

A close-up of a Nokia Windows Phone Mango. The screen displays the classic Metro UI with large red tiles. The 'Phone' tile shows a white telephone handset icon and the number '2'. The 'People' tile shows a collage of three people's faces. The 'Messaging' tile shows a speech bubble icon with a face and the number '3'. The 'Linked Inbox' tile shows an envelope icon and the number '9'. At the bottom of the screen, there is a wide photo tile showing a woman's profile looking out at a beach with waves. The phone's black bezel and a silver speaker grille are visible at the top.

 Windows® Phone

Using Phone Resources

Rob S. Miles | Microsoft MVP | University of Hull, UK
Andy Wigley | Microsoft MVP | Appa Mundi

Session 7.0

NOKIA

Course Schedule

- Session 1 – Tuesday, August 23, 2011
 - Building Windows Phone Apps with Visual Studio 2010
 - Silverlight on Windows Phone—Introduction
 - Silverlight on Windows Phone—Advanced
 - Using Expression to Build Windows Phone Interfaces
 - Windows Phone Fast Application Switching
 - Windows Phone Multi-tasking & Background Tasks
 - **Using Windows Phone Resources (Bing Maps, Camera, etc.)**
- Session 2 – Wednesday, August 24, 2011
 - Application Data Storage on Windows Phone
 - Using Networks with Windows Phone
 - Tiles & Notifications on Windows Phone
 - XNA for Windows Phone
 - Selling a Windows Phone Application

NOKIA

Microsoft

Topics

- Using the Contacts and Calendars in Windows Phone
- Launchers and Choosers
- The Windows Phone Camera
 - Taking still images
 - Manipulating the video stream
- The Windows Phone Microphone
- The Windows Phone Sensors
 - The Motion sensor
- Video Content
- Bing Maps

NOKIA

Microsoft

Reading Contacts and Calendars

4

Your Obligations

- Before an application uses contact data in an application you must inform the user and request their permission
- You must also ensure that your application manages contact data in a secure way
- It would be completely unacceptable for your application to make a copy of the contact list and upload this or use it for spamming

NOKIA

5 *Microsoft*

Application Capabilities

```
<Capabilities>
  <Capability Name="ID_CAP_LOCATION"/>
  <Capability Name="ID_CAP_MEDIALIB"/>
  ... other capabilities here
  <Capability Name="ID_CAP_CONTACTS"/>
  <Capability Name="ID_CAP_APPOINTMENTS"/>
</Capabilities>
```

- Before an application can use the Contacts and Appointments data on a phone the user has to authorise this capability
- Applications identify their capabilities in the WMAppManifest.xml file which is part of the application project

NOKIA**6 Microsoft**

Contacts and Calendar Providers

- A Windows Phone application can read contact and calendar information from the different accounts
 - Windows Live
 - Exchange (Outlook, Google)
 - Facebook
 - Aggregated accounts (Twitter, LinkedIn etc)
- The precise abilities depend on the source of the information being used

NOKIA

7 Microsoft

Data Sources

Data provider	Contact name	Contact picture	Other contact data	Calendar appointments
Windows Phone Device	Yes	Yes	Yes	Yes
Windows Live Social	Yes	Yes	Yes	Yes
Windows Live Rolodex	Yes	Yes	Yes	Yes
Exchange Accounts (Contacts from local address book only, not Global Address List.)	Yes	Yes	Yes	Yes
Mobile Operator Address Book	Yes	Yes	Yes	No
Facebook	Yes	Yes	No	No
Windows Live Aggregated Networks (Twitter, LinkedIn, etc.)	No	No	No	No

NOKIA

8 Microsoft

Finding Data Sources

```
Account a;  
  
if (a.Kind == StorageKind.Outlook)  
{  
    // Outlook account  
}
```

- Each account has a property that allows an application to identify the source of that account
- The `StorageKind` enumeration has values for all the account types available on Windows Phone
- An application can get the `Account` from an `Appointment` instance to determine its source

NOKIA**Microsoft**

Reading Contacts

```
using Microsoft.Phone.UserData;
...
private void loadButton_Click(object sender,
                                RoutedEventArgs e)
{
    Contacts cons = new Contacts();
    cons.SearchCompleted += new EventHandler
        <ContactsSearchEventArgs>(cons_SearchCompleted);

    cons.SearchAsync(String.Empty, FilterKind.None,
        "Contacts Load");
}
```

- This starts off a contacts load request
- This is not a filtered search, all contacts are returned

NOKIA**Microsoft**

Displaying Contacts

```
void cons_SearchCompleted(object sender,  
                           ContactsSearchEventArgs e)  
{  
    try{  
        //Bind the results to the user interface.  
        ContactResultsData.DataContext = e.Results;  
    }  
    catch (System.Exception){  
    }  
}
```

- This runs when the search returns
- It uses data binding to display the results on the screen in a **ListBox**

NOKIA**Microsoft**

Data Binding

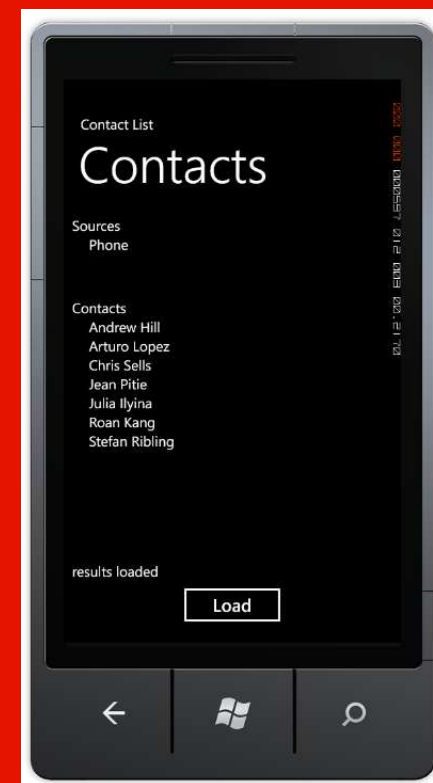
```
<ListBox Name="ContactResultsData"
        ItemsSource="{Binding}" Height="347"
        Margin="24,0,0,0" >
    <ListBox.ItemTemplate>
        <DataTemplate>
            <TextBlock Name="ContactResults"
                Text="{Binding Path=DisplayName,
                    Mode=OneWay}" />
        </DataTemplate>
    </ListBox.ItemTemplate>
</ListBox>
```

- This is the XAML that databinds the **DisplayName** property of the contact to a series of items in a list

NOKIA**Microsoft**

Demo

Demo 1: Contact Display



Filtering Searches

```
cons.SearchAsync("Ro", FilterKind.DisplayName,  
                "Contacts Load");  
// Find all the contacts with names beginning "Ro"
```

- By setting the `FilterKind` value on the search an application can apply a filter to the results
- There are a number of filters available
 - Display Name, Pinned to start, Email address, Phone number
- For advanced filtering a LINQ query could be used on the result enumeration
 - Although this might be a slow operation

NOKIA**Microsoft**

Displaying Appointments

```
apps.SearchAsync( new DateTime(2000, 1, 1),  
                  new DateTime(2012, 1, 1),  
                  "Appointment search");
```

- There is an analogous set of methods for finding appointments
- An application can search in a particular account type, for appointments in a particular time range
 - There is also a search method to search a particular account
- The reply is provided as a call back
- There are no appointments built into the emulator

NOKIA**15 Microsoft**

Creating a Contact

16

Creating a Contact

- An application cannot create a contact without the user being aware this action is being performed
 - This ensures that the user is always aware of what is going on when they are using the program
- To create a contact we use a “Chooser” which will display a confirmation message and give the user the option to create the contact or not
- The chooser is called asynchronously and will generate an event that tells the application whether the action succeeded or not



Launchers and Choosers

- A Chooser is used when an application wants to perform a task and get a response from the user
 - Select a contact or image
- On completion of the choice the application is resumed
- A Launcher called when application wants to perform a task
 - Visit a web page or place a phone call
 - The user can return to the application via the navigation stack
- In either case the application is made dormant/tombstoned while the task completes

NOKIA

18 Microsoft

Creating a Chooser

```
using Microsoft.Phone.Tasks;
...
private void MakeContactButton_Click(object sender,
                                     RoutedEventArgs e)
{
    SaveContactTask saveContact = new SaveContactTask();

    saveContact.FirstName = FirstNameTextBox.Text;
    saveContact.LastName = LastNameTextBox.Text;

    saveContact.Completed += new EventHandler
        <SaveContactResult>(saveContact_Completed);
    saveContact.Show();
}
```

- This creates a chooser to display the save contact dialog

NOKIA**Microsoft**

Chooser Completion

```
void saveContact_Completed(object sender,  
                           SaveContactResult e)  
{  
    if (e.TaskResult == TaskResult.OK)  
    {  
        MessageBox.Show("Saved OK");  
    }  
}
```

- This method is called when the chooser returns
- Some choosers return the result of selection the user has made
- This chooser returns whether the user saved the contact or not

NOKIA**Microsoft**

Application Switching

- When a chooser runs the application that invoked it will be deactivated
 - It will be made dormant or tombstoned
- It is important that it has the appropriate behaviours in place to deal with this
- There must be code to deal with Fast Application Switching to retain application state
- There is no guarantee that the user will come back from the chooser, they may launch other applications from within it

NOKIA**21Microsoft**

Demo

Demo 2: Make a Contact



Other Save Tasks

- There are a number of other savers available
- SaveEmailAddressTask
 - Save an email address
 - The address can be added to a contact
- SavePhoneNumberTask
 - Save a phone number
- SaveRingtoneTask
 - Save a sound file as a ringtone
- Note that there is no saver to create an appointment

NOKIA

23 Microsoft

Chooser Tasks

- Applications can create choosers to:
 - Get locations and routes from Bing Maps
 - Get addresses, phone numbers and email addresses
 - Select a picture from the media store
 - Capture a picture using the camera
 - Search the Marketplace and find applications
 - Invite players to a multi-game session

NOKIA

Microsoft

Launcher Tasks

- Applications can create launchers to:
 - Open a web page
 - Search the Marketplace and find applications
 - Place a phone call
 - Send an email
 - Send an SMS message
 - Share a url
 - Share a status message
 - Start a search using Bing

NOKIA

Microsoft

Using the Camera

26

The Windows Phone Camera

- The camera can be used in two ways
- The application can launch a chooser to allow the user to take a photograph
 - Can then use the image file in the application
- The application can stream video data from the camera
 - Can use this for product recognition or augmented reality

NOKIA

27 Microsoft

Capturing a photo

```
cameraTask = new CameraCaptureTask();  
  
cameraTask.Completed += new EventHandler<PhotoResult>  
    (cameraTask_Completed);  
  
cameraTask.Show();
```

- This task launches the camera so that the user can take a picture
- The `cameraTask_Completed` event fires when the picture is taken
- The emulator will return an image containing a white block

NOKIA**Microsoft**

Capture complete

```
void cameraCapture_Completed(object sender,
                              PhotoResult e)
{
    if (e.TaskResult == TaskResult.OK)
    {
        photoImage.Source = new BitmapImage(
                               new Uri(e.OriginalFileName));
    }
}
```

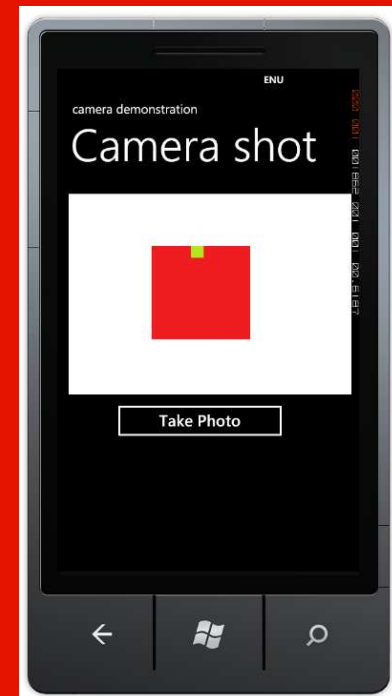
- When capture has been made, completed method is executed
- This version just displays the image on the screen
- You can access the photo stream data to save it

NOKIA**Microsoft**



Demo

Demo 3: Take a picture



In Application Capture

- It is also possible to capture an image from within your application
- An application can also display a viewfinder
- The application can also access the video data directly and use this for augmented reality or to create a customised viewfinder
 - eg. Barcode scanners
- The **PhotoCamera** class provides camera control and access to the camera video screen

NOKIA

31Microsoft

The PhotoCamera Class

```
using Microsoft.Devices;
...
PhotoCamera camera;
...
camera = new PhotoCamera();
//Set the VideoBrush source to the camera
viewfinderBrush.SetSource(camera);

camera.CaptureImageAvailable +=
    new EventHandler<ContentReadyEventArgs>
        (camera_CaptureImageAvailable);
```

- This creates a camera and binds a handler to the captured event
- The `viewfinderBrush` source is set to the camera

NOKIA**Microsoft**

Displaying the Viewfinder

```
<Rectangle Width="320" Height="240"  
            HorizontalAlignment="Left" >  
    <Rectangle.Fill>  
        <VideoBrush x:Name="viewfinderBrush" />  
    </Rectangle.Fill>  
</Rectangle>
```

- This is the rectangle in the xaml for the camera Silverlight page
- This will display the viewfinder on the screen
- The source for the viewfinder brush is set to the camera

NOKIA**Microsoft**

Initiating the Capture

```
private void PhotoButton_Click(object sender,  
                                RoutedEventArgs e)  
{  
    camera.CaptureImage();  
}
```

- This is the event handler for the photo button
- It asks the camera to take a picture
- The current camera settings are used for this
- You can override these (for example turn the flash on or off) by setting properties on the camera instance

NOKIA**Microsoft**

Saving the Image

```
using Microsoft.Xna.Framework.Media;
...
void camera_CaptureImageAvailable(object sender,
                                   ContentReadyEventArgs e)
{
    Deployment.Current.Dispatcher.BeginInvoke(delegate()
    {
        string fileName = DateTime.Now.Ticks.ToString()
                           + ".jpg";
        MediaLibrary library = new MediaLibrary();
        library.SavePictureToCameraRoll(fileName,
                                         e.ImageStream);
    });
}
```

- This saves the image in the camera roll

NOKIA**Microsoft**

Displaying the Image

```
using System.Windows.Media.Imaging;
...
void camera_CaptureImageAvailable(object sender,
                                   ContentReadyEventArgs e)
{
    Deployment.Current.Dispatcher.BeginInvoke(delegate()
    {
        BitmapImage b = new BitmapImage();
        b.CreateOptions = BitmapCreateOptions.None;
        b.SetSource(e.ImageStream);
        PictureImage.Source = b;
    });
}
```

- This saves the image in the camera roll

NOKIA**Microsoft**

Saving to Isolated Storage

```
using System.IO;
using System.IO.IsolatedStorage;
...

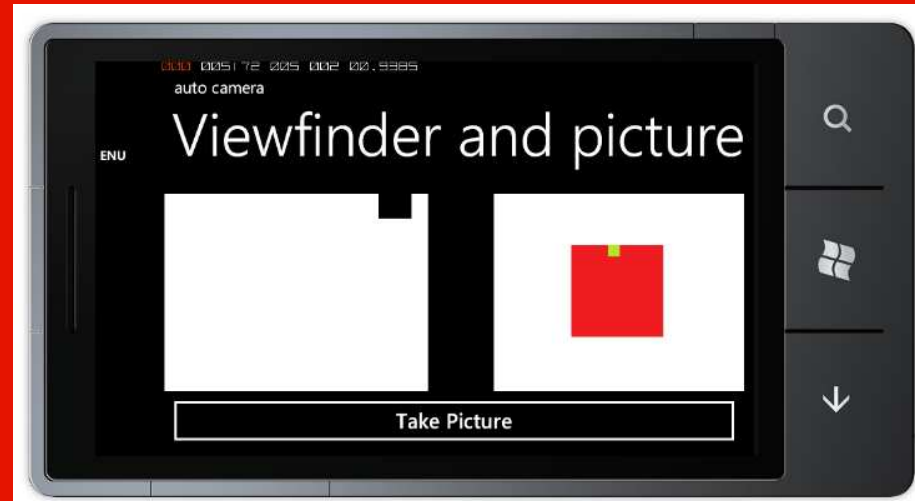
using (IsolatedStorageFile isStore =
    IsolatedStorageFile.GetUserStoreForApplication()) {
    using (IsolatedStorageFileStream targetStream =
        isStore.OpenFile(fileName, FileMode.Create,
            FileAccess.Write)) {
        WriteableBitmap bitmap = new WriteableBitmap(b);
        bitmap.SaveJpeg(targetStream, bitmap.PixelWidth,
            bitmap.PixelHeight, 0, 100);
    }
}
```

- This saves the image in isolated storage

NOKIA**Microsoft**



Demo



Demo 4: In application camera

Working with Image Data

- PhotoCamera class can also provide access to the video stream
- This can be used by applications to perform image processing on the live data
 - Augmented reality
 - Scanning applications
- We are going to use it to make a funky camera

NOKIA

39 Microsoft

Funky Image Processing

```
internal int FunkyColor(int color)
{
    int a = color >> 24;
    int r = (color & 0x00ff0000) >> 16;
    int g = (color & 0x0000ff00) >> 8;
    int b = (color & 0x000000ff);
    r += redOffset;
    g += greenOffset;
    b += blueOffset;
    return ((a & 0xFF) << 24) | ((r & 0xFF) << 16) |
           ((g & 0xFF) << 8) | (b & 0xFF);
}
```

- This separates out the primaries and adds an offset to each
- It is called for each pixel in the image

NOKIA**Microsoft**

Starting the Camera

```
camera = new Microsoft.Devices.PhotoCamera();

// Create the destination for the processed image
wb = new WriteableBitmap(640, 480);
this.ProcessedImage.Source = wb;

// Start the image pump when the camera is ready
camera.Initialized +=
    new EventHandler<CameraOperationCompletedEventArgs>
        (camera_Initialized);
```

- This creates the camera and the bitmap that will contain the image processed output
- It binds to the event fired when the camera is ready

NOKIA**Microsoft**

Starting the Camera

```
void camera_Initialized(object sender,  
                        CameraOperationCompletedEventArgs e)  
{  
    pumpARGBFrames = true;  
    ARGBFramesThread =  
        new System.Threading.Thread(PumpARGBFrames);  
    ARGBFramesThread.Start();  
}
```

- When the camera is ready we start the thread that will pump frames into our image processor
- This will run alongside our application

NOKIA**Microsoft**

Getting the Image Data

```
int[] ARGBPx = new int[640 * 480];  
...  
  
captureEvent.WaitOne();  
pauseFramesEvent.WaitOne(); //Thread sync with camera  
//Copies the current viewfinder frame into a buffer  
camera.GetPreviewBufferArgb32(ARGBPx);  
//Conversion to funky colours  
for (int i = 0; i < ARGBPx.Length; i++)  
{  
    ARGBPx[i] = FunkyColor(ARGBPx[i]);  
}
```

- This code grabs the preview buffer from the camera and processes it

NOKIA**Microsoft**

Drawing the Image Data

```
private WriteableBitmap wb;  
...  
pauseFramesEvent.Reset();  
Deployment.Current.Dispatcher.BeginInvoke(delegate()  
{  
    //Copy to WriteableBitmap  
    ARGBPx.CopyTo(wb.Pixels, 0);  
    wb.Invalidate();  
  
    pauseFramesEvent.Set();  
});
```

- This code writes the processed pixels back to a writeable bitmap that is displayed on the screen

NOKIA**Microsoft**

Drawing the Image Data

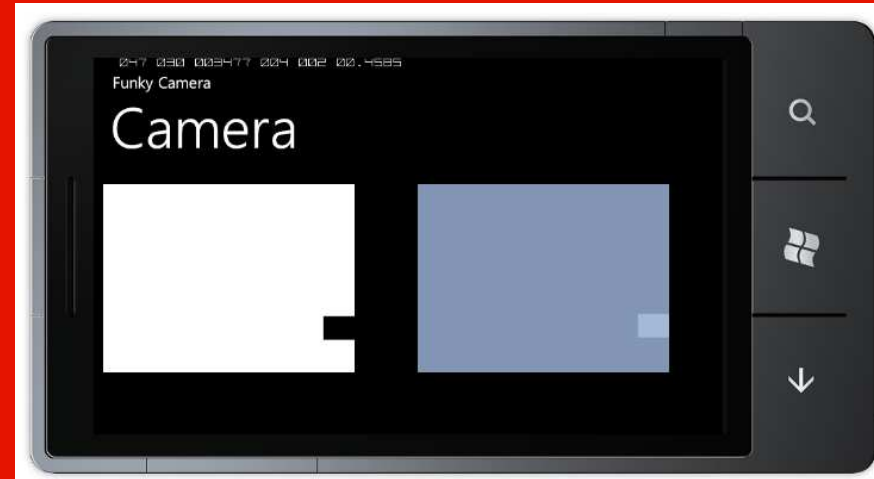
```
private WriteableBitmap wb;  
...  
pauseFramesEvent.Reset();  
Deployment.Current.Dispatcher.BeginInvoke(delegate()  
{  
    //Copy to WriteableBitmap  
    ARGBPx.CopyTo(wb.Pixels, 0);  
    wb.Invalidate();  
  
    pauseFramesEvent.Set();  
});
```

- This code writes the processed pixels back to a writeable bitmap that is displayed on the screen

NOKIA**Microsoft**



Demo



Demo 5: Funky camera

Using the Microphone

The Windows Phone Microphone

- The Windows Phone microphone can capture 16 bit audio
- Microphone input is managed as part of the XNA framework
- It is possible to record sound, process the audio and either store or replay it
- There is a complete example of how to do this on MSDN

<http://msdn.microsoft.com/en-us/library/gg442302.aspx>

NOKIA

Microsoft

Using Sensors

Sensors Available

- There are a number of different sensors:
 - Accelerometer
 - Compass
 - Gyroscope
- Not all the sensors are available on all phones
 - The Gyroscope is on Mango phones only
- All the sensors are used in the same way:
 - They will fire an event when they have a reading

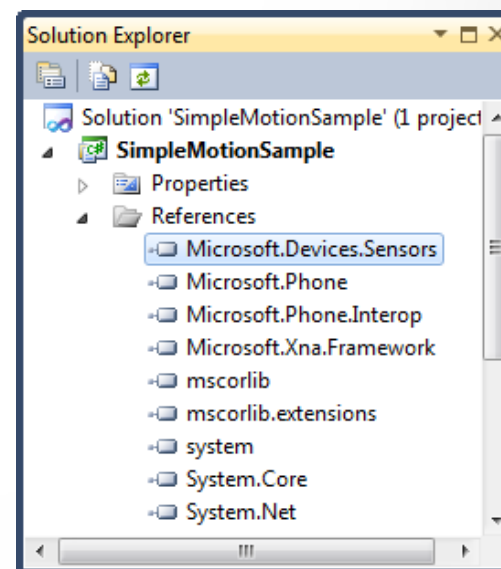
NOKIA

Microsoft

The Sensors Library

```
using Microsoft.Devices.Sensors;
```

- The motion sensors are in the Devices.Sensors library
- This must be added to the references of a project

**NOKIA****Microsoft**

Determining Sensor Availability

```
if (Gyroscope.IsSupported)
{
    // we have a gyro on the phone
}
```

- All the sensors expose an `IsSupported` property that can be used to test if they are on a particular device
- If an application tries to use a sensor which is not available it will throw an exception
 - Test before you try to use a device
- All phones will have an accelerometer

NOKIA**Microsoft**

The Motion Sensor

- The Motion Sensor brings together the readings from all the motion sensors in the phone
 - Accelerometer
 - Compass
 - Gyroscope (where fitted)
- It provides a “one stop shop” for applications that want to use device orientation

NOKIA

Microsoft

Motion Data

- The Motion sensor can provide:
 - Attitude
 - Pitch, Yaw and Roll
 - Rotation rate
 - Direction of Gravity
- It will use whatever sensors are available
 - It only works if the phone has a working Compass
- It can make it easier to create augmented reality applications and games

NOKIA

54 Microsoft

Using a Sensor

```
using Microsoft.Devices.Sensors;
...
Motion motion;
...
motion = new Motion();
motion.TimeBetweenUpdates =
    TimeSpan.FromMilliseconds(20);
motion.CurrentValueChanged += new EventHandler
    <SensorReadingEventArgs<MotionReading>>
    (motion_CurrentValueChanged);
```

- This code creates a motion sensor and binds a method to the changed event
- It also sets the time between updates for the sensor

NOKIA**Microsoft**

Starting the Motion Sensor

```
// Try to start the Motion API.  
try  
{  
    motion.Start();  
}  
catch (Exception)  
{  
    MessageBox.Show("unable to start the Motion API.");  
}
```

- This code starts the sensor running
- Note that it can throw an exception, which an application should catch and deal with

NOKIA**Microsoft**

Using the Motion Sensor

```
private void CurrentValueChanged(MotionReading e)
{
    // Update the display using the new motion readings
}
```

- When the `CurrentValueChanged` event fires the program can update the display accordingly
- Note that in an XNA game this will involve setting values that will be picked up by the `Update` method and used to update the game model

NOKIA**Microsoft**

Video Content

Video on the Phone

```
<MediaElement Name= "MediaPlayerback"  
                Source= "myvideo.wmv" AutoPlay="True"/>
```

- A Silverlight application can contain a single MediaElement that can play video
- The sample above plays a resource file that is part of the Silverlight project containing the application
- You can find a list of supported codecs here:

<http://msdn.microsoft.com/en-us/library/ff462087.aspx>

NOKIA

Microsoft

Streaming Video on the Phone

```
<MediaElement Name= "MediaPlayer"  
Source=  
"http://mschannel9.vo.msecnd.net/o9/mix/09/wmv/key01.wmv"  
AutoPlay="True"/>
```

- If you want to stream from the internet, just replace the source file with a url
- This implementation has the location hard coded into the XAML
- You can also do this under program control

NOKIA**Microsoft**

Controlling Playback

```
private void pauseButton_Click(object sender,  
                                RoutedEventArgs e)  
{  
    MediaPlayer.Pause();  
}
```

- The MediaElement exposes methods that can be used to control the media playback
- An application can also determine the properties of the media stream
 - To determine if it can be paused for example

NOKIA**Microsoft**

Smooth Streaming

- Windows Phone also supports Smooth Streaming
 - This is an adaptive streaming that manages the quality of the video signal in response to the abilities of the network connection
 - Playback quality is managed in real time to handle changes in network performance during viewing
- Uses server-side plugin and client code on the viewing device
- You can download the viewing software here:
<http://smf.codeplex.com/releases/view/63434#DownloadId=222617>

NOKIA

Microsoft

Bing Maps

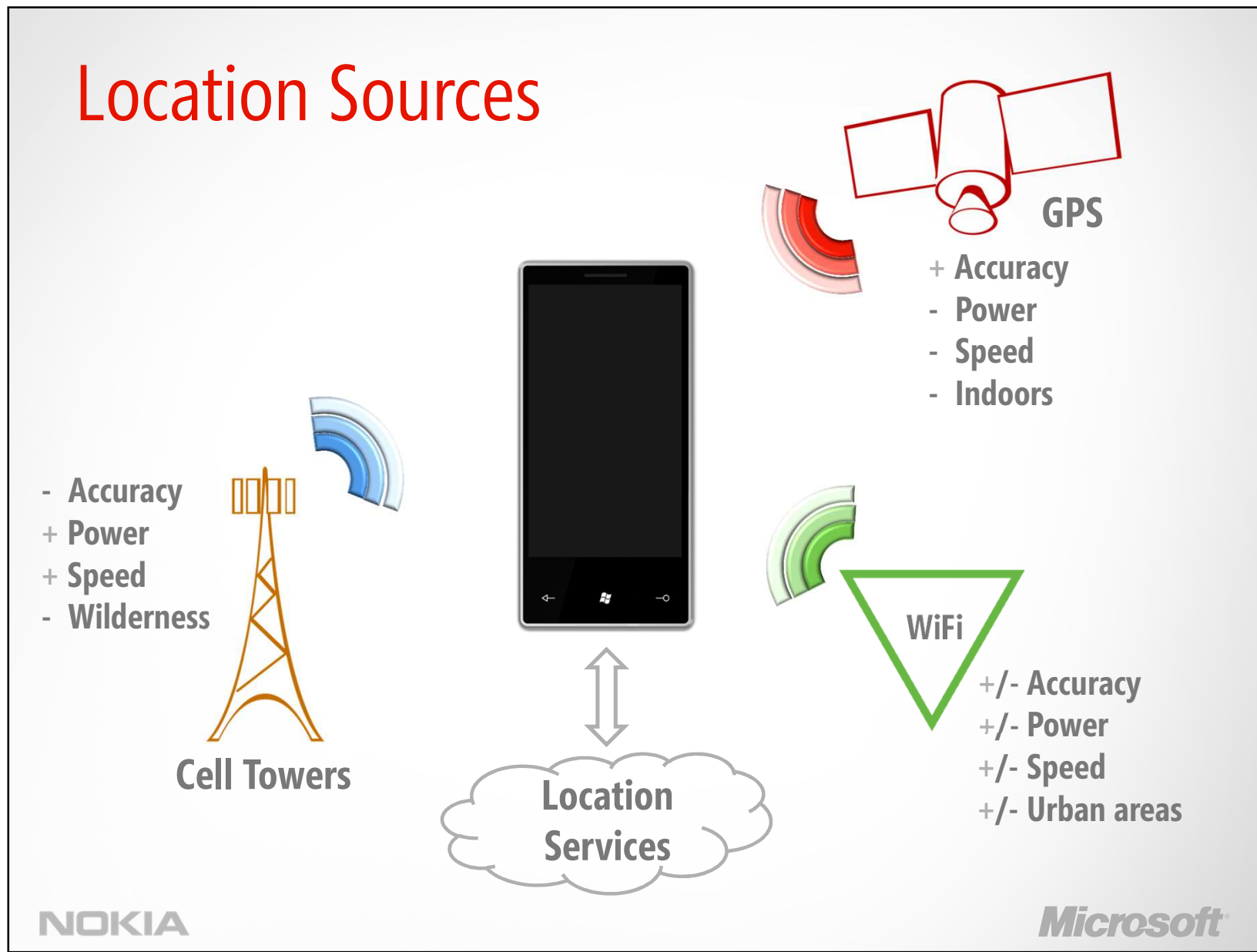
63

Features

- Smart device + cloud service
- Balance accuracy with time & power
- Consistent API across platforms

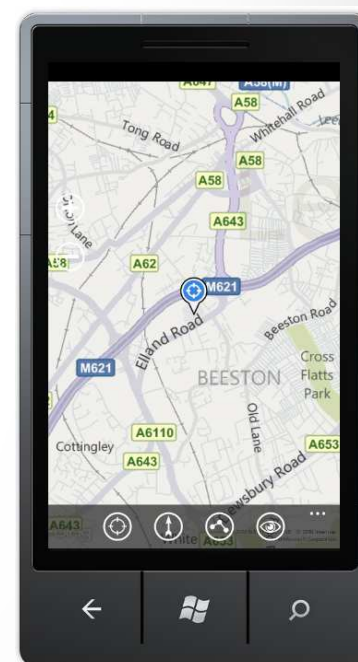
NOKIA

64 Microsoft



Silverlight Bing Map Control

- Optimize for Windows Phone
 - Touch Enabled
 - Pinch – zoom in/out
 - Translate
 - Caching
- Full parity with Silverlight Bing Map control
 - <http://www.microsoft.com/maps/isdk/Silverlight/>

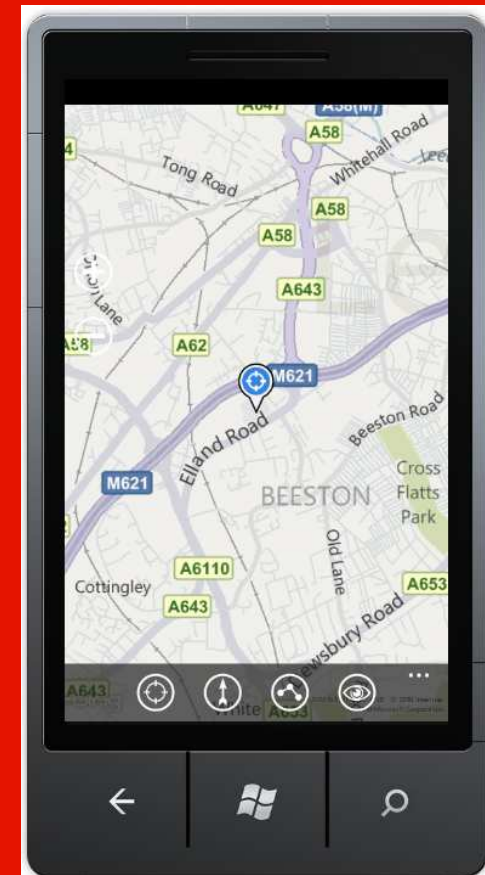


NOKIA

Microsoft

Demo

Silverlight Maps control



67

New Bing Maps Launchers in Windows Phone 7.5

BingMapsTask

- BingMapsTask makes launching the built-in Bing Maps application easy

```
Display.Click += ( o, e ) =>
{
    var longitude = Double.Parse(Long.Text);
    var latitude = Double.Parse(Lat.Text);

    var bingMapsTask = new BingMapsTask()
    {
        Center = new GeoCoordinate( longitude, latitude ),
        SearchTerm = Search.Text,
        ZoomLevel = Double.Parse( Zoom.Text )
    };

    bingMapsTask.Show();
};
```

NOKIA**Microsoft**

BingMapsDirectionsTask

- Launching built-in Bing Maps tasks with directions enabled is trivial too!

```
// create the task  
var task = new BingMapsDirectionsTask();  
task.Start = new LabeledMapLocation(startLabel, startCoordinate);  
task.End = new LabeledMapLocation(endLabel, endCoordinate);  
task.Show();
```

- If you do not specify one of Start or End parameters it defaults missing parameter to your current location
- Can also specify only the name of a location (but no coordinate) and it will automatically do a geocoding service lookup and display a selection list if multiple matches



Demo

BingMapsTask and BingMapsDirectionsTask



71

Review

- Applications can use phone resources by means of Launchers and Choosers
- Launchers start a behaviour, choosers can return a result
- Launchers and Choosers interrupt the running of the application
- Applications can capture images and video feeds from the camera
- Applications can use sensors and determine if they are present
- The MediaContent element provides for video playback
- Bing Maps can be used as a task or a program control

NOKIA

72 Microsoft



The information herein is for informational purposes only and represents the current view of Microsoft Corporation as of the date of this presentation. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information provided after the date of this presentation.

MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS PRESENTATION.

© 2011 Microsoft Corporation. All rights reserved.

Microsoft, Windows, Windows Vista and other product names are or may be registered trademarks and/or trademarks in the U.S. and/or other countries.

Microsoft