Creating Compass in Qt and WP7

This article demonstrates how to create a Compass in Qt and WP7.

Introduction

In this article we will create a Compass in QML using QML Compass Element which is a part of QtMobility 1.x and for WP7 we will use Compass Class. This is a magnetic compass used to determine the direction of the earth's magnetic north pole.

Compass using QML

Implementation

Let's create an empty project for both Qt and WP7. First we will work on Qt project and then will move on to WP7 project.

Qt Project (MainPage.qml)

Let's first create a circle using Rectangle Element as shown in the above figure. This circle will determine the compass outline. In QML Rectangle can be used to draw a circle with 1:1 dimensions and a radius.

```
Rectangle
{
 id:circle
 width: 250
 height: 250
 color: "transparent"
 border.color: "green"
 border.width: 3
 radius: width*0.5
 anchors.centerIn: parent
}
```

We add an Image to create the compass needle and rotate the Image with its change in degrees (angle).

```
 Image {
 id: arrowImage
 anchors.bottom: circle.verticalCenter
 anchors.horizontalCenter: circle.horizontalCenter
 height: circle.paintedHeight / 2
 fillMode: Image.PreserveAspectFit
 source: "upload.png"
 rotation: -needleRotation
 transformOrigin: Item.Bottom
}
```

QML Compass Element is used to get the change in magnetic north reading in the `Compass`.

```
 Compass {
 id: compass
 active: true
 onReadingChanged: {
    timerCompassCheck.running= false; //disable timer because Compass is working
    mainPage.needleRotation = reading.azimuth;
 }
}
```


(C) Copyright Nokia 2013. All rights reserved.

We also set a Timer which expires after 7 seconds to check whether the compass is working properly (for example, it may not be supported on device).

```
Timer {
 id: timerCompassCheck
 running: true
 repeat: false
 interval: 7000
 onTriggered: {
  compassTextError.visible=true;
     }
}
```

As soon as the application launches the Timer starts running and expires after 7 seconds, the Timer doesn't repeat as the *repeat* property is set to false. When the `onReadingChanged` signal of the Compass Element is called it first stops the Timer and gets the compass reading and set it to `needleRotation` property which then used to rotate the compass needle angle. If within 7 seconds the signal `onReadingChanged` of the Compass Element is not called, then the Timer expires and displays a message that the device doesn't support compass. A Text element is being added to display the magnetic north value of the compass change on the device screen.

```
Text {
        id:needleRotationText
        text: needleRotation + " °N"
        color: platformStyle.colorNormalLight
        font.pixelSize: 35
        anchors.top: parent.top
        anchors.topMargin: 60
        anchors.right: parent.right
        anchors.rightMargin: 30
    }
```

WP7 Project (MainPage.xaml)

To build a similar interface like Qt, let's first create a circle using Ellipse Class which defines the outline of the compass.

```
<Ellipse StrokeThickness="2" x:Name="circle" Width="400" Height="400"
                 Margin="0,90,0,0" Fill="Black">
                <Ellipse.Stroke>
                    <SolidColorBrush Color="{StaticResource PhoneAccentColor}" />
                </Ellipse.Stroke>
</Ellipse>
```

To create the compass needle we used Line Class.

```
<Line x:Name="line" X1="240" Y1="350" X2="240" Y2="270"  Stroke="{StaticResource PhoneForegroundBrush}" StrokeThickness="4"></Line>
```

Now we add the reference `Microsoft.Devices.Sensors` to initialize and detect the compass presence in the device.

```
Compass compass;
if (Compass.IsSupported)
            {
                compass = new Compass();
                compass.TimeBetweenUpdates = TimeSpan. FromSeconds(1);
                compass.CurrentValueChanged += new EventHandler<SensorReadingEventArgs<CompassReading>>(compass_CurrentValueChanged)
                compass.Start();
            }
            else
            {
                MessageBox.Show("Device doesn't suport compass");
            }
```

`TimeBetweenUpdates` property will update the compass data in every second. `CurrentValueChanged` event handler will give the compass value each time there is any change in the compass needle position. And then we call the `Start()` method to start the compass. The change in compass value is then passed to the `compass_CurrentValueChanged()` method. Which then pass to a separate thread using `Dispatcher.BeginInvoke()` method.

```
void compass_CurrentValueChanged(object sender, SensorReadingEventArgs<CompassReading> e)
        {
            Dispatcher.BeginInvoke(() => UpdateLine(e.SensorReading));
        }
```

```
private void UpdateLine(CompassReading e)
        {
            degreeText.Text = e.MagneticHeading.ToString("0");
            line.X2 = line.X1 - (200 * Math.Sin(MathHelper.ToRadians((float)e.MagneticHeading)));
            line.Y2 = line.Y1 - (200 * Math.Cos(MathHelper.ToRadians((float)e.MagneticHeading)));
        }
```

The `MagneticHeading` gives the magnetic north pole value of the compass and is displayed in a TextBlock. We also add `Microsoft.Xna.Framework` reference into our project to find out the needle direction using `MathHelper` class.

Source Code

- The full source code of Qt example is available here: File:CompassQML v1.3.zip
- The full source code of WP7 example is available here: File:CompassWp7.zip

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.