18 http://msdn.microsoft.com/en-us/library/system.windows.controls.mediaelement(v=vs.95).aspx

19 http://www.bigbuckbunny.org/

```
    <Grid x:Name="LayoutRoot" Background="Transparent">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>

        <!--TitlePanel contains the name of the application and page title-->
        <StackPanel x:Name="TitlePanel" Grid.Row="0" Margin="12,17,0,28">
            <TextBlock x:Name="ApplicationTitle" Text="MediaUnderstanding"
                    Style="{StaticResource PhoneTextNormalStyle}"/>
            <TextBlock x:Name="PageTitle" Text="MediaPlayback" Margin="9,-7,0,0"
                    Style="{StaticResource PhoneTextTitle1Style}"/>
        </StackPanel>

        <!--ContentPanel - place additional content here-->
        <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
            <MediaElement x:Name="mediaPlayer" HorizontalAlignment="Left" Margin="0"/>
        </Grid>
    </Grid>

</phone:PhoneApplicationPage>
```

Code Behind:

```
using Microsoft.Phone.Controls;
using System;
using System.Windows;

/// <summary>
/// This example uses a <see cref="System.Windows.Controls.MediaElement"/>
/// for media playback. A short video of the movie "Big Buck Bunny"
/// (<see href="http://www.bigbuckbunny.org/"/>) which was published under the
/// Creative Commence license has been added to the application
/// folder and will be shown after the application starts.
/// If the device can't play the media for whatever reason, a
/// message will be shown.
/// </summary>
/// <remarks>
/// Bachelorthesis - "Media Programming on Mobile Devices (Windows Phone)"
/// by Markus Besau
/// July, 2013
/// </remarks>
namespace MediaPlayback
{
    public partial class MainPage : PhoneApplicationPage
    {
        public MainPage()
        {
            InitializeComponent();

            // set media source and start playback
            mediaPlayer.Source = new Uri("bigbuckbunny.m4v", UriKind.Relative);
            mediaPlayer.Play();
            mediaPlayer.MediaFailed += new

             EventHandler<ExceptionRoutedEventArgs>(mediaPlayer_PlaybackFailed);
        }

        private void mediaPlayer_PlaybackFailed(object sender,
             ExceptionRoutedEventArgs e)
```

```
        {
            // display a message if the device can't play the media
            MessageBox.Show("Can't playback media!");
        }
    }
}
```

5.10 Multimedia: Media Recording – AUDIO

This application shows how to record audio from a microphone on the device using the *Microphone*[20] API. The devices default microphone can be accessed by *Microphone.Default*, all other microphones, if available, can be access by the *ReadOnlyCollection Microphone.All*.

The Microphone output is being written to a *MemoryStream*[21]. Any other stream could be used equally.

The playback uses a *SoundEffect*[22] and *SoundEffectInstance*[23]. A *SoundEffectInstance* is needed if it should be able to stop the playback, otherwise it is sufficient to just use a *SoundEffect*.

Another way to play audio would be to use a *MediaElement* as shown in the "Multimedia: Check Capabilities and Media Playback" example

Note: Since this application uses the XNA Framework, it's necessary to hook up a *XNAAsyncDispatcher* in the App.xaml.cs.

References added:

Microsoft.Xna.Framework

XAML:

```
<phone:PhoneApplicationPage
    x:Class="MediaRecordingAudio.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
    xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d" d:DesignWidth="480" d:DesignHeight="768"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
```

20 http://msdn.microsoft.com/en-us/library/microsoft.xna.framework.audio.microphone.aspx

21 http://msdn.microsoft.com/de-de/library/system.io.memorystream.aspx

22 http://msdn.microsoft.com/en-us/library/microsoft.xna.framework.audio.soundeffect.aspx

23 http://msdn.microsoft.com/en-us/library/microsoft.xna.framework.audio.soundeffectinstance.aspx

```
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
    SupportedOrientations="Portrait" Orientation="Portrait"
    shell:SystemTray.IsVisible="True">

    <!--LayoutRoot is the root grid where all page content is placed-->
    <Grid x:Name="LayoutRoot" Background="Transparent">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>

        <!--TitlePanel contains the name of the application and page title-->
        <StackPanel x:Name="TitlePanel" Grid.Row="0" Margin="12,17,0,28">
            <TextBlock x:Name="ApplicationTitle" Text="MediaUnderstanding"
             Style="{StaticResource PhoneTextNormalStyle}"/>
           <TextBlock x:Name="PageTitle" Text="AUDIO Recording" Margin="9,-7,0,0"
             Style="{StaticResource PhoneTextTitle1Style}"/>
        </StackPanel>

        <!--ContentPanel - place additional content here-->
        <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
            <Button x:Name="btRecord" Content="Start Recording" Height="80"
             VerticalAlignment="Top" Click="btRecord_Click" />
            <Button x:Name="btPlay" IsEnabled="False" Content="Start Playing"
             Height="80" Margin="0,80,0,0" VerticalAlignment="Top"
             Click="btPlay_Click" />
        </Grid>
    </Grid>

</phone:PhoneApplicationPage>
```

Code Behind:

```
using Microsoft.Phone.Controls;
using Microsoft.Xna.Framework.Audio;
using System;
using System.IO;
using System.Windows;

namespace MediaRecordingAudio
{
    /// <summary>
    /// This application shows how to record audio from the built in microphone
    /// and how to save it using a <see cref="System.IO.MemoryStream"/>.
    /// (note: any other stream could be used equally)
    /// The second part shows how to play audio using
    /// <see cref="Microsoft.XNA.Framework.Audio.SoundEffect"/> and
    /// <see cref="Microsoft.XNA.Framework.Audio.SoundEffectInstance"/>.
    /// A SoundEffectInstance is needed if it should be able to stop the
    /// playback. Otherwise it's sufficient to just use a SoundEffect.
    ///
    /// Another way to play audio would be to use a
    /// <see cref="System.Windows.Controls.MediaElement"/> and
    /// hide the control.
    /// </summary>
    /// <remarks>
    /// Bachelorthesis - "Media Programming on Mobile Devices (Windows Phone)"
    /// by Markus Besau
    /// July, 2013
```

```
    /// </remarks>
    public partial class MainPage : PhoneApplicationPage
    {
        private bool m_Recording;
        private bool m_Playing;
        private Microphone m_Mic;
        private MemoryStream m_MemoryStream;
        private SoundEffect m_SoundEffect;
        private SoundEffectInstance m_SoundEffectInstance;

        public MainPage()
        {
            InitializeComponent();

            m_Mic = Microphone.Default;
            m_Mic.BufferReady += Microphone_BufferReady;
        }

        private void btRecord_Click(object sender, RoutedEventArgs e)
        {
            if (m_Recording)
            {
                // STOP RECORDING
                m_Mic.Stop();
                m_MemoryStream.Position = 0; // reset the streams position

                btRecord.Content = "Start Recording";
                m_Recording = false;
                btPlay.IsEnabled = true;
            }
            else
            {
                // START RECORDING
                m_MemoryStream = new MemoryStream();
                m_Mic.Start();

                btRecord.Content = "Stop Recording";
                m_Recording = true;
            }
        }

        private void btPlay_Click(object sender, RoutedEventArgs e)
        {
            if (m_Playing)
            {
                // STOP PLAYING
                m_SoundEffectInstance.Stop();

                btPlay.Content = "Start Playing";
                m_Playing = false;
            }
            else
            {
                // START PLAYING
                m_SoundEffect = new SoundEffect(m_MemoryStream.ToArray(),
                    m_Mic.SampleRate, AudioChannels.Mono);

                // create a SoundEffectInstance to be able to stop the playback
                // if there is no need to be able to stop the playback,
                // one can skip the SoundEffectInstance and simply use
                    SoundEffect.Play()
```

```
                m_SoundEffectInstance = m_SoundEffect.CreateInstance();
                m_SoundEffectInstance.Play();

                btPlay.Content = "Stop Playing";
                m_Playing = true;
            }
        }

        private void Microphone_BufferReady(object sender, EventArgs e)
        {
            byte[] buffer = new byte[1024];
            int bytesRead = 0;

            while ((bytesRead = m_Mic.GetData(buffer, 0, buffer.Length)) > 0)
            {
                // write microphone input to memory stream;
                // any other kind of stream can be used equally
                m_MemoryStream.Write(buffer, 0, bytesRead);
            }
        }
    }
}
```

App.xaml.cs:

```
public App()
{
      [...]
      // hook up the XNA Async Dispatcher
      this.ApplicationLifetimeObjects.Add(new
            XNAAsyncDispatcher(TimeSpan.FromMilliseconds(50)));
}

public class XNAAsyncDispatcher : IApplicationService
{
      private DispatcherTimer frameworkDispatcherTimer;

      public XNAAsyncDispatcher(TimeSpan dispatchInterval)
      {
            this.frameworkDispatcherTimer = new DispatcherTimer();
            this.frameworkDispatcherTimer.Tick +=
                new EventHandler(frameworkDispatcherTimer_Tick);
            this.frameworkDispatcherTimer.Interval = dispatchInterval;
      }

      void IApplicationService.StartService(ApplicationServiceContext context)
      {
            this.frameworkDispatcherTimer.Start();
      }

      void IApplicationService.StopService()
      {
            this.frameworkDispatcherTimer.Stop();
      }

      void frameworkDispatcherTimer_Tick(object sender, EventArgs e)
      {
            FrameworkDispatcher.Update();
      }
}
```

5.11 Multimedia: Media Recording – VIDEO

This example shows how to display a camera video stream, capture the video and playback it after being saved.

To get the camera stream calling *CaptureDeviceConfiguration.GetDefaultVideoCapture-Device( )* first is necessary to initiate the camera. After that, we can use a *CaptureSource* in combination with a *VideoBrush* and *Rectangle* to display the camera stream.

This application saves the video to the applications isolated storage using a *FileSink*. (Note: the video could also be saved to the Windows Phone media library).

For playback the video is being read using a *IsolatedStorageFileStream* and a *MediaElement*.

Note: If the camera stream shall be shown after watching a playback, it is important to disable the *MediaElement*, especially disposing the *FileStream*, otherwise the camera stream can't be displayed again.

References added:
None

XAML:

```
<phone:PhoneApplicationPage
    x:Class="MediaRecordingVideo.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
    xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d" d:DesignWidth="480" d:DesignHeight="768"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
    SupportedOrientations="Landscape"
    shell:SystemTray.IsVisible="True">

    <!--LayoutRoot is the root grid where all page content is placed-->
    <Grid x:Name="LayoutRoot">
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*"/>
            <ColumnDefinition Width="150"/>
        </Grid.ColumnDefinitions>

        <Rectangle x:Name="viewfinderRectangle" Height="480" Width="640"/>
        <MediaElement x:Name="mediaPlayer" Height="480" Width="640"
                      Visibility="Collapsed" />
        <StackPanel Orientation="Vertical" Grid.Column="1" Margin="0"
                      VerticalAlignment="Center">
            <Button x:Name="btRecord" Content="REC" Margin="0" Width="120"
```

```
                    Click="btRecord_Click"/>
            <Button x:Name="btPlay" Content="PLAY" Margin="0" Width="120"
                    Click="btPlay_Click" />
        </StackPanel>
    </Grid>

</phone:PhoneApplicationPage>
```

Code Behind:

```
using Microsoft.Phone.Controls;
using System;
using System.IO;
using System.IO.IsolatedStorage;
using System.Windows;
using System.Windows.Media;

namespace MediaRecordingVideo
{
    /// <summary>
    /// This applications shows how to capture and playback video using
    /// <see cref="System.Windows.Media.VideoCaptureDevice"/> and <see
                    cref="System.Windows.Media.CaptureSource"/>
    /// for capturing and <see cref="System.Windows.Controls.MediaElement"/> for
         playback.
    ///
    /// The camera stream is being displayed in a <see
                    cref="System.Windows.Shapes.Rectangle"/> using
    /// a <see cref="System.Windows.Media.VideoBrush"/> as source.
    ///
    /// The video is being saved in the applications isolated storage using a
    /// <see cref="System.Windows.Media.FileSink"/>. It could also be saved to
    /// the Windows Phone media library.
    ///
    /// For playback the video is being read using a
    /// <see cref="System.IO.IsolatedStorage.IsolatedStorageFileStream"/>.
    /// </summary>
    /// <remarks>
    /// Bachelorthesis - "Media Programming on Mobile Devices (Windows Phone)"
    /// by Markus Besau
    /// July, 2013
    /// </remarks>
    public partial class MainPage : PhoneApplicationPage
    {
        private VideoBrush m_VideoRecorderBrush;
        private CaptureSource m_CaptureSource;
        private VideoCaptureDevice m_CaptureDevice;
        private FileSink m_FileSink;
        private string m_FileName = "MediaUnderstanding_VideoRecording.mp4";
        private IsolatedStorageFileStream m_FileStream;
        private bool m_Recording = false;

        public MainPage()
        {
            InitializeComponent();

            initializeVideoRecorder();
        }

        public void initializeVideoRecorder()
```

```
        {
            // create the VideoRecorder objects
            m_CaptureSource = new CaptureSource();
            m_FileSink = new FileSink();

            // get the default camera
            m_CaptureDevice =
             CaptureDeviceConfiguration.GetDefaultVideoCaptureDevice();

            // initialize the camera if it exists on the device
            if (m_CaptureDevice != null)
            {
                // create the VideoBrush for the viewfinder
                m_VideoRecorderBrush = new VideoBrush();
                m_VideoRecorderBrush.SetSource(m_CaptureSource);

                // display the viewfinder image on the rectangle
                viewfinderRectangle.Fill = m_VideoRecorderBrush;

                // start video capture and display it on the viewfinder
                m_CaptureSource.Start();
            }
            else
            {
                MessageBox.Show("The device does not have a camera!");
            }
        }

        private void startVideoPreview()
        {
            try
            {
                // add captureSource to videoBrush
                m_VideoRecorderBrush.SetSource(m_CaptureSource);

                // add videoBrush to the viewFinder
                viewfinderRectangle.Fill = m_VideoRecorderBrush;

                // start camera stream
                m_CaptureSource.Start();
            }
            catch (Exception e)
            {
                // if starting videoPreview fails, display the error
                MessageBox.Show(e.Message);
            }
        }

        private void startRecording()
        {
            try
            {
                // stop capture to hook up fileSink
                m_CaptureSource.Stop();

                // hooking up the camera-file-stream
                m_FileSink.CaptureSource = m_CaptureSource;
                m_FileSink.IsolatedStorageFileName = m_FileName;

                // start recording
                m_CaptureSource.Start();
```

```
            }
            catch (Exception e)
            {
                // if recording fails, display the error
                MessageBox.Show(e.Message);
            }
        }

        private void stopRecording()
        {
            try
            {
                // stop capture to unhook fileSink
                m_CaptureSource.Stop();

                // disconnect fileSink
                m_FileSink.CaptureSource = null;
                m_FileSink.IsolatedStorageFileName = null;

                startVideoPreview();
            }
            catch (Exception e)
            {
                // if stop fails, display the error
                MessageBox.Show(e.Message);
            }
        }

        private void btRecord_Click(object sender, EventArgs e)
        {
            if (!m_Recording)
            {
                // start recording
                startRecording();
                btRecord.Content = "STOP";
                m_Recording = true;
            }
            else
            {
                // stop recording
                stopRecording();
                btRecord.Content = "REC";
                m_Recording = false;
            }
        }

        private void btPlay_Click(object sender, EventArgs e)
        {
            // set visibility of mediaelement and viewfinder
            mediaPlayer.Visibility = System.Windows.Visibility.Visible;
            viewfinderRectangle.Visibility = System.Windows.Visibility.Collapsed;

            // stop capture
            m_CaptureSource.Stop();

            // create the file stream and attach it to the MediaElement
            m_FileStream = new IsolatedStorageFileStream(m_FileName, FileMode.Open,
                    FileAccess.Read,
              IsolatedStorageFile.GetUserStoreForApplication());
            mediaPlayer.SetSource(m_FileStream);
```

```
            // add an event handler for the end of playback
            mediaPlayer.MediaEnded += new RoutedEventHandler(playbackEnded);

            // start video playback
            mediaPlayer.Play();
        }

        private void playbackEnded(object sender, RoutedEventArgs e)
        {
            // set visibility of mediaelement and viewfinder
            mediaPlayer.Visibility = System.Windows.Visibility.Collapsed;
            viewfinderRectangle.Visibility = System.Windows.Visibility.Visible;

            disposeVideoPlayer();
            startVideoPreview();
        }

        private void disposeVideoPlayer()
        {
            /*
             * disabling the MediaElement, especially unhooking the FileStream
             * is necessary, otherwise the ViewFinder won't start properly
             */

            if (mediaPlayer != null)
            {
                // stop the MediaElement
                mediaPlayer.Stop();

                // remove playback objects
                mediaPlayer.Source = null;
                m_FileStream = null;

                // remove the event handler
                mediaPlayer.MediaEnded -= playbackEnded;
            }
        }
    }
}
```

5.12 Take A SnapShot Image

This application shows how to take a picture from the built-in camera. This time, the camera stream is being displayed using a *VideoBrush*[24] embedded in a *Canvas*[25]. Since this application only supports portrait mode (although landscape would also be no problem), the image needs to be rotated using a *RotateTransform*[26].

24 http://msdn.microsoft.com/en-us/library/system.windows.media.videobrush(v=vs.95).aspx
25 http://msdn.microsoft.com/en-us/library/system.windows.controls.canvas(v=vs.95).aspx
26 http://msdn.microsoft.com/en-us/library/system.windows.media.rotatetransform(v=vs.95).aspx

Unlike the other examples, this application saves the image to the Windows Phone media library using *MediaLibrary*[27] to store the image.

To display the last taken image it is necessary to load it from the store, get it in a *BitmapImage*[28] and set it as a source for the *Image*[29].

References added:

Microsoft.Xna.Framework

XAML:

```
<phone:PhoneApplicationPage
    x:Class="SnapShot.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
    xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d" d:DesignWidth="480" d:DesignHeight="768"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
    SupportedOrientations="Portrait" Orientation="Portrait"
    shell:SystemTray.IsVisible="True">

    <!--LayoutRoot is the root grid where all page content is placed-->
    <Grid x:Name="LayoutRoot" Background="Transparent">
        <Grid.RowDefinitions>
            <RowDefinition/>
            <RowDefinition Height="80"/>
        </Grid.RowDefinitions>

        <Canvas x:Name="canViewFinder" Height="640" Width="480"
             VerticalAlignment="Top">
            <Canvas.Background>
                <VideoBrush x:Name="brushViewFinder"/>
            </Canvas.Background>
        </Canvas>
        <Image x:Name="image" Height="640" Width="480" VerticalAlignment="Top"
                    Visibility="Collapsed"/>
        <StackPanel Orientation="Horizontal" Grid.Row="1"
             HorizontalAlignment="Center">
            <Button x:Name="btSnapshot" Content="Take Snapshot"
             Click="btSnapshot_Click"/>
            <Button x:Name="btView" Content="View last picture" Click="btView_Click"/>
        </StackPanel>
    </Grid>

</phone:PhoneApplicationPage>
```

27 http://msdn.microsoft.com/en-us/library/microsoft.xna.framework.media.medialibrary.aspx

28 http://msdn.microsoft.com/en-us/library/system.windows.media.imaging.bitmapimage(v=vs.95).aspx

29 http://msdn.microsoft.com/en-us/library/system.windows.controls.image(v=vs.95).aspx

Code Behind:

```
using Microsoft.Devices;
using Microsoft.Phone.Controls;
using Microsoft.Xna.Framework.Media;
using System;
using System.Windows;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Linq;

namespace SnapShot
{
    /// <summary>
    /// This application shows how to take a picture from the built-in camera.
    /// This time the camera stream is being displayed using a
    /// <see cref="System.Windows.Media.VideoBrush"/> embedded in a
    /// <see cref="System.Windows.Controls.Canvas"/>.
    ///
    /// Since this application only supports portrait a
    /// <see cref="System.Windows.Media.RotateTransform"/> is needed.
    ///
    /// Unlike the other examples this application saves the image
    /// to the Windows Phone media library using
    /// <see cref="Microsoft.XNA.Framework.Media.MediaLibrary"/>
    /// to store the image.
    ///
    /// To display the last taken image it's necessary to load it
    /// from the store, get it in a <see
      cref="System.Windows.Media.Imaging.BitmapImage"/>
    /// and set it as source for the <see cref="System.Windows.Controls.Image"/>.
    /// </summary>
    /// <remarks>
    /// Bachelorthesis - "Media Programming on Mobile Devices (Windows Phone)"
    /// by Markus Besau
    /// July, 2013
    /// </remarks>
    public partial class MainPage : PhoneApplicationPage
    {
        private PhotoCamera m_Camera;
        private string m_FileName = "MediaUnderstanding_Snapshot.jpg";
        private MediaLibrary m_Library = new MediaLibrary();
        private bool m_Viewing = false;

        public MainPage()
        {
            InitializeComponent();
            InitializeCamera();
        }

        private void InitializeCamera()
        {
            // get the camera
            m_Camera = new PhotoCamera(CameraType.Primary);

            // rotate the image
            RotateTransform rotate = new RotateTransform();
            rotate.Angle = 90;
            rotate.CenterX = 240;
            rotate.CenterY = 320;
            brushViewFinder.Transform = rotate;
```

```
            // set the camera stream as source
            brushViewFinder.SetSource(m_Camera);
            // hook up eventhandlers
            m_Camera.CaptureImageAvailable += new
                    EventHandler<ContentReadyEventArgs>(cam_CaptureImageAvailable);
        }

        private void btSnapshot_Click(object sender, RoutedEventArgs e)
        {
            // take a picture
            m_Camera.CaptureImage();
        }

        private void btView_Click(object sender, RoutedEventArgs e)
        {
            if (m_Viewing)
            {
                // STOP VIEWING

                // hide image and show viewFinder
                canViewFinder.Visibility = Visibility.Visible;
                image.Visibility = Visibility.Collapsed;

                m_Viewing = false;
                btView.Content = "View last picture";
            }
            else
            {
                // START VIEWING
                // hide viewFinder and show image
                canViewFinder.Visibility = Visibility.Collapsed;
                image.Visibility = Visibility.Visible;
                m_Library = new MediaLibrary();

                // using a LINQ/Lambda Expression to get the picture
                // it would also be possible to simply loop through the Pictures-Array
                // to get the desired picture, but since LINQ is awesome we prefer
                    it ;)
                var pics = m_Library.Pictures.Where(p =>
                            p.Name.Equals(m_FileName)).ToList();
                if (pics.Count > 0)
                {
                    // load image as bitmap and set it as a source
                    BitmapImage bitmap = new BitmapImage();
                    bitmap.SetSource(pics.Last().GetImage());
                    image.Source = bitmap;
                }

                m_Viewing = true;
                btView.Content = "Hide picture";
            }
        }

        private void cam_CaptureImageAvailable(object sender, ContentReadyEventArgs e)
        {
            // save the picture to the Windows Phone media library
            m_Library.SavePictureToCameraRoll(m_FileName, e.ImageStream);
        }
    }
}
```

5.13 Process Video/Image – Threshold An Image

This application shows how to process an image and create its grayscale image.

A *Slider*[30] is being used to enable the user to select the desired threshold. RGB values are being calculated for each pixel and used to determine the colors grayscale representation. If the value exceeds the threshold, the pixels color will be changed to white, otherwise to black.

When all pixels have been processed, the image will be written using a *MemoryStream* and set as source for the *Image* control.

References added:

None

XAML:

```
<phone:PhoneApplicationPage
    x:Class="ProcessImage.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
    xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d" d:DesignWidth="480" d:DesignHeight="768"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
    SupportedOrientations="Portrait" Orientation="Portrait"
    shell:SystemTray.IsVisible="True">

    <!--LayoutRoot is the root grid where all page content is placed-->
    <Grid x:Name="LayoutRoot" Background="Transparent">
        <Grid.RowDefinitions>
            <RowDefinition/>
            <RowDefinition Height="160"/>
        </Grid.RowDefinitions>

        <Image x:Name="image" Height="640" Width="480" Source="/sample_photo_01.jpg"
                    VerticalAlignment="Top" Visibility="Visible"/>
        <StackPanel Orientation="Vertical" Grid.Row="1" HorizontalAlignment="Center">
            <Slider x:Name="slThreshold" Width="480" Minimum="0" Maximum="255"
             Value="128"
                    LargeChange="10" SmallChange="10"/>
            <Button x:Name="btThreshold" Content="Threshold it"
             Click="btThreshold_Click"/>
        </StackPanel>
    </Grid>

</phone:PhoneApplicationPage>
```

30 http://msdn.microsoft.com/en-us/library/system.windows.controls.slider(v=vs.95).aspx

Code Behind:

```
using Microsoft.Phone.Controls;
using System;
using System.IO;
using System.Windows;
using System.Windows.Media.Imaging;

namespace ProcessImage
{
    /// <summary>
    /// This application shows how to process an image and create
    /// its grayscale image.
    /// A <see cref="System.Windows.Controls.Slider"/> is being used
    /// to enable the user to select the desired threshold.
    /// RGB values are calculated for each pixel. After that the pixels
    /// grayscale representation is calculated. If the value exceeds the
    /// threshold, the pixels color will change to white, otherwise to black.
    ///
    /// When all pixels have been processed, the image will be written
    /// using a <see cref="System.IO.MemoryStream"/> and set as source
    /// for the shown <see cref="System.Windows.Controls.Image"/>.
    /// </summary>
    /// <remarks>
    /// Bachelorthesis - "Media Programming on Mobile Devices (Windows Phone)"
    /// by Markus Besau
    /// July, 2013
    /// </remarks>
    public partial class MainPage : PhoneApplicationPage
    {
        private bool m_Thresholded;

        public MainPage()
        {
            InitializeComponent();
        }

        private void btThreshold_Click(object sender, RoutedEventArgs e)
        {
            if (m_Thresholded)
            {
                // RESET IMAGE

                // load original image

               BitmapImage src = new BitmapImage(new Uri("sample_photo_01.jpg",
                    UriKind.Relative));
                image.Source = src;
                btThreshold.Content = "Threshold it";
                m_Thresholded = false;
            }
            else
            {
                // THRESHOLD IMAGE

                // get threshold from slider
                int threshold = (int)slThreshold.Value;

                // load current image into a writeable bitmap
```

```
                WriteableBitmap img = new WriteableBitmap(image, null);

                int idx = 0;
                foreach (var pixel in img.Pixels)
                {
                    // calculate RGB values from int representation
                    int r = (pixel & 0xFF);
                    int g = (pixel >> 8 & 0xFF);
                    int b = (pixel >> 8 & 0xFF);

                    // calculate gray value of current pixel
                    int gray = (int)(0.299 * r + 0.587 * g + 0.114 * b);

                    // threshold it
                    if (gray > threshold)
                        img.Pixels.SetValue(0xffffff, idx);
                    else
                        img.Pixels.SetValue(0x000000, idx);

                    idx++;
                }

                // write the thresholded image to a MemoryStream and display it
                using (MemoryStream ms = new MemoryStream())
                {
                    img.SaveJpeg(ms, img.PixelWidth, img.PixelHeight, 0, 100);
                    BitmapImage bmp = new BitmapImage();
                    bmp.SetSource(ms);
                    image.Source = bmp;
                }

                btThreshold.Content = "Reset";
                m_Thresholded = true;
            }
        }
    }
}
```

5.14 Accelerator And Gyroscope Sensor

This application shows how to use a *Gyroscope*[31] and *Accelerometer*[32] to obtain movement data of the device.

If the devices provides a *Gyroscope* it will be used to determine the movement, otherwise the *Accelerometer* will be used.

Any changes will be displayed to the corresponding *TextBoxes*.

31 http://msdn.microsoft.com/EN-US/library/windowsphone/develop/microsoft.devices.sensors.gyroscope(v=vs.105).aspx

32 http://msdn.microsoft.com/EN-US/library/windowsphone/develop/microsoft.devices.sensors.accelerometer(v=vs.105).aspx

References added:

Microsoft.Xna.Framework

Microsoft.Devices.Sensors

XAML:

```
<phone:PhoneApplicationPage
    x:Class="AcceleratorSensor.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
    xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d" d:DesignWidth="480" d:DesignHeight="768"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
    SupportedOrientations="Portrait" Orientation="Portrait"
    shell:SystemTray.IsVisible="True">

    <!--LayoutRoot is the root grid where all page content is placed-->
    <Grid x:Name="LayoutRoot" Background="Transparent">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>

        <!--TitlePanel contains the name of the application and page title-->
        <StackPanel x:Name="TitlePanel" Grid.Row="0" Margin="12,17,0,28">
            <TextBlock x:Name="ApplicationTitle" Text="MediaUnderstanding"
                       Style="{StaticResource PhoneTextNormalStyle}"/>
            <TextBlock x:Name="PageTitle" Text="Accelerator Sensor" Margin="9,-7,0,0"
                       Style="{StaticResource PhoneTextTitle1Style}"/>
        </StackPanel>

        <!--ContentPanel - place additional content here-->
        <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
            <Grid.RowDefinitions>
                <RowDefinition Height="70" />
                <RowDefinition Height="70" />
                <RowDefinition Height="70" />
                <RowDefinition Height="*" />
            </Grid.RowDefinitions>
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="40" />
                <ColumnDefinition />
            </Grid.ColumnDefinitions>

            <TextBlock Text="x:" Grid.Row="0" Grid.Column="0"
             VerticalAlignment="Center" />
            <TextBlock Text="y:" Grid.Row="1" Grid.Column="0"
             VerticalAlignment="Center" />
            <TextBlock Text="z:" Grid.Row="2" Grid.Column="0"
             VerticalAlignment="Center" />

            <TextBox IsReadOnly="True" Grid.Row="0" Grid.Column="1"
             Name="tbX"></TextBox>
            <TextBox IsReadOnly="True" Grid.Row="1" Grid.Column="1"
             Name="tbY"></TextBox>
```

```
            <TextBox IsReadOnly="True" Grid.Row="2" Grid.Column="1"
             Name="tbZ"></TextBox>
        </Grid>
    </Grid>
</phone:PhoneApplicationPage>
```

Code Behind:

```
using Microsoft.Devices.Sensors;
using Microsoft.Phone.Controls;
using Microsoft.Xna.Framework;
using System;
using System.Windows;

namespace AcceleratorSensor
{
    /// <summary>
    /// This application shows how to use a <see
      cref="Microsoft.Devices.Sensors.Gyroscope"/>
    /// and a <see cref="Microsoft.Devices.Sensors.Accelerometer"/> to obtain movement
    /// data of the device.
    /// If the device provides a Gyroscope it will be used to determine the movement,
    /// otherwise the Accelerometer will be used.
    /// Any changes will be displayed to the corresponding TextBoxes.
    /// </summary>
    /// <remarks>
    /// Bachelorthesis - "Media Programming on Mobile Devices (Windows Phone)"
    /// by Markus Besau
    /// July, 2013
    /// </remarks>
    public partial class MainPage : PhoneApplicationPage
    {
        Gyroscope m_Gyroscope;
        Accelerometer m_Accelerometer;

        public MainPage()
        {
            InitializeComponent();
            if (!Gyroscope.IsSupported)
            {
                // device does not have a gyroscope
                MessageBox.Show("sorry, your device doesn't support gyroscope data,
                           switching to Accelerometer!");

                // start Accelerometer
                m_Accelerometer = new Accelerometer();
                m_Accelerometer.CurrentValueChanged += new
                           EventHandler<SensorReadingEventArgs<AccelerometerReading>>(
                                  accelerometer_CurrentValueChanged);
                m_Accelerometer.Start();
            }
            else
            {
                // start Gyroscope
                m_Gyroscope = new Gyroscope();
                m_Gyroscope.CurrentValueChanged += gyroscope_CurrentValueChanged;
                m_Gyroscope.Start();
            }
```

```
        }

        private void accelerometer_CurrentValueChanged(object sender,
                    SensorReadingEventArgs<AccelerometerReading> e)
        {
            // accelerometer sensor data changed, update UI
            Dispatcher.BeginInvoke(() => UpdateUI(e.SensorReading));
        }

        private void gyroscope_CurrentValueChanged(object sender,
                    SensorReadingEventArgs<GyroscopeReading> e)
        {
            // gyroscope sensor data changed, update UI
            Dispatcher.BeginInvoke(() => UpdateUI(e.SensorReading));
        }

        private void UpdateUI(AccelerometerReading _reading)
        {
            // write X/Y/Z values to TextBoxes
            Vector3 accReading = _reading.Acceleration;
            tbX.Text = accReading.X.ToString();
            tbY.Text = accReading.Y.ToString();
            tbZ.Text = accReading.Z.ToString();
        }

        private void UpdateUI(GyroscopeReading _reading)
        {
            // write X/Y/Z values to TextBoxes
            Vector3 gyroReading = _reading.RotationRate;
            tbX.Text = gyroReading.X.ToString();
            tbY.Text = gyroReading.Y.ToString();
            tbZ.Text = gyroReading.Z.ToString();
        }
    }
}
```

5.15 Data-Binding

.NET *Data-Binding*[33] provides a simple way for applications to present and interact with data and is used on all .NET project types which uses XAML for the UI (Silverlight, WPF, Silverlight for Windows Phone). Controls and their properties can be bound directly to various data sources. Data-binding can be used both on content and design properties.

Control properties are data bound by using the *{Binding MyValue}* key-phrase. All other Data-Binding properties (like the *StringFormat*,

33 http://code.msdn.microsoft.com/wpapps/Simple-data-binding-in-XAML-39e8e9ad

UpdateSourceTrigger, *One Time / One Way / Two Way*) can be set between the curly braces.

Child properties can be referred to as MainProperty.ChildProperty (e.g. Adress.Housenumber). The *DataContext* can also be bound, if the child control should use a child property.

References added:

None

XAML:

```
<phone:PhoneApplicationPage
    x:Class="Databinding.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
    xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d" d:DesignWidth="480" d:DesignHeight="768"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
    SupportedOrientations="Portrait" Orientation="Portrait"
    shell:SystemTray.IsVisible="True"
    xmlns:toolkit="clr-
      namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone.Controls.Toolkit">

    <!--LayoutRoot is the root grid where all page content is placed-->
    <Grid x:Name="LayoutRoot" Background="Transparent">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>

        <!--TitlePanel contains the name of the application and page title-->
        <StackPanel x:Name="TitlePanel" Grid.Row="0" Margin="12,17,0,28">
            <TextBlock x:Name="ApplicationTitle" Text="MediaUnderstanding"
                    Style="{StaticResource PhoneTextNormalStyle}"/>
            <TextBlock x:Name="PageTitle" Text="DataBinding" Margin="9,-7,0,0"
                    Style="{StaticResource PhoneTextTitle1Style}"/>
        </StackPanel>

        <!--ContentPanel - place additional content here-->
        <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
            <StackPanel Orientation="Vertical">
                <!--other properties than simple text can be data bound as well, e.g.
                          color-->
                <StackPanel Orientation="Vertical" Background="{Binding Color}">
                    <!--databinding also works with subelements/properties, e.g.
                                University.Title-->
                    <TextBlock Text="{Binding University.Title}" Height="50"/>
                    <!--the datacontext can also be bound,
                       the element and its children will use the bound element as
                                datacontext-->
                    <TextBlock DataContext="{Binding University}" Text="{Binding ID}"
```

```
                        Height="50" />
                </StackPanel>
                <TextBlock Text="{Binding Studies}" Height="50" />
                <TextBlock Text="{Binding Program}" Height="50" />
                <TextBlock Text="{Binding ID}" Height="50" />
                <Button x:Name="btNext" Content="Next Course" Click="btNext_Click"/>
            </StackPanel>
        </Grid>
    </Grid>

</phone:PhoneApplicationPage>
```

Code Behind:

```
using System.Collections.Generic;
using System.Windows;
using System.Windows.Media;
using Microsoft.Phone.Controls;

namespace Databinding
{
    /// <summary>
    /// This application shows how to use Data-Binding on UI controls. Elements
    /// can be bound to a variety of data sources. If the Data-Binding has been
    /// hooked up properly, the UI does not need a notification when the data
    /// changes but will display it automatically.
    /// Data-Binding can also be used for user input.
    /// If the <see cref="System.Windows.Data.Binding.UpdateSourceTrigger"/>
    /// has been set the user input will be automatically written in the bound data.
    /// </summary>
    /// <remarks>
    /// Bachelorthesis - "Media Programming on Mobile Devices (Windows Phone)"
    /// by Markus Besau
    /// July, 2013
    /// </remarks>
    public partial class MainPage : PhoneApplicationPage
    {
        private List<Course> m_ListCourses;
        private int m_CurrentCourse;

        public MainPage()
        {
            InitializeComponent();

            // create list and dummy values
            m_ListCourses = new List<Course>();
            m_ListCourses.Add(new Course()
            {
                University = new University() {ID = 1, Title = "TU Vienna" },
                Studies = "Computer Sciences",
                Program = "SEPM",
                Color = new SolidColorBrush(Colors.Gray),
                ID = "Course_002"
            });
            m_ListCourses.Add(new Course()
            {
                University = new University() { ID = 2, Title = "Uni Vienna" },
                Studies = "History",
                Program = "Greek History",
                Color = new SolidColorBrush(Colors.Magenta),
```

```
                ID = "Course_123"
            });
             m_ListCourses.Add(new Course()
            {
                University = new University() { ID = 3, Title = "WU Vienna"},
                Studies = "Business",
                Program = "Some Business Stuff",
                Color = new SolidColorBrush(Colors.Orange),
                ID = "Course_548"
            });

             // set datacontext for MainPage.xaml and all children
             this.DataContext = m_ListCourses[0];
            m_CurrentCourse = 1;
        }

        private void btNext_Click(object sender, RoutedEventArgs e)
        {
             // set DataContext for the Grid "ContentPanel" and all children
            ContentPanel.DataContext = m_ListCourses[m_CurrentCourse];
            m_CurrentCourse++;
             if (m_CurrentCourse == m_ListCourses.Count)
                m_CurrentCourse = 0;
        }
    }

    public class Course
    {
        public University University { get; set; }
        public string Studies { get; set; }
        public string Program { get; set; }
        public Brush Color { get; set; }
        public string ID { get; set; }
    }

    public class University
    {
        public int ID { get; set; }
        public string Title { get; set; }
    }
}
```

5.16 Application Menu

This application shows how to use the *ApplicationBar*[34] for Silverlight for Windows Phone applications to provide the user with a simple but intuitive and consistent menu.

Both *ApplicationBarIconButtons* and *ApplicationBarMenuItems* are being used. The icons used are from the default icons which come with the Windows Phone SDK.

34 http://msdn.microsoft.com/en-us/library/windowsphone/develop/microsoft.phone.shell.applicationbar(v=vs.105).aspx

Further information:

http://msdn.microsoft.com/en-us/library/windowsphone/develop/ff431813(v=vs.105).aspx

References added:

None

XAML:

```
<phone:PhoneApplicationPage
    x:Class="ApplicationMenu.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
    xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d" d:DesignWidth="480" d:DesignHeight="768"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
    SupportedOrientations="Portrait" Orientation="Portrait"
    shell:SystemTray.IsVisible="True">

    <!--LayoutRoot is the root grid where all page content is placed-->
    <Grid x:Name="LayoutRoot" Background="Transparent">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>

        <!--TitlePanel contains the name of the application and page title-->
        <StackPanel x:Name="TitlePanel" Grid.Row="0" Margin="12,17,0,28">
            <TextBlock x:Name="ApplicationTitle" Text="MediaUnderstanding"
                    Style="{StaticResource PhoneTextNormalStyle}"/>
            <TextBlock x:Name="PageTitle" Text="App Menu" Margin="9,-7,0,0"
                    Style="{StaticResource PhoneTextTitle1Style}"/>
        </StackPanel>
        <Grid Grid.Row="1">
            <TextBlock x:Name="tbNumber" VerticalAlignment="Top"/>
        </Grid>
    </Grid>
    <phone:PhoneApplicationPage.ApplicationBar>
        <shell:ApplicationBar IsVisible="True" IsMenuEnabled="True">
            <shell:ApplicationBarIconButton x:Name="btIncrease"
                    IconUri="/Images/appbar.add.rest.png" Text="increase"
                    Click="btIncrease_Click"/>
            <shell:ApplicationBarIconButton x:Name="btDecrease"
                    IconUri="/Images/appbar.minus.rest.png" Text="decrease"
                    Click="btDecrease_Click" />
            <shell:ApplicationBar.MenuItems>
                <shell:ApplicationBarMenuItem x:Name="btReset" Text="reset"
                            Click="btReset_Click"/>
            </shell:ApplicationBar.MenuItems>
        </shell:ApplicationBar>
    </phone:PhoneApplicationPage.ApplicationBar>
</phone:PhoneApplicationPage>
```

Code Behind:

```
using Microsoft.Phone.Controls;
using System;

namespace ApplicationMenu
{
    /// <summary>
    /// This application shows how to use the <see
      cref="Microsoft.Phone.Shell.ApplicationBar"/> for
    /// Silverlight for Windows Phone Applications.
    /// Both <see cref="Microsoft.Phone.Shell.ApplicationBarIconButton"/>
    /// and <see cref="Microsoft.Phone.Shell.ApplicationBarMenuItem"/> are used.
    /// </summary>
    /// <remarks>
    /// Bachelorthesis - "Media Programming on Mobile Devices (Windows Phone)"
    /// by Markus Besau
    /// July, 2013
    /// </remarks>
    public partial class MainPage : PhoneApplicationPage
    {
        private int m_Number;

        public MainPage()
        {
            InitializeComponent();
            tbNumber.Text = m_Number.ToString();
        }

        private void btReset_Click(object sender, EventArgs e)
        {
            m_Number = 0;
            tbNumber.Text = m_Number.ToString();
        }

        private void btDecrease_Click(object sender, EventArgs e)
        {
            tbNumber.Text = (--m_Number).ToString();
        }

        private void btIncrease_Click(object sender, EventArgs e)
        {
            tbNumber.Text = (++m_Number).ToString();
        }
    }
}
```

5.17 LiveTiles

This application shows how to use Windows Phone LiveTiles using the *ShellTile*[35] API.

The application will create and add a new tile to the Windows Phone start screen and create a *PeriodicTask*[36] to update the tile once.

It is necessary to create a second project where the background task logic will be nested to actually use background tasks.

Note: Only the *OnInvoke(ScheduledTask task)* method in *ScheduledAgent.cs* needs to be overridden, the rest is auto-generated code from the ScheduledAgent project template.

References added:

LiveTileTaskAgent

Code Behind:

```
using Microsoft.Phone.Controls;
using Microsoft.Phone.Scheduler;
using Microsoft.Phone.Shell;
using System;
using System.Linq;

namespace LiveTile
{
    /// <summary>
    /// This application shows how to use Live Tiles using
    /// <see cref="Microsoft.Phone.Shell.ShellTile"/>. The
    /// application will create and add a new tile to the
    /// Windows Phone start screen and create a periodic task
    /// to update the tile once.
    /// It is necessary to create a second project where the
    /// background task logic will be nested to actually
    /// use background tasks. This application uses
    /// <see cref="LiveTileTaskAgent.ScheduledAgent"/>.
    /// </summary>
    /// <remarks>
    /// Bachelorthesis - "Media Programming on Mobile Devices (Windows Phone)"
    /// by Markus Besau
    /// July, 2013
    /// </remarks>
    public partial class MainPage : PhoneApplicationPage
    {
        public MainPage()
        {
            InitializeComponent();

            // create background agent to periodically update the live tile
```

35 http://msdn.microsoft.com/en-US/library/windowsphone/develop/microsoft.phone.shell.shelltile(v=vs.105).aspx

36 http://msdn.microsoft.com/en-US/library/windowsphone/develop/microsoft.phone.scheduler.periodictask(v=vs.105).aspx

```
            PeriodicTask task = new PeriodicTask("MediaUnderstandingTask");
            task.Description = "A simple periodic Task";
            task.ExpirationTime = DateTime.Now.AddDays(1);
             // task will expire in 1 day

            // check if task already exists and if so, remove it
            if (ScheduledActionService.Find(task.Name) != null)
                ScheduledActionService.Remove(task.Name);

            // add task
            ScheduledActionService.Add(task);

            // check if tile already exists
           ShellTile tile = ShellTile.ActiveTiles.FirstOrDefault(p =>
                    p.NavigationUri.ToString().Contains("TileID=2"));
            if (tile == null)
            {
                // create new tile
                StandardTileData newTile = new StandardTileData()
                {
                    // set tile properties
                    BackBackgroundImage = new Uri("Blue.jpg", UriKind.Relative),
                    Title = "Front",
                    Count = 1,
                    BackTitle = "Back",
                    BackContent = "Back content",
                    BackgroundImage = new Uri("Red.jpg", UriKind.Relative)
                };

                // add new tile
               ShellTile.Create(new Uri("/MainPage.xaml?TileID=2", UriKind.Relative),
                    newTile);
            }
        }
    }
}
```

ScheduledAgent.cs:

```
using Microsoft.Phone.Scheduler;
using Microsoft.Phone.Shell;
using System;
using System.Windows;
using System.Linq;

namespace LiveTileTaskAgent
{
    public class ScheduledAgent : ScheduledTaskAgent
    {
        private static volatile bool _classInitialized;

        /// <remarks>
        /// ScheduledAgent constructor, initializes the UnhandledException handler
        /// </remarks>
        public ScheduledAgent()
        {
            if (!_classInitialized)
            {
                _classInitialized = true;
                // Subscribe to the managed exception handler
```

```
                Deployment.Current.Dispatcher.BeginInvoke(delegate
                {
                      Application.Current.UnhandledException +=
                          ScheduledAgent_UnhandledException;
                });
            }
        }

        /// Code to execute on Unhandled Exceptions
        private void ScheduledAgent_UnhandledException(object sender,
                    ApplicationUnhandledExceptionEventArgs e)
        {
            if (System.Diagnostics.Debugger.IsAttached)
            {
                // An unhandled exception has occurred; break into the debugger
                System.Diagnostics.Debugger.Break();
            }
        }

        /// <summary>
        /// Agent that runs a scheduled task
        /// </summary>
        /// <param name="task">The invoked task</param>
        /// <remarks>
        /// This method is called when a periodic or
             resource intensive task is invoked
        /// </remarks>
        protected override void OnInvoke(ScheduledTask task)
        {
            if (task is PeriodicTask)
            {
                // get the applications tile
                ShellTile tile = ShellTile.ActiveTiles.FirstOrDefault(p =>
                          p.NavigationUri.ToString().Contains("TileID=2"));
                if (tile != null)
                {
                    // create a new tile with updated values
                    StandardTileData newTile = new StandardTileData()
                    {
                        Title = "updated by task",
                        Count = DateTime.Now.Minute,
                        BackgroundImage = new Uri("Green.jpg", UriKind.Relative)
                    };

                    // update the tile
                    tile.Update(newTile);
                }
            }
        }
    }
}
```