How to handle phone calls and other interruptions in Windows Phone

This example explains how to deal with an incoming phone call or other interruption (e.g. alarm, alert, popup) in a Windows Phone app.

Introduction

When an application is running you may receive some interruption - for example an incoming phone call - which will partially or completely obscure your application UI. The Obscured and Unobscured events allow developer to handle this scenario.

The `Obscured` and `Unobscured` event handlers are associated with the `PhoneApplicationFrame` API which is a part of **Microsoft.Phone.Controls** namespace. The `Obscured` event triggers when the shell chrome covers the frame: the `Unobscured` event triggers when the shell chrome is no longer covering the frame.

In this sample I have designed separate-separate UI to demonstrate each real time scenario.

The media player is loading...

Implementation

Step-1: Create the Windows Phone application using traditional way.

File -> **New Project** -> **Silverlight for Windows Phone** and choose “Windows Phone Application” Template. Here I have named the application “WmDev_CallHandling”.

Step-2: Design UI (User interface) of **MainPage.xaml** that contains the following buttons to navigate the each screen with specific case scenario.

- On Timer Test
- Alarm alert
- Reminder alert
- Story board
- Fm Radio
- Message Box

XAML code:

```
<Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
      <Button Content="on timer test" Height="72" HorizontalAlignment="Left"
Margin="85,62,0,0" Name="btnCall" VerticalAlignment="Top" Width="280"
Click="btnCall_Click" />
      <Button Content="alarms" Height="72" HorizontalAlignment="Left"
Margin="85,135,0,0" Name="btnAlarm" VerticalAlignment="Top" Width="280"
Click="btnAlarm_Click" />
      <Button Content="reminders" Height="72" HorizontalAlignment="Left"
Margin="85,212,0,0" Name="btnRiminder" VerticalAlignment="Top" Width="280"
Click="btnRiminder_Click" />
      <Button Content="storyboard" Height="72" HorizontalAlignment="Left"
Margin="85,293,0,0" Name="btnStoryboard" VerticalAlignment="Top" Width="280"
Click="btnStoryboard_Click" />
      <Button Content="message box" Height="72" HorizontalAlignment="Left"
Margin="85,449,0,0" Name="btnMsgBox" VerticalAlignment="Top" Width="280"
Click="btnMsgBox_Click" />
      <Button Content="fm player" Height="72" HorizontalAlignment="Left"
Margin="85,371,0,0" Name="btnFm" VerticalAlignment="Top" Width="280" Click="btnFm_Click"
/>
</Grid>
```


(C) Copyright Nokia 2014. All rights reserved.

Design View:

Design View

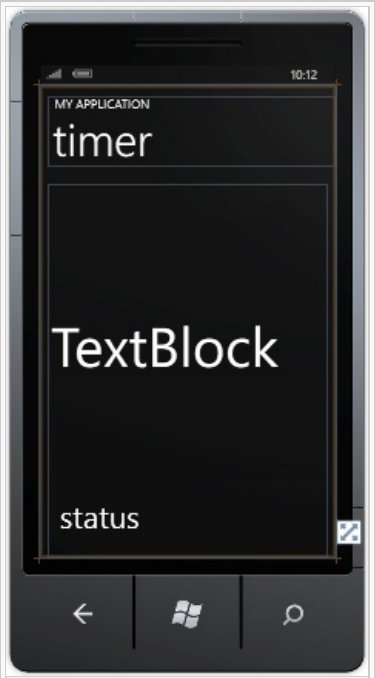
Step-3: In this explanation I have considered “On Timer Test” case scenario only when an incoming call arrival.

- Add a new page **IncomingCall.xaml** in the project.
- Add two Text Blocks for timer value and status display.

XAML code:

```
<Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
      <TextBlock Height="124" FontSize="90" HorizontalAlignment="Left"
Margin="6,200,0,0" Name="output" Text="TextBlock" VerticalAlignment="Top" Width="444" />
      <TextBlock Height="71" HorizontalAlignment="Left" Margin="20,506,0,0"
Name="txtStatus" Text="status" VerticalAlignment="Top" Width="404" FontSize="50" />
</Grid>
```

Design View:



Design View - To test the On Timer test case scenario

(C) Copyright Nokia 2014. All rights reserved.

Step-4: Add the `Obscured` and `Unobscured` event to `PhoneApplicationFrame` in the constructor of the **IncomingCall.xaml.cs**.

```
// private variable
DispatcherTimer oTimer = new DispatcherTimer();
CultureInfo oCultureInfo = CultureInfo.InvariantCulture;

public IncomingCall()
{
      InitializeComponent();

      this.oTimer.Interval = TimeSpan.FromMilliseconds(1);
      this.oTimer.Tick += new EventHandler(timer_Tick);
      this.oTimer.Start();

      PhoneApplicationFrame phoneAppRootFrame = (Application.Current as App).RootFrame;
      phoneAppRootFrame.Obscured += OnObscured;
      phoneAppRootFrame.Unobscured += Unobscured;
}
```

Step-5: Now in the `OnObscured` method and `Unobscured` method to handle the timer. When a phone call receive timer should stop and when the call completed or call ended that time timer should start from the resume state in the application or game. The `OnObscured` method will be occurred when phone call received and `Unobscured` event will be occurred when phone call ended. See the following code snippet for the same.

```
void OnObscured(Object sender, ObscuredEventArgs e)
{
     oTimer.Stop();
     txtStatus.Text = "Obscured event occurred";
}

void Unobscured(Object sender, EventArgs e)
{
     oTimer.Start();
     txtStatus.Text = "Unobscured event occurred";
}
```

Step-6: Now run the application and test it on the actual phone device for receiving incoming call scenario.

Real world interruption handling

If your application is whole or partially obscured it may make sense for it to be paused or stopped (this depends on the context). The code example uses the example of stopping and starting a timer - you might similarly stop or start games, audio, video where appropriate for your application.

Download sample code

Source code here: File:WmDev CallHandling.zip

Summary

This article shows how to catch the `Obscured` and `Unobscured` events so that you can modify your application to handle interruptions.

Reference

- PhoneApplicationFrame Events

(C) Copyright Nokia 2014. All rights reserved.

- Obscured Event
- Unobscured Event
- DispatcherTimer
- CultureInfo

(C) Copyright Nokia 2014. All rights reserved.