Gauges – Simple Gauge Control in WPF and Silverlight

Content

This document describes how to use LightningChart Gauges to make different kind of measurements and databinding with a MS Visual Studio/Blend based on XAML and .NET Framework in WPF and Silverlight projects.

Installation process is described in LightningChart Gauge User's manual.

Prerequisites

Basic knowledge of C# and/or XAML, Visual Studio 2010 or newer is assumed.

LightningChart Gauges v.1.0 edition is installed and demo projects are found to be functional.

Arction Toolbox is installed in Visual Studio manually or automatically during the LightningChart Gauge installation.

MS Visual Studio/Blend 2013 or newer is required for editing the project due requirement of .NET Framework 4 or newer profile, and last version of Silverlight.

Description

Gauge allows presenting a wide variety of data using plenty of scales that makes able to have several data readings at the same time.

Add LightningChart Gauges control and design an application

Create a WPF or Silverlight application, which creates and opens main window automatically. Set Height and Width properties to "280" both by changing settings in Layout property tree or coding `Height="280" Width="280"`.

Drag and drop component "Arction" from the Visual Studio toolbox to the form in window editor and set Width and Height to auto, the gauge will bind the size of the window.

The window editor should look like this:

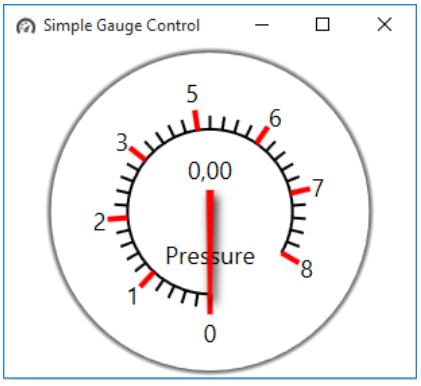

Figure 1. Window

The source as text in WPF:

```
<Window
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:agc="http://www.arction.com/gauges/"
        xmlns:Shaders="clr-
namespace:Arction.Shaders;assembly=Arction.WPF.Gauges"
        x:Class="WpfGaugeTutorial.MainWindow"
        Title="Simple Gauge Control"
        Height="280" Width="280">
    <Grid>
        <agc:Gauge/>
    </Grid>
</Window>
```

The source as text in Silverlight:

```
<UserControl
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:agc="http://www.arction.com/gauges/"
x:Class="SilverlightTutorial.MainPage"
    mc:Ignorable="d"
    d:DesignHeight="280" d:DesignWidth="280">
    <Grid>
        <agc:Gauge/>
    </Grid>
</UserControl>
```

Next step is adding some controllers to simulate data gathering from a sensor. This controller will be a slider. We have already defined that the gauge will scaled according to the window size, thus our application will be fully scalable. For that purpose, modify the XAML designer code by adding

```
<Grid.RowDefinitions>
    <RowDefinition Height="something"></RowDefinition>
    <RowDefinition Height="something"></RowDefinition>
</Grid.RowDefinitions>
```

and drop a slider to the designer, then set the slider to the row with index 1. The gauge will be in a row with index 0 automatically.

```
<Slider Grid.Row="1" Margin="10,0" />
```

Set the Data Binding for the Slider and Gauge. Slider will bind the RangeEnd and RangeBegin from the gauge, at the same time it will get the value from the slider and a needle will show it.

```
<agc:Gauge x:Name="gauge" Width="Auto" Height="Auto">
    <agc:Gauge.PrimaryScale>
       <agc:Scale Value="{Binding Value, ElementName=slider}"/>
    </agc:Gauge.PrimaryScale>
</agc:Gauge>
<Slider x:Name="slider" Grid.Row="1"
        Margin="10,0"
        Maximum="{Binding PrimaryScale.RangeEnd, ElementName=gauge}"
        Minimum="{Binding PrimaryScale.RangeBegin, ElementName=gauge}"/>
```

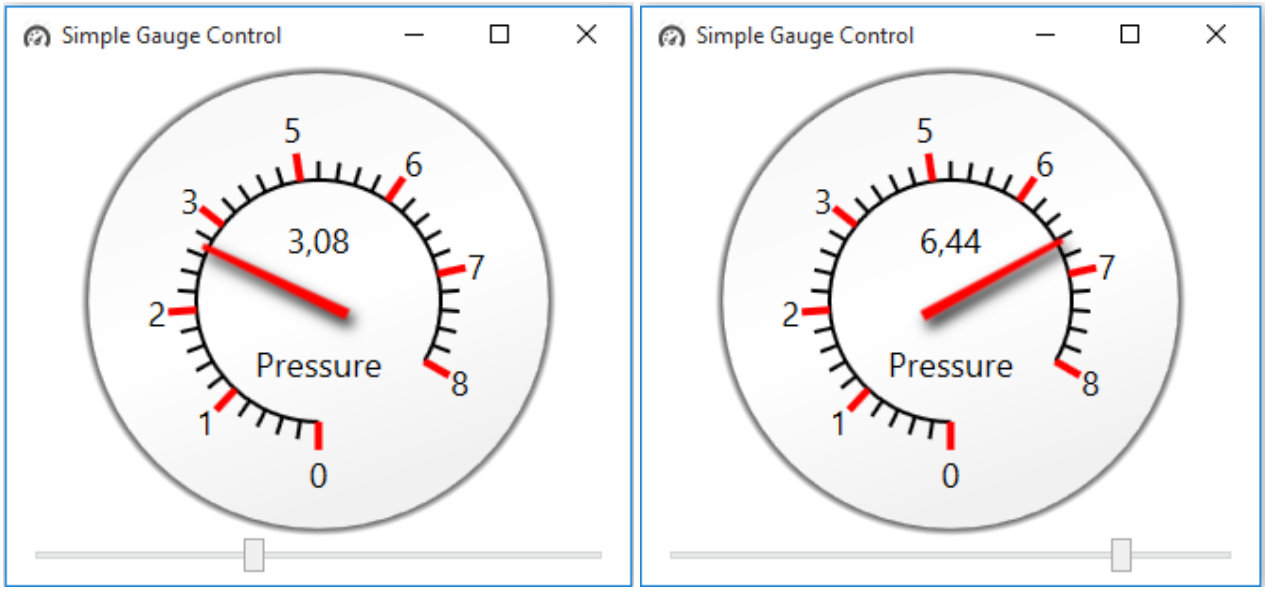

Figure 2. Examples of Value Biding between a Slider and Gauge

Converters in LightningChart Gauge

Here we can modify a little bit our gauge, which will be able to gather data from a sensor and show it in two scales, first scale is PrimaryScale (Bars) and SecondaryScale(Atm) for that purpose use IValueConverter during the data binding process.

The gauge will present both values at the same time.

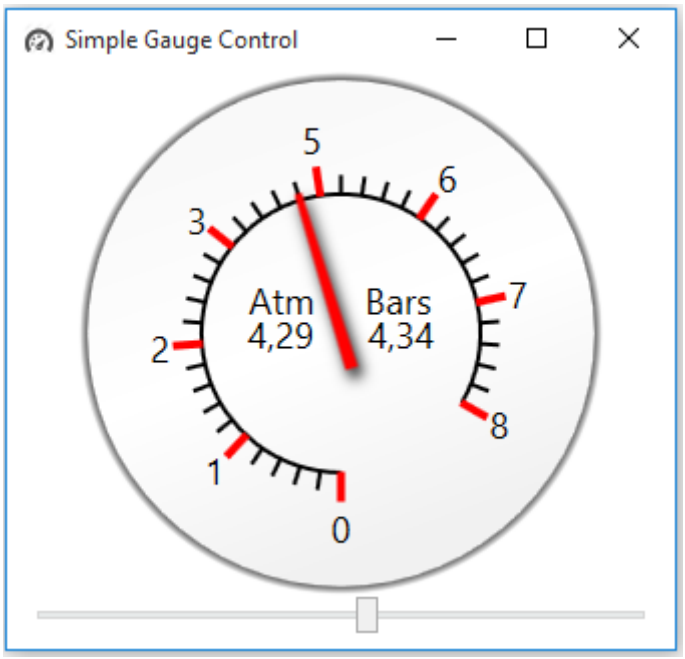

Figure 3. Gauge with two scales measures two readings at the same time

Create new class ***BarsToAtmosphereConverter.cs***, and implement IValueConverter interface:

```
/* Tutorial:
 * Simple Gauge WPF & Silverlight application
 * Bars To Atmosphere Converter
 */
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Data;

namespace Tutorial
{
    class BarsToAtmosphereConverter: IValueConverter
    {
        // convert Bars to Atm
        public object Convert(object value, Type targetType, object parameter,
System.Globalization.CultureInfo culture)
        {
            return (double)value*0.986923;
        }

        // convert Atm to Bars
        public object ConvertBack(object value, Type targetType, object parameter,
System.Globalization.CultureInfo culture)
        {
            return (double)value / 0.986923;
        }
    }
}
```

Converter will be active after assembling into Resource dictionary for the Window/UserControl in WPF and Silverlight. Define a namespace with converter:

```
xmlns:converter="clr-namespace:Tutorial"
```

Then define ***BarsToAtmosphereConverter*** class in **Resources**. After that we can use converter during the Data Binding.

```
<Window.Resources>
    <converter:BarsToAtmosphereConverter x:Key="BarsToAtm"/>
</Window.Resources>
```

Data Binding expression:

```
"{Binding [property name], Converter={StaticResource BarsToAtm}, ElementName=[name]}"
```

Gauges – Simple Gauge Control in WPF and Silverlight

The code inside the Grid Container for both Silverlight and WPF application:

```
<Grid>
    <Grid.RowDefinitions>
        <RowDefinition Height="10*"></RowDefinition>
        <RowDefinition Height="1*"></RowDefinition>
    </Grid.RowDefinitions>
    <!–Gauge start-->
    <agc:Gauge x:Name="gauge" StrokeThickness="4" Width="Auto" Height="Auto">
         <!--Secondary scale-->
        <agc:Gauge.SecondaryScales>
             <agc:Scale Name="atm"
                        RangeBegin="{Binding PrimaryScale.RangeBegin,
Converter={StaticResource BarsToAtm}, ElementName=gauge}"
                        RangeEnd="{Binding PrimaryScale.RangeEnd, Converter={StaticResource
BarsToAtm}, ElementName=gauge}"
                        Value="{Binding Value, Converter={StaticResource BarsToAtm},
ElementName=slider}"
                        ValueFormat="F2" AngleBegin="270" AngleEnd="-30" MinorTickCount="0"
ArcPadding="43"
                        ValueIndicatorDistanceType="Absolute" ValueIndicatorAngle="180"
ValueIndicatorDistance="30"
                        LabelAngle="150" LabelDistanceType="Absolute" LabelDistance="34">
                 <agc:Scale.Label>
                     <TextBlock Text="Atm"></TextBlock>
                 </agc:Scale.Label>
                </agc:Scale>
            </agc:Gauge.SecondaryScales>

         <!--Parimary scale-->
            <agc:Gauge.PrimaryScale>
                <agc:Scale Name="bars"
                           Value="{Binding Value, ElementName=slider}"
                           ValueIndicatorDistanceType="Absolute" ValueIndicatorAngle="180"
LabelDistanceType="Absolute"
                           LabelAngle="30" LabelDistance="33" ValueIndicatorDistance="-30">
                    <agc:Scale.Label>
                        <TextBlock Text="Bars"></TextBlock>
                    </agc:Scale.Label>
                </agc:Scale>
            </agc:Gauge.PrimaryScale>
            <agc:Gauge.Stroke>
                <RadialGradientBrush>
                    <GradientStop Color="Black" Offset="0.95"/>
                    <GradientStop Color="White" Offset="1"/>
                    <GradientStop Color="#FF3C3C3C" Offset="0.2"/>
                </RadialGradientBrush>
            </agc:Gauge.Stroke>
    </agc:Gauge>
    <!–Gauge end-->

    <!--Slider-->
    <Slider x:Name="slider" Grid.Row="1"
            Margin="10,0"
            Maximum="{Binding PrimaryScale.RangeEnd, ElementName=gauge}"
            Minimum="{Binding PrimaryScale.RangeBegin, ElementName=gauge}"/>
    </Grid>
```

Compile and run the application.