Accelerometer

Introduction

- The accelerometer sensor measures acceleration forces such as gravity and the forces caused by moving the mobile telephone.
- The Windows Phone has one accelerometer sensor at the minimum.

The `AccelerometerReading` Class

- The accelerometer data is delivered to our application through the `AccelerometerReading` class.
- The AccelerometerReading class includes the `X`, `Y` and `Z` properties. Each one of them provides a value in between -1 and 1. These three values indicate the direction of the acceleration for each axis.

The `Microsoft.Devices.Sensors` Assembly

- This assembly includes the sensor framework. In order to develop an application that uses the sensor framework we must add to our project a reference to this assembly.
- Once the reference to this assembly was added to our project we can add the following using statement into our code.

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

The `AccelerometerSensor` Class

- This is the main class we will use to access the accelerometer sensor.

```
...
AccelerometerSensor sensor = new AccelerometerSensor();
...
```

The `Portrait` and `Landscape` Modes

❖ The default template generated by the Visual Studio supports both the portrait and the landscape modes. When the user changes the phone position it automatically changes the display between these two modes.

```
...
public MainPage()
{
    InitializeComponent();
    SupportedOrientations = SupportedPageOrientation.Portrait
          | SupportedPageOrientation.Landscape;
}
...
```

The `Portrait` and `Landscape` Modes

- When using the accelerometer sensor this automatic behavior can be distracting. When using the accelerometer sensor it is highly recommended to change the application to support one layout mode only.

```
...
public MainPage()
{
    InitializeComponent();
    SupportedOrientations = SupportedPageOrientation.Portrait;
}
...
```

The `ReadingChanged` Event

- This event is raised when the accelerometer sensor has a new sensor reading.

```
...
public MainPage()
{
    InitializeComponent();
    AccelerometerSensor sensor = new AccelerometerSensor();
    sensor.ReadingChanged += new
        EventHandler<AccelerometerReadingAsyncEventArgs>(MyMethodA);
}
...
```

The `Start()` Method

- We start the accelerometer by calling the `Start()` method.

```
...
try
{
    accelerometer.Start();
}
catch (AccelerometerStartFailedException ex)
{
    ...
}
...
```

The `Stop()` Method

- We stop the accelerometer by calling the `Stop()` method.

```
...
try
{
    accelerometer.Stop();
}
catch (AccelerometerStopFailedException ex)
{
    ...
}
...
```

Threads Issues

- The event handler is called from another thread. It isn't executed from the thread that draws our page.
- Using `Deployment.Current.Dispatcher` we can indirectly call another method and have it executed within the thread that draws our page.

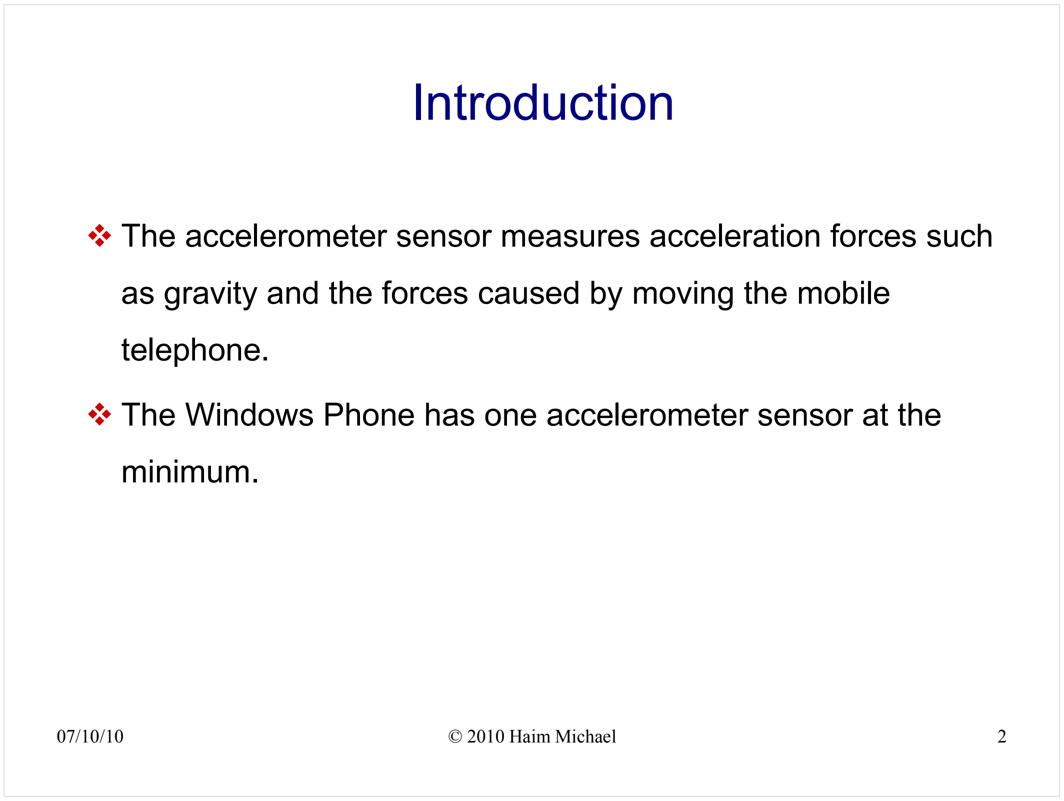
```
...
void MyMethodA(object sender, AccelerometerReadingAsyncEventArgs e)
{
    Deployment.Current.Dispatcher.BeginInvoke(() => MyMethodB(e));
}
...
```

Threads Issues

- Within the second method we will be able to access the sensor data during the execution of the thread that draws our page and update the user interface accordingly.

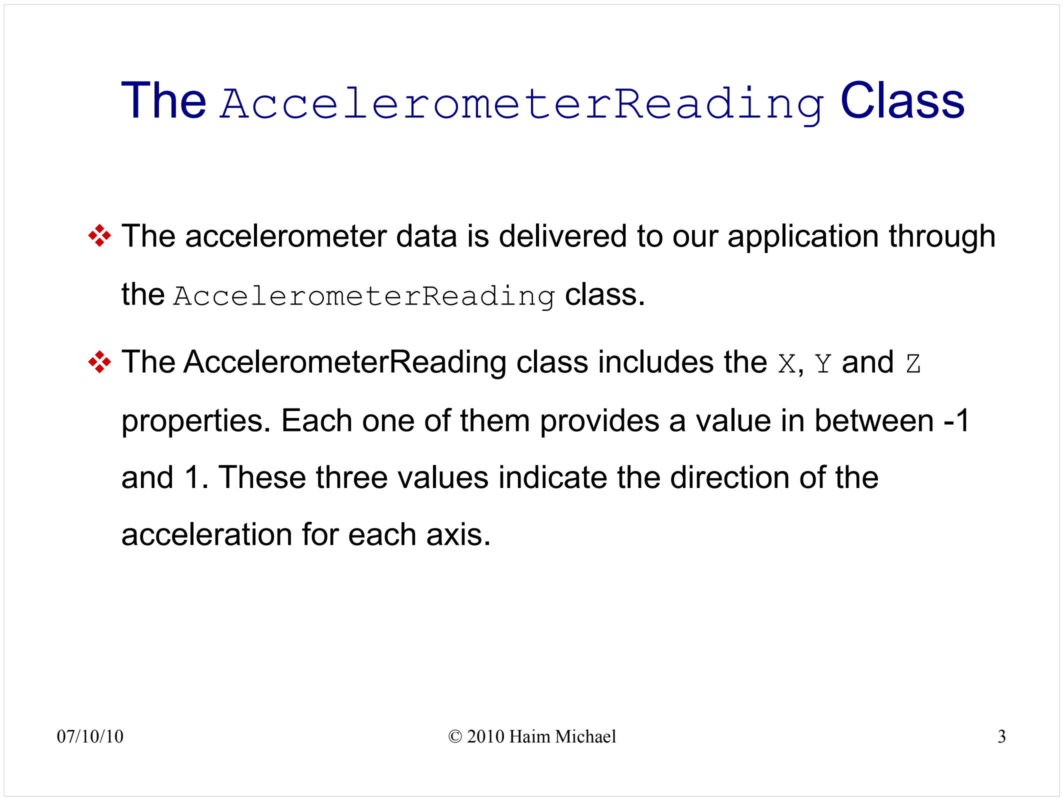
```
...
void MyMethodB(object sender, AccelerometerReadingAsyncEventArgs e)
{
    double x = e.Value.Value.X;
    double y = e.Value.Value.Y;
    double z = e.Value.Value.Z;
    ...
}
...
```

Accelerometer



Introduction

- The accelerometer sensor measures acceleration forces such as gravity and the forces caused by moving the mobile telephone.
- The Windows Phone has one accelerometer sensor at the minimum.



The `AccelerometerReading` Class

- The accelerometer data is delivered to our application through the `AccelerometerReading` class.
- The AccelerometerReading class includes the `X`, `Y` and `Z` properties. Each one of them provides a value in between -1 and 1. These three values indicate the direction of the acceleration for each axis.

07/10/10 © 2010 Haim Michael 3

The `Microsoft.Devices.Sensors` Assembly

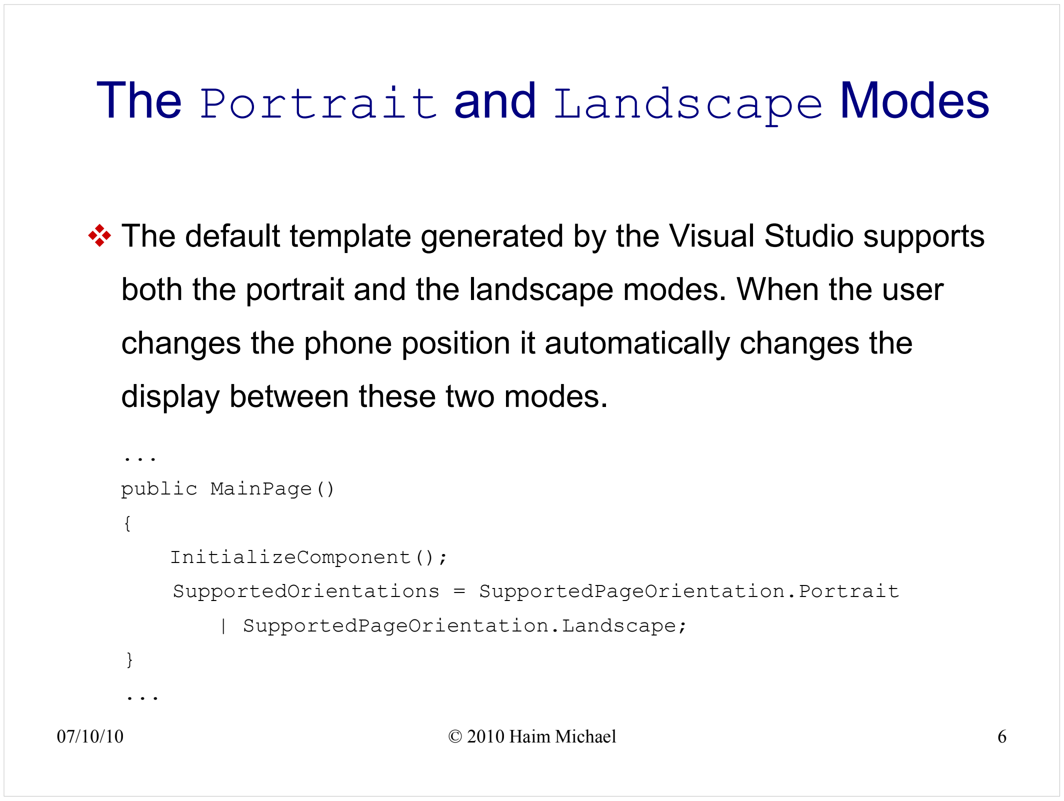
- This assembly includes the sensor framework. In order to develop an application that uses the sensor framework we must add to our project a reference to this assembly.
- Once the reference to this assembly was added to our project we can add the following using statement into our code.

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

The `AccelerometerSensor` Class

- This is the main class we will use to access the accelerometer sensor.

```
...
AccelerometerSensor sensor = new AccelerometerSensor();
...
```



The `Portrait` and `Landscape` Modes

- The default template generated by the Visual Studio supports both the portrait and the landscape modes. When the user changes the phone position it automatically changes the display between these two modes.

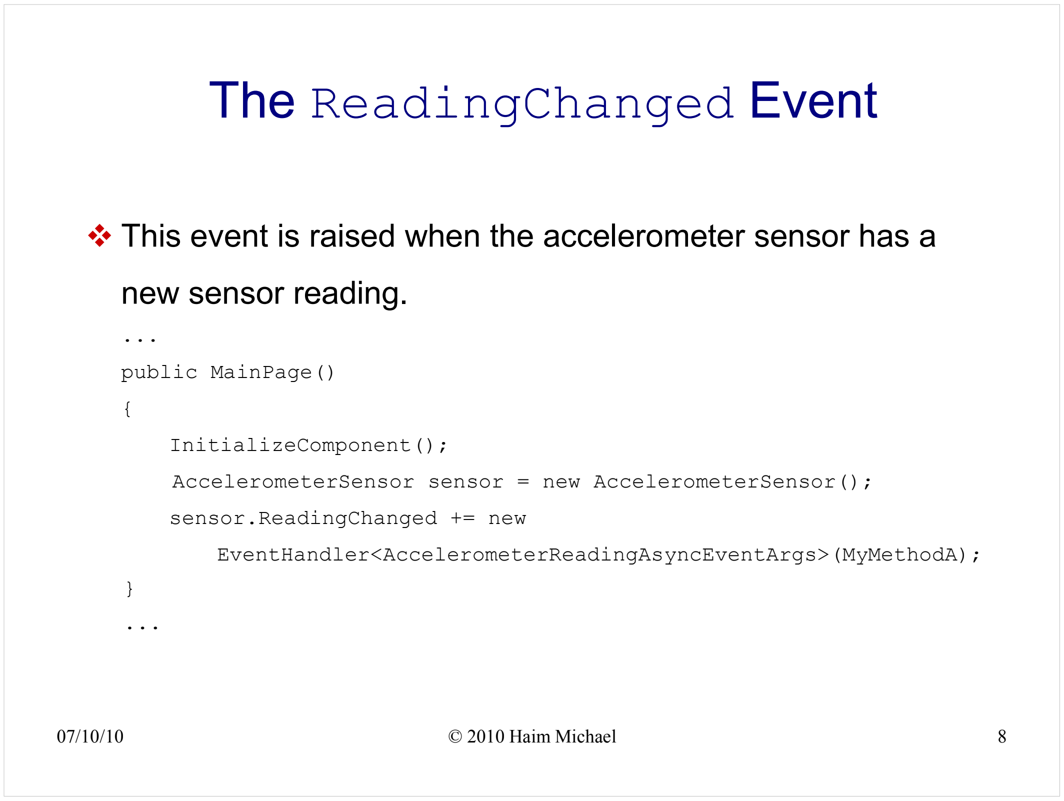
```
...
public MainPage()
{
    InitializeComponent();
    SupportedOrientations = SupportedPageOrientation.Portrait
        | SupportedPageOrientation.Landscape;
}
...
```

07/10/10 © 2010 Haim Michael 6

The `Portrait` and `Landscape` Modes

- When using the accelerometer sensor this automatic behavior can be distracting. When using the accelerometer sensor it is highly recommended to change the application to support one layout mode only.

```
...
public MainPage()
{
    InitializeComponent();
    SupportedOrientations = SupportedPageOrientation.Portrait;
}
...
```



The `ReadingChanged` Event

- This event is raised when the accelerometer sensor has a new sensor reading.

```
...
public MainPage()
{
    InitializeComponent();
    AccelerometerSensor sensor = new AccelerometerSensor();
    sensor.ReadingChanged += new
        EventHandler<AccelerometerReadingAsyncEventArgs>(MyMethodA);
}
...
```

The `Start()` Method

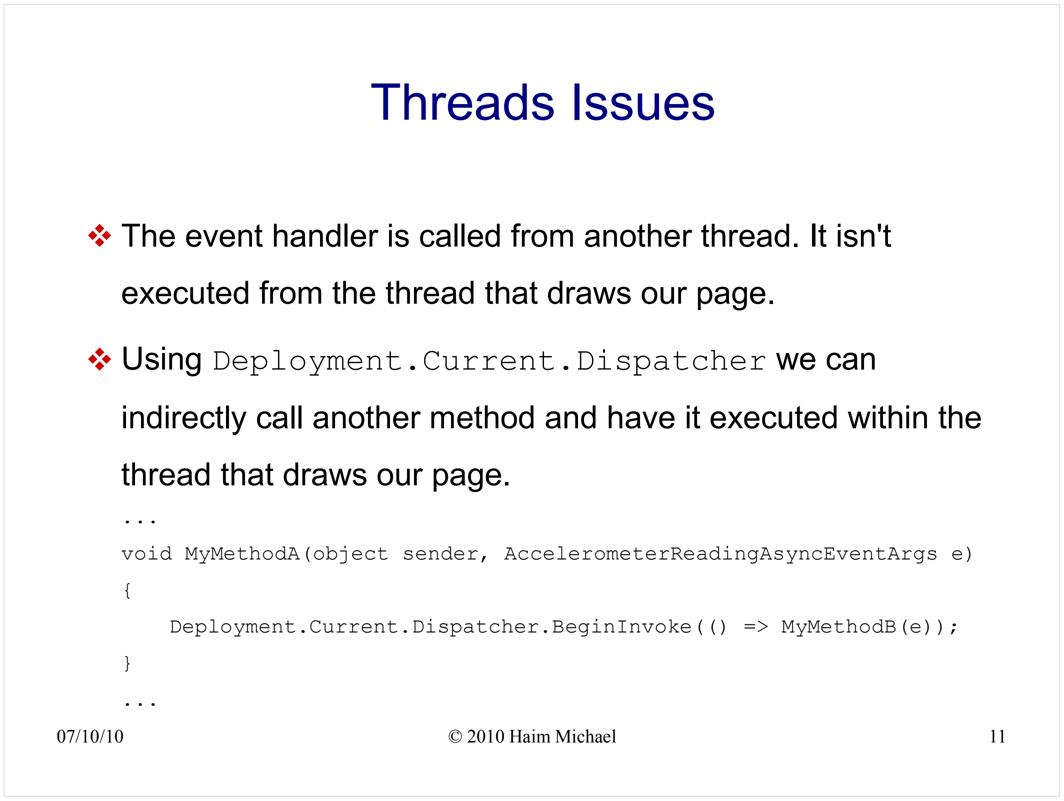
- We start the accelerometer by calling the `Start()` method.

```
...
try
{
    accelerometer.Start();
}
catch (AccelerometerStartFailedException ex)
{
    ...
}
...
```

The `Stop()` Method

- We stop the accelerometer by calling the `Stop()` method.

```
...
try
{
    accelerometer.Stop();
}
catch (AccelerometerStopFailedException ex)
{
    ...
}
...
```



Threads Issues

- The event handler is called from another thread. It isn't executed from the thread that draws our page.
- Using `Deployment.Current.Dispatcher` we can indirectly call another method and have it executed within the thread that draws our page.

```
...
void MyMethodA(object sender, AccelerometerReadingAsyncEventArgs e)
{
    Deployment.Current.Dispatcher.BeginInvoke(() => MyMethodB(e));
}
...
```

07/10/10 © 2010 Haim Michael 11

Threads Issues

- Within the second method we will be able to access the sensor data during the execution of the thread that draws our page and update the user interface accordingly.

```
...
void MyMethodB(object sender, AccelerometerReadingAsyncEventArgs e)
{
    double x = e.Value.Value.X;
    double y = e.Value.Value.Y;
    double z = e.Value.Value.Z;
    ...
}
...
```