BulletGraph for WinForms

2019.08.20 更新

グレープシティ株式会社

BulletGraph for Winforms

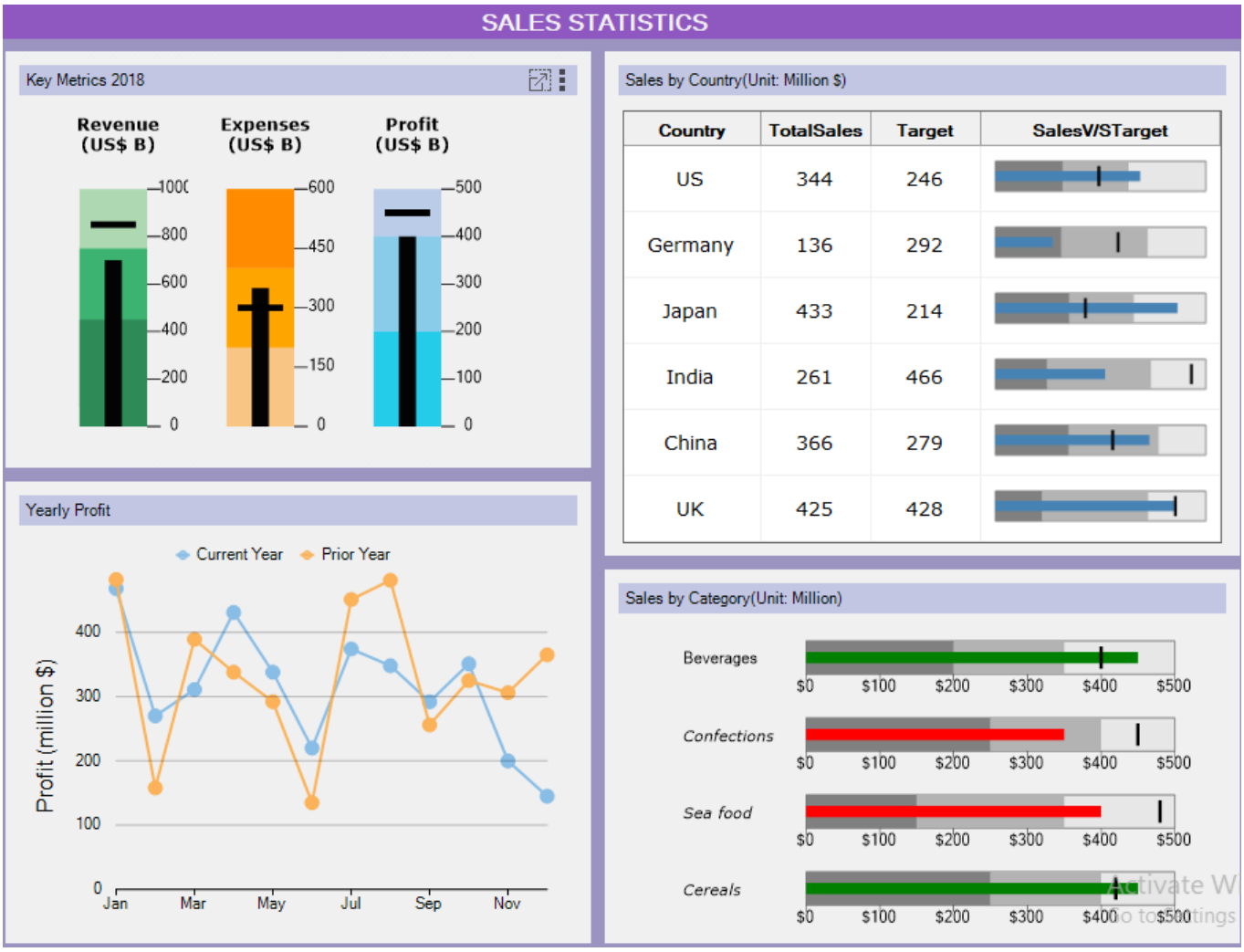
目次

BulletGraph for WinForms の概要

Data visualization helps a user to effectively consume data analytics and explore insights. A type of linear gauge, bullet graph is a data visualization tool that displays a single key measure on a linear scale along with comparative measures and different qualitative ranges. The bullet graph was developed by dashboard design expert Stephen Few. Unlike gauges, bullet graphs do not take up a lot of space to show a single measure. They support more efficient reading, and can be used to analyse performance values such as profits, sales revenue, sales performance, KPIs etc.

BulletGraph for WinForms comes with a no-frills linear design.The BulletGraph is a flexible, lightweight and high-peformance data visualization control. When it comes to metrics, **BulletGraph** packs a lot of interesting features that can drive your business goals, such as configurable orientation, export to image, custom styling and more.

The following picture illustrates some of the capabilities of BulletGraph in presenting data on a dashboard in a sale statistics scenario. You can notice how the Bullet Graph control has been utilized to show different types of information like revenue, expenses, profit, sales etc. without taking up much space on the dashboard.



主な特長

The major features of **BulletGraph** control are elucidated below:

- **Efficient data presentation**

  The Bullet Graph control helps present a lot of information without taking up much space. It allows to showcase a rich display of information by letting you compare the featured measure with one or more comparative measure. This will help the user gain greater insight into the data. For example, bullet graphs become extremely helpful in comparing the company growth of current year with preceding year.

- **Configurable orientation**

  Though the default orientation of the C1BulletGraph is horizontal, the control can also be given a vertical orientation programmatically, if needed.

- **Qualitative ranges**

  The Bullet Graph displays the featured measure in context of qualitative ranges of performance, such as good/fair/bad or below expectations/meets expectations/exceeds expectation.

- **Export to image**

  The Bullet Graph control can be exported to various image formats such as PNG and JPEG for further use.

- **Styling**

  The Bullet Graph control provides numerous styling options to customize the visual appearance of the control, such as labels, marks and ranges and other UI elements.

- **Integration with other controls**

  Bullet Graph can be conveniently integrated with FlexGrid, MS DataGridView control and dashboard layout. This allows users to present data in a way that is easier to understand.

コアクラス

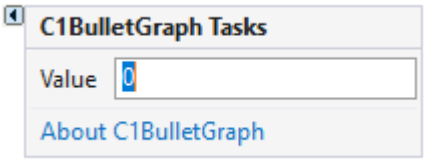
The following table lists some of the key classes and their properties. You can click on the cross-references to get to the API documentation that provides detailed description for the key members.

C1BulletGraph
Properties: **Target**, **Value**, **ShowValue**, **Minimum**, **Maximum**, **Origin**, **Orientation**, **GraphScale**, **IsReversed**, **Good**, **Bad**, **Caption**, **Styles** **Methods**: **GetImage**

設計時サポート

The **C1BulletGraph** control includes a smart tag in Visual Studio.

The smart tag appears similar to the image below.



The **C1BulletGraph Tasks** menu provides the following options:

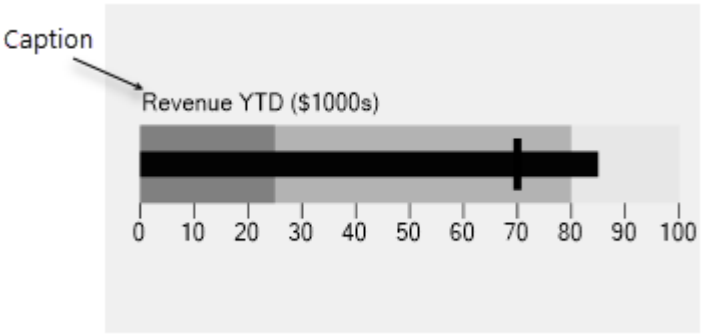
- **Value:**
  This allows you to prescribe a numerical value between the minimum and maximum limits of the gauge to set the current 'value' for the Bullet Graph.
- **About C1BulletGraph:**
  Clicking **About C1BulletGraph** displays a dialog box, which is helpful in finding the version number of the product and other resources.

C1BulletGraphの要素

The major building blocks of the **BulletGraph for WinForms** are as follows:

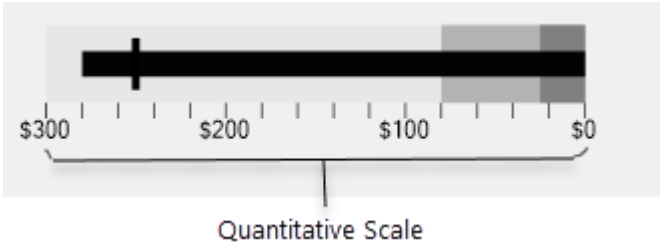
Text label

The **Text Label** or **Caption** in **C1BulletGraph** identifies the measure. The default formatting of Text Label in BulletGraph is with respect to position, color, orientation and size. In case of horizontally-oriented graphs, text labelling is aligned to the left by default, while in vertically-oriented graphs it is aligned at the top. The default font color of the text label is 100% black and legible.



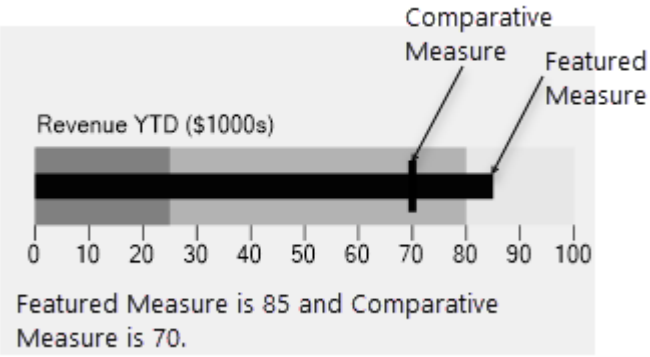
Quantitative scale

The quantitative scale of the **BulletGraph** resembles and functions like the quantitative scale of a two-dimensional XY graph. The quantitative scale consists of text labels and tick marks that identify equal intervals of measure, which can be observed along the top or bottom in the horizontally-oriented graph, or along the left or right side of the vertically-oriented graph. The tick marks can be assigned a position along the inner or outer edge of the plot area. The range of the quantitative feature usually begins at zero, but it may begin with a negative number or a value greater than zero if required.



Featured measure

The Featured Measure is the portion of the bullet graph that is visually prominent and it displays the primary data. This featureis encoded as a bar, like the bar on a bar graph. The only exception is when the quantitative scale begins at a value greater than zero. In such a case, the featured measure will be encoded as a symbol such as an X or a dot.



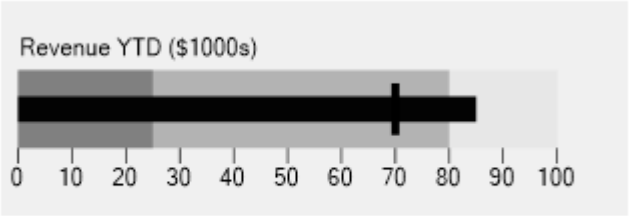
Featured Measure is 85 and Comparative Measure is 70.

Comparative measure

The Comparative measure is visually less dominant as compared to the featured measure, but easy to see in relation to the featured measure. It is encoded as a short line that runs perpendicular to the BulletGraph. Whenever the featured measure intersects a comparative measure, the comparative measure usually appears behind the featured measure.

Qualitative ranges

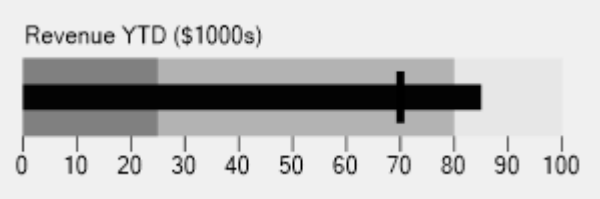
The qualitative ranges indicate the qualitative state of the featured measure and the degree to which it resides within that state. For example, if the measure extends into a range that represents “good”, then the distance that it travels into this range is indicative of how good it is. The number of qualitative ranges can span between two and five, though the ideal range is three.



Different background fill colors showing qualitative ranges.

クイックスタート

This section will help you get started with Bullet Graph quickly.



The image above shows how the BulletGraph control appears at runtime.

1. Create a new **Windows Forms** application.
2. Drag and drop the **C1BulletGraph** control to the form.
3. Set the width and height of the Bullet Graph control to 250 and 100 respectively.
4. From the properties window, set the Value property of the Bullet Graph control to 85.

   Note: The default values for the scale is 0 to 100.
5. Set the Target property of the Bullet Graph control to 70.
6. Expand the Caption node and set the Text property to 'Revenue YTD ($1000s)'.
7. Run the application and see how the BulletGraph control appears at runtime.

BulletGraphの操作

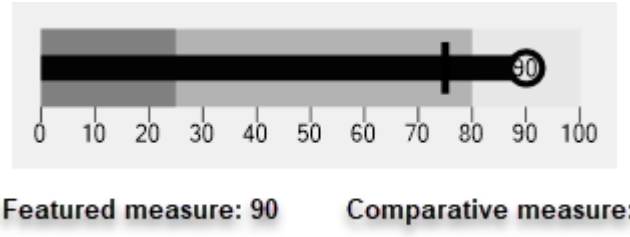
The best way to analyze and comprehend your data is to visualize it. **BulletGraph for WinForms** helps visualize image in an intuitive way. This section will walk you through the critical aspects of C1BulletGraph.

メジャー

Measure typically refers to quantitative data such as profit, sales, revenue, expenditure and so on. There are two types of measures, featured and comparative measure. Featured measure refers to the primary metric that is displayed by a bullet graph. Comparative measure refers tothe metric against which you want to compare the featured measure such as a target or some past result. The comparative measure usually appears behind the featured measure.

Bullet graphs help you analyze the performance of a given metric against target standards. Measures are considered good when they are high, for example, revenue and profit bullet graphs. But in the case of an expenditure bullet graph, measures are considered good only when they are low. So, the idea of whether a measure is good or bad will change with the context.

This section discusses how to set the value of the featured measure and comparative measure for the BulletGraph control.



Featured Measure

The value of the featured measure is encoded as a bar in the bullet graph and is set using the **Value** property of the **C1BulletGraph** class.

This is depicted programmatically below:

- **Visual Basic**

```
'注目のメジャーの値を設定します
C1BulletGraph1.Value = 90
```

- **C#**

```
//注目のメジャーの値を設定します
c1BulletGraph1.Value =90;
```

In case you want to explicitly display the value of the featured measure as text in the bullet graph, you can make use of the **ShowValue** property of the **C1BulletGraph** class.

The code below shows an example where this property is set to true.

- **Visual Basic**

```
'注目のメジャーの値をブレットグラフにテキストとして表示します
C1BulletGraph1.ShowValue = True
```

- **C#**

```
//注目のメジャーの値をブレットグラフにテキストとして表示します
c1BulletGraph1.ShowValue = true;
```

Comparative Measure

The value of the comparative measure can be set using the **Target** property of the **C1BulletGraph** class. This value is encoded as a short line running perpendicular to the orientation of the bullet graph.

The code below depicts an example.

- **Visual Basic**

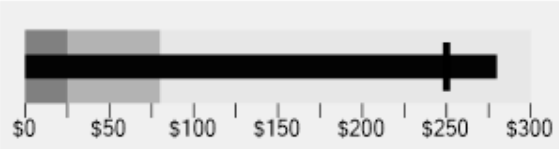
```
'比較メジャーの値を設定します
C1BulletGraph1.Target = 75
```

- **C#**

```
//比較メジャーの値を設定します
c1BulletGraph1.Target = 75;
```

定量的スケール

The quantitative scale of the bullet graph consists of tick marks and text labels that identify equal intervals of the featured measure. You can customize the range of the quantitative using the **Minimum** and **Maximum** properties of the **C1BulletGraph** class. Maximum specifies the end value of the scale, while Minimum specifies the start value of the scale.



The image depicts the quantitative scale of BulletGraph control.

The code below shows how you can assign different values to Minimum and Maximum of the quantitative scale for a bullet graph:

- **Visual Basic**

```
'定量的スケールの開始値を設定します
C1BulletGraph1.Minimum = 0
'定量的スケールの終了値を設定します
C1BulletGraph1.Maximum = 300
```

- **C#**

```
c1BulletGraph1.Minimum = 0; //定量的スケールの開始値を設定します
c1BulletGraph1.Maximum = 300; //定量的スケールの終了値を設定します
```

The C1BulletGraph class also provides a **GraphScale** property to customize the bullet graph scale. This property is of the type **BulletGraphScale** class. The BulletGraphScale class contans the properties such as **MarksInterval** and **LabelsInterval** to

change the space between each tick mark and label on the scale of a bullet graph. You can further choose to show or hide the scale labels and scale marks in a bullet graph with **ShowLabels** and **ShowMarks** properties of the BulletGraphScale class. You can also specify formatting for the scale labels with the **Format** property.

The bullet graph scale can be customized as shown in the code below:

- **Visual Basic**

```
'定量的スケールで各目盛りの間の間隔を設定します
C1BulletGraph1.GraphScale.MarksInterval = 25
'定量的スケールで各ラベルの間の間隔を設定します
C1BulletGraph1.GraphScale.LabelsInterval = 50
'スケールのラベルを表示します
C1BulletGraph1.GraphScale.ShowLabels = True
'スケールのマークを表示します
C1BulletGraph1.GraphScale.ShowMarks = True
'スケールのラベルを書式設定します
C1BulletGraph1.GraphScale.Format = "$#,##0"
```

- **C#**

```
c1BulletGraph1.GraphScale.MarksInterval = 25; //定量的スケールで各目盛りの間の間隔を設定します
c1BulletGraph1.GraphScale.LabelsInterval = 50; //定量的スケールで各ラベルの間の間隔を設定します
c1BulletGraph1.GraphScale.ShowLabels = true; //スケールのラベルを表示します
c1BulletGraph1.GraphScale.ShowMarks = true; //スケールのマークを表示します
c1BulletGraph1.GraphScale.Format = "$#,##0"; //スケールのラベルを書式設定します
```

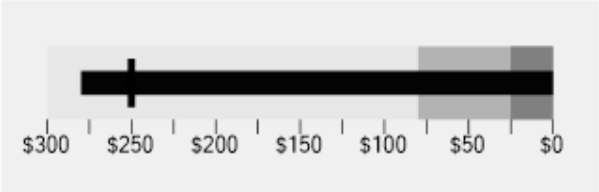
The BulletGraph control also allows you to show the scale values in reversed order, that is from maximum to minimum using the **IsReversed** property of the C1BulletGraph class.

- **Visual Basic**

```
'スケール値を逆の順序で表示します(最大から最小)
C1BulletGraph1.IsReversed = True
```

- **C#**

```
c1BulletGraph1.IsReversed = true; //スケール値を逆の順序で表示します(最大から最小)
```



The bar which is used to encode the featured measure begins from the minimum value of the scale. This is a default setting. However, in case you want to start the bar from a different value, you can use the **Origin** property of the C1BulletGraph class. The Origin property becomes useful when you want to start the featured measure bar from zero rather than a negative value.

- **Visual Basic**

```
'注目のメジャーを表すバーの原点を設定します
C1BulletGraph1.Origin = 0
```

- **C#**

```
c1BulletGraph1.Origin = 0; //注目のメジャーを表すバーの原点を設定します
```

If the quantitative scale begins at a value greater than zero, that is the minimum value of the scale is set to a value which is greater than zero, then the featured measure is represented on the bullet graph by the symbol 'X' instead of a bar.

- **Visual Basic**

```
'定量的スケールの開始値を0より大きい値に設定します
C1BulletGraph1.Minimum = 30
```

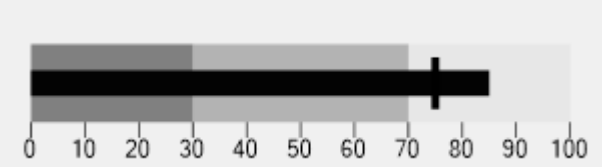
- **C#**

```
c1BulletGraph1.Minimum = 30; //定量的スケールの開始値を0より大きい値に設定します
```

For example, the code given above sets the minimum value of the quantitative scale to a value greater than zero, say 30, hence the featured measure is represented by the symbol 'X' in the BulletGraph control as depicted in the image shown below.

定性的範囲

The qualitative range is an element of the bullet graph that helps to divide the quantitative scale into different zones to signal whether the featured measure is in good, bad, or some other state.



The image above depicts the three different qualitative ranges of BulletGraph control, indicated with varying intensities of black color.

The code below shows how you can assign different qualitative ranges or properties such as **Good** and **Bad** to the bullet graph, both starting and ending values.

- **Visual Basic**

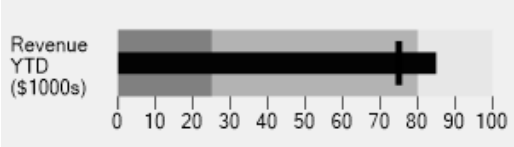
```
'メジャーのBadに対して範囲の開始値を設定します
C1BulletGraph1.Bad.From = 0
'メジャーのBadに対して範囲の終了値を設定します
C1BulletGraph1.Bad.To = 30
'メジャーのGoodに対して範囲の開始値を設定します
C1BulletGraph1.Good.From = 30
'メジャーのGoodに対して範囲の終了値を設定します
C1BulletGraph1.Good.To = 70
```

- **C#**

```
c1BulletGraph1.Bad.From =0; //メジャーのBadに対して範囲の開始値を設定します
c1BulletGraph1.Bad.To =30; //メジャーのBadに対して範囲の終了値を設定します
c1BulletGraph1.Good.From = 30; //メジャーのGoodに対して範囲の開始値を設定します
c1BulletGraph1.Good.To = 70; //メジャーのGoodに対して範囲の終了値を設定します
```

BulletGraphのキャプション

A caption helps to provide helpful information to the user about the control. Similarly, the caption (or textlabel) of the C1BulletGraph conveys the information being represented by the bullet graph, such as sales, revenue, expenditure, customer count, number of defects and so on. It is mainly used for identifying the featured measure.



The image above shows the caption of the BulletGraph control. Here, it is revenue.

The bullet graph caption can be added and customized using the **Caption** property of C1BulletGraph class. This property is of the type **BulletGraphCaption** class and can be used to define the text, alignment and position of the bullet graph caption using the **Text**, **Alignment** and **Position** properties respectively. The BulletGraphCaption also exposes the **Height** and the **Width** properties to specify the height and width of the bullet graph caption.

The code below shows how you can set the caption of the BulletGraph control:

- **Visual Basic**

```
'キャプションのテキストを設定します
C1BulletGraph1.Caption.Text = "Revenue YTD ($1000s)"
'レイアウト四角形に対するキャプションの配置を設定します
C1BulletGraph1.Caption.Alignment = StringAlignment.Near
'グラフに対するキャプションの位置を設定します
C1BulletGraph1.Caption.Position = BulletGraphCaptionPosition.Left
'キャプションの幅を設定します
C1BulletGraph1.Caption.Width = 55
'キャプションの高さを設定します
C1BulletGraph1.Caption.Height = 40
```

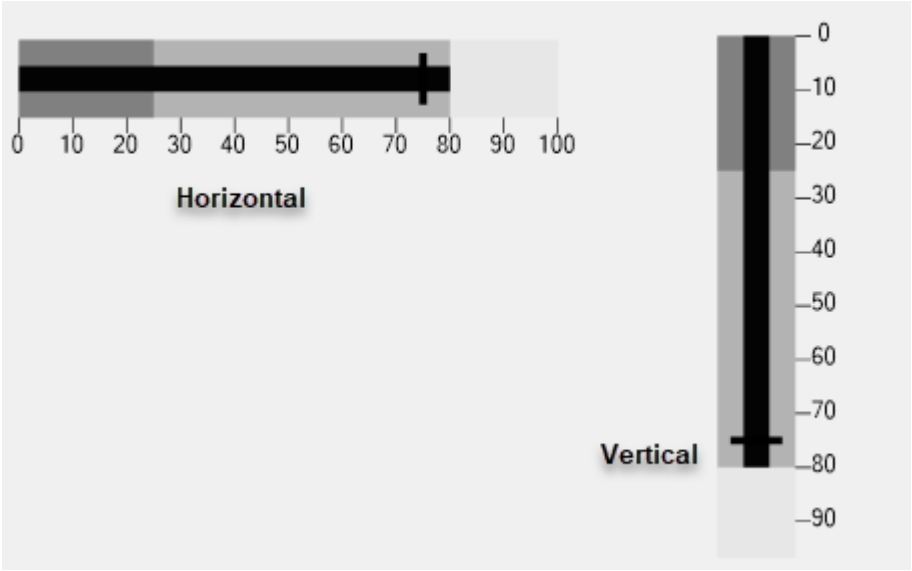
- **C#**

```
c1BulletGraph1.Caption.Text = "Revenue YTD ($1000s)"; //キャプションのテキストを設定します
c1BulletGraph1.Caption.Alignment = StringAlignment.Near; //レイアウト四角形に対するキャプションの配置を設定します
c1BulletGraph1.Caption.Position = BulletGraphCaptionPosition.Left; //グラフに対するキャプションの位置を設定します
c1BulletGraph1.Caption.Width =55; //キャプションの幅を設定します
c1BulletGraph1.Caption.Height = 40; //キャプションの高さを設定します
```

方向の設定

A bullet graph can have either horizontal or vertical orientation. Though the default orientation of the control is horizontal, the **Orientation** property of the **C1BulletGraph** class can be used to assign an orientation to the bullet graph. This property accepts values (Horizontal/Vertical) from the C1GaugeOrientation enum.

The image below shows the horizontal and vertical orientation in BulletGraph control.



For example, to orient the bullet graph vertically, the Orientation property of C1BulletGraph class can be set to C1GaugeOrientation.Vertical as depicted in the code below:

- **Visual Basic**

```
'BulletGraphを垂直方向に設定します
C1BulletGraph1.Orientation = C1GaugeOrientation.Vertical
```

- **C#**

```
//BulletGraphを垂直方向に設定します
c1BulletGraph1.Orientation = C1GaugeOrientation.Vertical;
```

画像へのエクスポート

The export option in a control helps you to save and store the screenshots of the control in various file formats. Since the BulletGraph control is a data visualization control, it can be exported to different image formats such as PNG, JPEG etc.

To export a BulletGraph control as an image, you can make use of the **GetImage** method of the **C1BulletGraph** class. This method returns an object of the type **Image** which is the screenshot of the BulletGraph control. You can optionally specify the height, width, and pixel format of the image to be returned using different overloads of this method.

Once you get the control's image, you can use the **Save** method of the **Image** class to save the returned image.

The code below depicts how you can export the BulletGraph control to an image on the click of a button:

- **Visual Basic**

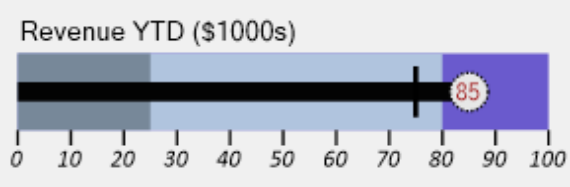
```
Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
    Dim dialog = New SaveFileDialog()
    dialog.Filter = "PNG|*.png|JPEG |*.jpeg"

    If dialog.ShowDialog() = DialogResult.OK Then
        Dim ext As String = dialog.FileName.Split("."c)(1)
        Dim bulletGraphImage As Image = C1BulletGraph1.GetImage()
        bulletGraphImage.Save(dialog.FileName)
    End If
End Sub
```

- **C#**

```
private void export_Click(object sender, EventArgs e)
{
    var dialog = new SaveFileDialog();
    dialog.Filter = "PNG|*.png|JPEG |*.jpeg";
    if (dialog.ShowDialog() == DialogResult.OK)
    {
        string ext = dialog.FileName.Split('.')[1];
        Image bulletGraphImage = c1BulletGraph1.GetImage();
        bulletGraphImage.Save(dialog.FileName);
    }
}
```

スタイル

The look and feel of a bullet graph control can be customized according to your preference. The **C1BulletGraph** class provides a **Styles** property that can be used to customize the different visual parts of the BulletGraph control. It is of the type **BulletGraphTheme** class. Styling can be applied to BulletGraph control using different classes for different visual parts of the control.



The image given above depicts the use of **Styles** property in **WinForms** application.

The BulletGraph control consists of several UI elements such as labels, marks and ranges. The BulletGraphTheme class provides several properties to handle the styling of the BulletGraph control and the above mentioned elements. Each of these properties is of a specific class type and is used to style specific parts of the BulletGraph control.

These properties are given below:

- **Common:** This property is of the type **BulletGraphCommonStyle** class. It allows you to customize the appearance (**graph offset**) of the BulletGraph control.
  The code below shows an example to style the DataFilter control:
  - **Visual Basic**

```
'BulletGraphをスタイルします
C1BulletGraph1.Styles.Common.GraphOffset = 10 'グラフのオフセットを設置します
```

  - **C#**

```
//BulletGraphをスタイルします
c1BulletGraph1.Styles.Common.GraphOffset = 10; //グラフのオフセットを設置します
```

- **Labels:** This property is of the type **BulletGraphLabelStyles** class. It allows you to customize the appearance (color, font and font-size) of different labels the BulletGraph control is composed of such as the **caption** label, **scale** labels and the featured **value** label.
  The code below shows as example to style the bullet graph labels:
  - **Visual Basic**

```
'ラベルをスタイルします
C1BulletGraph1.Styles.Labels.Scale.Color = Color.Black
C1BulletGraph1.Styles.Labels.Scale.Font = New Font("Calibri", 9, FontStyle.Italic)
C1BulletGraph1.Styles.Labels.Caption.FontSize = 20
C1BulletGraph1.Styles.Labels.Value.Color = Color.Brown
```

  - **C#**

```
//ラベルをスタイルします
c1BulletGraph1.Styles.Labels.Scale.Color = Color.Black;
c1BulletGraph1.Styles.Labels.Scale.Font = new Font("Calibri", 9, FontStyle.Italic);
c1BulletGraph1.Styles.Labels.Caption.FontSize = 20;
c1BulletGraph1.Styles.Labels.Value.Color = Color.Brown;
```

- **Marks:** This property is of the type **BulletGraphMarkStyles** class. It allows you to customize the appearance of different marks in the BulletGraph such as the comparative measure or **target** mark (lines that runs perpendicular to the existing orientation of the graph), **scale** tick marks and the featured **value** mark (the circle within which the featured value label is shown).
  The code below shows an example to style the bullet graph marks:
  - **Visual Basic**

```
'マークをスタイルします
C1BulletGraph1.Styles.Marks.Target.Color = Color.Black
C1BulletGraph1.Styles.Marks.Target.Thickness = 4
C1BulletGraph1.Styles.Marks.Scale.Thickness = 2
C1BulletGraph1.Styles.Marks.Value.Width = 30
C1BulletGraph1.Styles.Marks.Value.Border.LineStyle = C1GaugeBorderStyle.Dot
C1BulletGraph1.Styles.Marks.Value.Border.Thickness = 5
```

  - **C#**

```
//マークをスタイルします
c1BulletGraph1.Styles.Marks.Target.Color = Color.Black;
c1BulletGraph1.Styles.Marks.Target.Thickness = 4;
c1BulletGraph1.Styles.Marks.Scale.Thickness = 2;
c1BulletGraph1.Styles.Marks.Value.Width = 30;
c1BulletGraph1.Styles.Marks.Value.Border.LineStyle = C1GaugeBorderStyle.Dot;
c1BulletGraph1.Styles.Marks.Value.Border.Thickness = 2;
```

- **Ranges:** This property is of the type **BulletGraphRangeStyles** class. It allows you to customize the appearance

of different ranges in the BulletGraph control. It is composed of a main bullet graph bar, good/bad ranges and the featured measure bar.
The code below shows as example to style the bullet graph ranges:

- **Visual Basic**

```
'範囲をスタイルします
C1BulletGraph1.Styles.Ranges.Bar.Color = Color.SlateBlue
C1BulletGraph1.Styles.Ranges.Good.Color = Color.LightSteelBlue
C1BulletGraph1.Styles.Ranges.Bad.Color = Color.LightSlateGray
C1BulletGraph1.Styles.Ranges.Value.Width = 15
```

- **C#**

```
//範囲をスタイルします
c1BulletGraph1.Styles.Ranges.Bar.Color = Color.SlateBlue;
c1BulletGraph1.Styles.Ranges.Good.Color = Color.LightSteelBlue;
c1BulletGraph1.Styles.Ranges.Bad.Color = Color.LightSlateGray;
c1BulletGraph1.Styles.Ranges.Value.Width = 15;
```

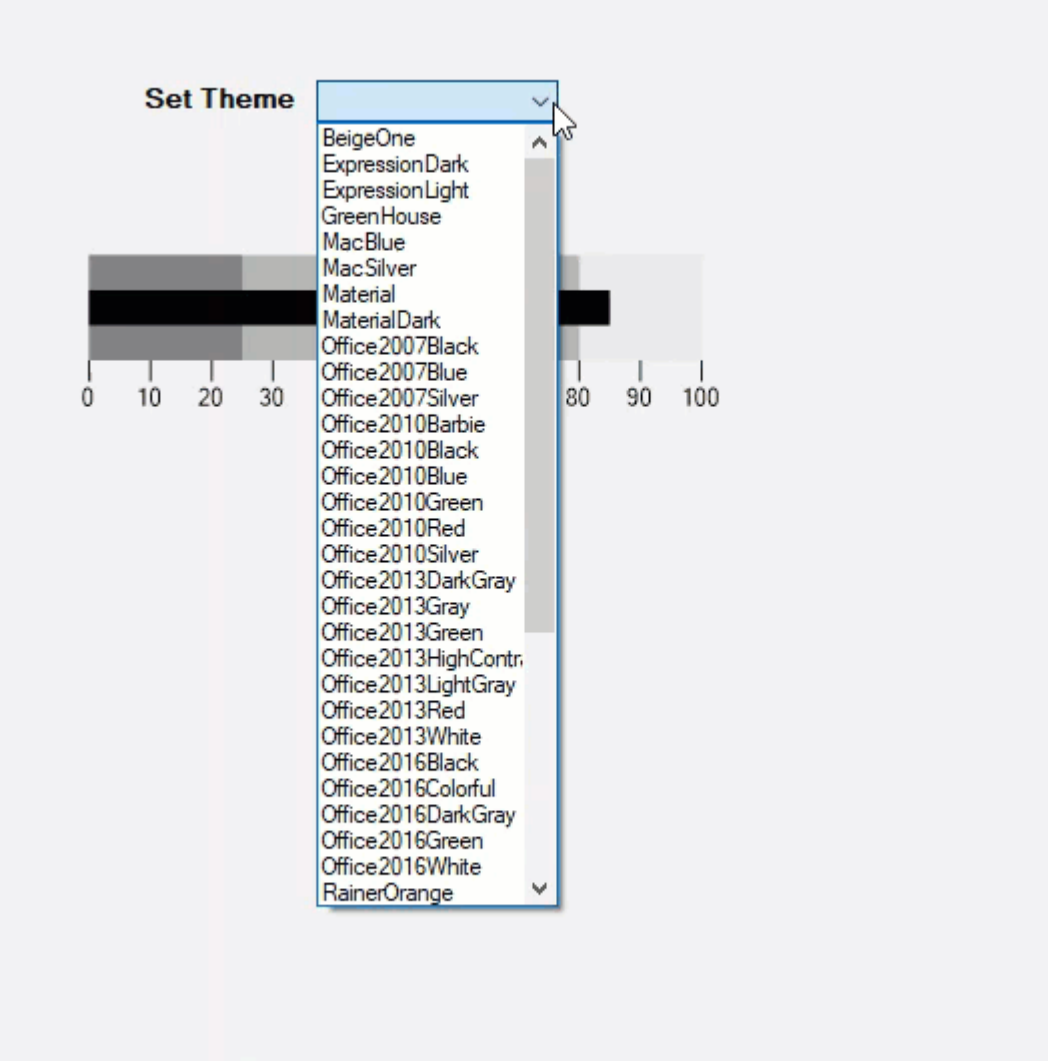
テーマ

Customizing the look and feel of a control is an important part of the application development which is required to give it a seamless user interface. Our Themes for WinForms provides an easy and intuitive way to apply themes to all ComponentOne controls. It allows you to style the control using the **C1ThemeController** component and **C1ThemeDesigner** application.

Using C1ThemeController

The **C1ThemeController** component can be used to apply built-in themes to the controls. The component loads and manages visual themes and applies them to the control. You can simply drag and drop the **C1ThemeController** from the toolbox on your form and use the built-in themes to style the control. For example, BeigeOne, Material, MaterialDark, Office2016Green, etc. are some available built-in themes.

The following GIF shows different themes getting applied on the **C1BulletGraph** control.



To apply a built-in theme to the Bullet Graph control, follow these steps:

1. Drag and drop the **c1ThemeController** and **C1BulletGraph** controls on the form. Observe that the **c1ThemeController** is added to the Component tray.
2. Right-click the **C1BulletGraph** control and select Properties from the context menu.
3. In the Properties window, click the dropdown next to the Theme on **c1ThemeController1** property and select a theme from the ComponentOne themes list.

Using C1ThemeDesigner application

C1ThemeDesigner application provides you an easy-to-use UI so that you can create your own themes or modify the existing ones. The application gets installed on your system along with ComponentOne WinForms Edition and can be accessed through **C1ThemeDesigner.4.exe** kept at C:\Program Files (x86)\ComponentOne\Apps\v4.0. For more information on using **C1ThemeDesigner** application to create and modify themes, see C1Theme Designer Application Overview.

チュートリアル

The walkthrough topics in this section are created with the assumption that you are familiar with the **BulletGraph** control and know how to use it in general.

DataGridViewとのBulletGraphの使用

Learn how to use the **BulletGraph** control with the **DataGridView** control.

FlexGridとのBulletGraphの使用

Learn how to use the **BulletGraph** control with **the FlexGrid** control.

DataGridViewとのBulletGraphの使用

Consider a scenario where you have data in a tabular format and you want to visualize the same using a visual control. This walkthrough explains how this scenario can be implemented using MS DataGridView as the control for tabular representation of data and BulletGraph as the control for visual representation of the same data.

Country	Sales (millions)	Target (millions)	Bad (millions)	Good (millions)	SalesV/STarget
US	$ 159	$ 401	$ 82	$ 222	
Germany	$ 24	$ 357	$ 67	$ 207	
Japan	$ 286	$ 448	$ 78	$ 247	
India	$ 366	$ 466	$ 122	$ 294	
China	$ 25	$ 385	$ 148	$ 244	
UK	$ 84	$ 385	$ 132	$ 300	
Denmark	$ 220	$ 357	$ 69	$ 290	
Indonesia	$ 10	$ 376	$ 172	$ 263	

The BulletGraph control can be rendered as an image in the **DataGridView** cells. To learn how this can be done, follow the steps given below:

1. **Create Data Source for DataGridView**
2. **Configure DataGridView control**
3. **Configure BulletGraph control**
4. **Rendering BulletGraph In DataGridView**

Note: The BulletGraph control can also be rendered as an image in bound column of the DataGridView control using similar approach.

Step 1: Create Data Source for DataGridView

1. Create a new **Windows Forms** application.
2. Drag and drop the **MS DataGridView** control from the Toolbox onto your form. From the Properties window, set its **Dock** property to Fill.
3. To populate the **DataGridView** with data, create a data table containing relevant data.

The code below defines a method named '**CreateDataTable**' to create a data table that contains the country-wise sales information of a company.

- **Visual Basic**

```
Private Sub CreateDataTable()
    _dt = New DataTable()
    _dt.Columns.Add("Country", GetType(String))
    _dt.Columns.Add("Sales", GetType(Double))
    _dt.Columns.Add("Target", GetType(Double))
    _dt.Columns.Add("Bad", GetType(Double))
    _dt.Columns.Add("Good", GetType(Double))
    Dim countries As String() = {"US", "Germany", "Japan", "India", "China", "UK", "Denmark", "Indonesia"}
    Dim random As Random = New Random()

    For i As Integer = 0 To countries.Length - 1
        Dim totalSales As Integer = random.[Next](0, 500)
```

```
        Dim target As Integer = random.[Next](351, 499)
        Dim bad As Integer = random.[Next](50, 200)
        Dim good As Integer = random.[Next](201, 350)
        _dt.Rows.Add(New Object() {countries(i), totalSales, target, bad, good})
    Next
End Sub
```

- **C#**

```
private void CreateDataTable()
{
    _dt = new DataTable();
    _dt.Columns.Add("Country", typeof(String));
    _dt.Columns.Add("Sales", typeof(Double));
    _dt.Columns.Add("Target", typeof(Double));
    _dt.Columns.Add("Bad", typeof(Double));
    _dt.Columns.Add("Good", typeof(Double));
    string[] countries = { "US", "Germany", "Japan", "India", "China", "UK", "Denmark", "Indonesia" };
    Random random = new Random();
    for (int i = 0; i < countries.Length; i++)
    {
        int totalSales = random.Next(0, 500);
        int target = random.Next(351, 499);
        int bad = random.Next(50, 200);
        int good = random.Next(201, 350);
        _dt.Rows.Add(new object[] { countries[i], totalSales, target, bad, good });
    }
}
```

Note that _dt is declared as a private global variable of type **DataTable**.

Step 2: Configure DataGridView control

1. To bind the grid with data, assign the data table created in the previous section to the DataGridView's DataSource property.
2. To display BulletGraph in DataGridView cells, add an additional unbound **DataGridViewImageColumn** to the DataGridView control.
3. Optionally, customize the DataGridView's appearance using appropriate properties of the DataGridView control.
4. To render the BulletGraph control as an image in the added unbound column 'SalesV/STarget' of the DataGridView, subscribe to the **CellPainting** event of the DataGridView class.

The code below creates a method named '**SetUpGrid**', to configure the above mentioned settings of the DataGridView control.

- **Visual Basic**

```
Private Sub SetUpGrid()
    DataGridView1.DataSource = _dt
    '非連結DataGridViewImageColumnをDataGridViewに追加します
    'この列は、BulletGraphコントロールを画像として表示するために使用されます
    Dim dataGridViewImageColumn As DataGridViewImageColumn = New DataGridViewImageColumn()
    dataGridViewImageColumn.Name = "SalesV/STarget"
    DataGridView1.Columns.Add(dataGridViewImageColumn)
    'DataGridViewの追加プロパティを設定します(オプション)
    DataGridView1.ColumnHeadersDefaultCellStyle.Font = New Font(FontFamily.GenericSansSerif, 8.5F, FontStyle.Bold)
    DataGridView1.Font = New Font(FontFamily.GenericSansSerif, 9.0F, FontStyle.Regular)
    DataGridView1.Columns(DataGridView1.Columns.Count - 1).AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill

    For i As Integer = 0 To DataGridView1.Rows.Count - 1
        DataGridView1.Rows(i).Height = 40
    Next

    DataGridView1.RowHeadersVisible = False
    DataGridView1.AllowUserToAddRows = False
    DataGridView1.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter
    DataGridView1.DefaultCellStyle.Format = "$ #,##0"
    DataGridView1.ColumnHeadersDefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter
    DataGridView1.Columns("Sales").HeaderText = "Sales" & vbLf & "(millions)"
    DataGridView1.Columns("Target").HeaderText = "Target" & vbLf & "(millions)"
    DataGridView1.Columns("Bad").HeaderText = "Bad" & vbLf & "(millions)"
    DataGridView1.Columns("Good").HeaderText = "Good" & vbLf & "(millions)"
    'DataGridViewのCellPaintingイベントを購読します
    AddHandler DataGridView1.CellPainting, AddressOf DataGridView1_CellPainting
End Sub
```

- **C#**

```
private void SetUpGrid()
{
    dataGridView1.DataSource = _dt;

    //非連結DataGridViewImageColumnをDataGridViewに追加します
    //この列は、BulletGraphコントロールを画像として表示するために使用されます
    DataGridViewImageColumn dataGridViewImageColumn = new DataGridViewImageColumn();
    dataGridViewImageColumn.Name = "SalesV/STarget";
    dataGridView1.Columns.Add(dataGridViewImageColumn);
```

```
    //DataGridViewの追加プロパティを設定します(オプション)
    dataGridView1.ColumnHeadersDefaultCellStyle.Font= new Font(FontFamily.GenericSansSerif, 8.5f, FontStyle.Bold);
    dataGridView1.Font = new Font(FontFamily.GenericSansSerif, 9f, FontStyle.Regular);
    dataGridView1.Columns[dataGridView1.Columns.Count - 1].AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;
    for (int i = 0; i < dataGridView1.Rows.Count; i++)
    {
        dataGridView1.Rows[i].Height = 40;
    }
    dataGridView1.RowHeadersVisible = false;
    dataGridView1.AllowUserToAddRows = false;
    dataGridView1.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;
    dataGridView1.DefaultCellStyle.Format = "$ #,##0";
    dataGridView1.ColumnHeadersDefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;
    dataGridView1.Columns["Sales"].HeaderText = "Sales\n(millions)";
    dataGridView1.Columns["Target"].HeaderText = "Target\n(millions)";
    dataGridView1.Columns["Bad"].HeaderText = "Bad\n(millions)";
    dataGridView1.Columns["Good"].HeaderText = "Good\n(millions)";

    //DataGridViewのCellPaintingイベントを購読します
    dataGridView1.CellPainting += DataGridView1_CellPainting;
}
```

Step 3: Configure BulletGraph control

1. Now create a global instance of the **BulletGraph** control and configure the general settings of the control. These settings will be applied to all the bullet graphs, which will be rendered in the **DataGridView** cells.
   Note that although we need to display the BulletGraph in each row of the unbound column, we require only one instance of the BulletGraph control. This is because as we have already rendered the BulletGraph control's image in the appropriate row, the same BulletGraph instance can be reconfigured to obtain the image for other bullet graphs.

   The code given below defines a method named '**ConfigureBulletGraph**' to instantiate the BulletGraph control and specify its general properties:

   - **Visual Basic**

```
Private Sub ConfigureBulletGraph()
    'BulletGraphコントロールを構成します
    _bulletGraph = New C1BulletGraph()
    _bulletGraph.Minimum = 0 '定量的スケールの開始値を設定します
    _bulletGraph.Maximum = 500 '定量的スケールの終了値を設定します
    _bulletGraph.GraphScale.ShowLabels = False 'スケールのラベルを非表示にします
    _bulletGraph.GraphScale.ShowMarks = False 'スケールの目盛りを非表示にします


    'BulletGraphコントロールをスタイルします
    _bulletGraph.BackColor = Color.White
    _bulletGraph.Styles.Ranges.Value.Color = Color.SteelBlue
    _bulletGraph.Styles.Ranges.Bar.Border.Color = Color.Gray
    _bulletGraph.Styles.Ranges.Bar.Border.LineStyle = C1GaugeBorderStyle.Solid
End Sub
```

   - **C#**

```
private void ConfigureBulletGraph()
{
    //BulletGraphコントロールを構成します
    _bulletGraph = new C1BulletGraph();
    _bulletGraph.Minimum = 0; //定量的スケールの開始値を設定します
    _bulletGraph.Maximum = 500; //定量的スケールの終了値を設定します
    _bulletGraph.GraphScale.ShowLabels = false; //スケールのラベルを非表示にします
    _bulletGraph.GraphScale.ShowMarks = false; //スケールの目盛りを非表示にします

    //BulletGraphコントロールをスタイルします
    _bulletGraph.BackColor = Color.White;
    _bulletGraph.Styles.Ranges.Value.Color = Color.SteelBlue;
    _bulletGraph.Styles.Ranges.Bar.Border.Color = Color.Gray;
    _bulletGraph.Styles.Ranges.Bar.Border.LineStyle = C1GaugeBorderStyle.Solid;
}
```

   Note: The **_bulletGraph** is declared as a private global variable of type **C1BulletGraph**.

Step 4: Rendering BulletGraph In DataGridView

1. To render the BulletGraph control as an image in the 'SalesV/STarget' column of the DataGridView, use the DataGridView's CellPainting event. In the **CellPainting** event handler, specify the values for the comparative measure, featured measure and qualitative ranges of the BulletGraph that needs to be drawn. We are setting these values in this event because for each bullet graph that is drawn in a specific row of the unbound column, these values will be different.
2. Once the BulletGraph is fully configured, retrieve an image of the BulletGraph control with the help of the **GetImage** method of **C1BulletGraph** class.
3. Draw the retrieved image in the DataGridView's cell using the DrawImage method of Graphics class as shown:
   - **Visual Basic**

```
Private Sub DataGridView1_CellPainting(ByVal sender As Object, ByVal e As DataGridViewCellPaintingEventArgs)
    If e.RowIndex >= 0 AndAlso DataGridView1.Columns(e.ColumnIndex).Name = "SalesV/STarget" Then
        _bulletGraph.Bounds = e.CellBounds
```

```
            _bulletGraph.Value = CDbl(_dt.Rows(e.RowIndex)("Sales")) '注目のメジャーの値を設定します
            _bulletGraph.Target = CDbl(_dt.Rows(e.RowIndex)("Target")) '比較メジャーの値を設定します
            _bulletGraph.Bad.To = CDbl(_dt.Rows(e.RowIndex)("Bad")) 'Good範囲の終了値を設定します
            _bulletGraph.Good.To = CDbl(_dt.Rows(e.RowIndex)("Good")) 'Bad範囲の終了値を設定します

            'BulletGraph画像を取得し、取得した画像をDataGridViewのセルに描画します
            Dim bulletGraphImg As Image = _bulletGraph.GetImage()
            e.Graphics.DrawImage(bulletGraphImg, e.CellBounds)
            e.Paint(e.CellBounds, DataGridViewPaintParts.Border)
            e.Handled = True
        End If
    End Sub
```

- **C#**

```
private void DataGridView1_CellPainting(object sender, DataGridViewCellPaintingEventArgs e)
{
    if (e.RowIndex >= 0 && dataGridView1.Columns[e.ColumnIndex].Name == "SalesV/STarget")
    {
        _bulletGraph.Bounds = e.CellBounds;
        _bulletGraph.Value = (double)_dt.Rows[e.RowIndex]["Sales"]; //注目のメジャーの値を設定します
        _bulletGraph.Target = (double)_dt.Rows[e.RowIndex]["Target"]; //比較メジャーの値を設定します
        _bulletGraph.Bad.To = (double)_dt.Rows[e.RowIndex]["Bad"]; //Good範囲の終了値を設定します
        _bulletGraph.Good.To = (double)_dt.Rows[e.RowIndex]["Good"]; //Bad範囲の終了値を設定します

        //BulletGraph画像を取得し、取得した画像をDataGridViewのセルに描画します
        Image bulletGraphImg = _bulletGraph.GetImage();
        e.Graphics.DrawImage(bulletGraphImg, e.CellBounds);
        e.Paint(e.CellBounds,DataGridViewPaintParts.Border);
        e.Handled = true;
    }
}
```

4. Call the **CreateDataTable, SetUpGrid** and **ConfigureBulletGraph** methods in the **Form1_Load** event handler.
    - **Visual Basic**

```
Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
    'DataGridViewのDataSourceとして使用されるDataTableを作成します
    CreateDataTable()
    '非連結列をDataGridViewに追加し、DataGridViewプロパティを設定します
    SetUpGrid()
    'BulletGraphコントロールの共通プロパティを設定します
    ConfigureBulletGraph()
End Sub
```

    - **C#**

```
private void Form1_Load(object sender, EventArgs e)
{
    //DataGridViewのDataSourceとして使用されるDataTableを作成します
    CreateDataTable();

    //非連結列をDataGridViewに追加し、DataGridViewプロパティを設定します
    SetUpGrid();

    //BulletGraphコントロールの共通プロパティを設定します
    ConfigureBulletGraph();
}
```

5. Run the application. Observe how the bullet graphs are getting displayed in the 'SalesV/STarget' column of the DataGridView control.

FlexGridとのBulletGraphの使用

This walkthrough explains the steps to display the **BulletGraph** control in FlexGrid cells. The BulletGraph control cannot be directly added to the FlexGrid cells but you can alternatively render it as an image in the FlexGrid cells. To achieve the same, you first need to add an unbound column to the FlexGrid and then use the CellPainting event to render the BulletGraph control as an image in the added unbound column.

Country	Sales (in millions)	Target (in millions)	Bad (in millions)	Good (in millions)	SalesV/STarget
US	$ 435	$ 452	$ 181	$ 347	
Germany	$ 246	$ 452	$ 53	$ 248	
Japan	$ 250	$ 493	$ 190	$ 282	
India	$ 177	$ 377	$ 194	$ 208	
China	$ 89	$ 383	$ 56	$ 333	
UK	$ 211	$ 360	$ 114	$ 297	
Denmark	$ 148	$ 391	$ 154	$ 324	
Indonesia	$ 4	$ 386	$ 61	$ 208	

Note: The BulletGraph control can also be rendered as an image in bound column of the FlexGrid control using similar approach.

1. Create a new **Windows Forms** application.
2. Drag and drop the FlexGrid control from the **Toolbox** onto your form. From the **Properties** window, set **its Dock** property to Fill.
3. Create a **DataTable** using the following method '**CreateDataTable**'. This data table will be used as the DataSource for the FlexGrid.
   - **Visual Basic**

```
Private Sub CreateDataTable()
    _dt = New DataTable()
    _dt.Columns.Add("Country", GetType(String))
    _dt.Columns.Add("Sales", GetType(Double))
    _dt.Columns.Add("Target", GetType(Double))
    _dt.Columns.Add("Bad", GetType(Double))
    _dt.Columns.Add("Good", GetType(Double))
    Dim countries As String() = {"US", "Germany", "Japan", "India", "China", "UK", "Denmark", "Indonesia"}
    Dim random As Random = New Random()

    For i As Integer = 0 To countries.Length - 1
        Dim totalSales As Integer = random.[Next](0, 500)
        Dim target As Integer = random.[Next](351, 499)
        Dim bad As Integer = random.[Next](50, 200)
        Dim good As Integer = random.[Next](201, 350)
        _dt.Rows.Add(New Object() {countries(i), totalSales, target, bad, good})
    Next
End Sub
```

   - **C#**

```
private void CreateDataTable()
{
    _dt = new DataTable();
    _dt.Columns.Add("Country", typeof(String));
    _dt.Columns.Add("Sales", typeof(Double));
    _dt.Columns.Add("Target", typeof(Double));
    _dt.Columns.Add("Bad", typeof(Double));
    _dt.Columns.Add("Good", typeof(Double));
    string[] countries = { "US", "Germany", "Japan", "India", "China", "UK", "Denmark", "Indonesia" };
    Random random = new Random();
    for (int i = 0; i < countries.Length; i++)
    {
        int totalSales = random.Next(0, 500);
        int target = random.Next(351, 499);
        int bad = random.Next(50, 200);
        int good = random.Next(201, 350);
        _dt.Rows.Add(new object[] { countries[i], totalSales, target, bad, good });
    }
}
```

   Note: The _dt is declared as a private global variable of type **DataTable**.

4. Create a method named '**SetUpGrid**' to add an unbound column to the FlexGrid control. This column will be used for showing the BulletGraph control as an image. Also, subscribe to the **OwnerDrawCell** event. This event will be used to render the BulletGraph control as an image in the unbound column 'SalesV/STarget' of the FlexGrid.
   - **Visual Basic**

```
Private Sub SetUpGrid()
    C1FlexGrid1.DataSource = _dt

    'FlexGridに非連結列を追加します
    'この列は、BulletGraphコントロールを画像として表示するために使用されます
    Dim unboundCol As Column = C1FlexGrid1.Cols.Add()
    unboundCol.Name = "SalesV/STarget"
    unboundCol.Caption = "SalesV/STarget"
    unboundCol.AllowEditing = False

    'FlexGridの追加プロパティを設定します(オプション)
```

```
    C1FlexGrid1.Cols.Fixed = 0
    C1FlexGrid1.AllowEditing = False
    C1FlexGrid1.Rows(0).StyleNew.Font = New Font(FontFamily.GenericSansSerif, 9.0F, FontStyle.Bold)
    C1FlexGrid1.Styles.Normal.Font = New Font(FontFamily.GenericSansSerif, 9.0F, FontStyle.Regular)
    C1FlexGrid1.Rows.DefaultSize = 40
    C1FlexGrid1.Cols.DefaultSize = 90
    C1FlexGrid1.ExtendLastCol = True
    C1FlexGrid1.Cols("Sales").Caption = "Sales" & vbLf & "(in millions)"
    C1FlexGrid1.Cols("Target").Caption = "Target" & vbLf & "(in millions)"
    C1FlexGrid1.Cols("Bad").Caption = "Bad" & vbLf & "(in millions)"
    C1FlexGrid1.Cols("Good").Caption = "Good" & vbLf & "(in millions)"

    For i As Integer = 0 To C1FlexGrid1.Cols.Count - 1
        C1FlexGrid1.Cols(i).Format = "$ #,##0"
        C1FlexGrid1.Cols(i).TextAlign = TextAlignEnum.CenterCenter
        C1FlexGrid1.Cols(i).TextAlignFixed = TextAlignEnum.CenterCenter
    Next

    C1FlexGrid1.DrawMode = C1.Win.C1FlexGrid.DrawModeEnum.OwnerDraw

    'OwnerDrawCellイベントを購読します
    AddHandler C1FlexGrid1.OwnerDrawCell, AddressOf C1FlexGrid1_OwnerDrawCell
End Sub
```

- **C#**

```
private void SetUpGrid()
{
    c1FlexGrid1.DataSource = _dt;

    //FlexGridに非連結列を追加します
    //この列は、BulletGraphコントロールを画像として表示するために使用されます
    Column unboundCol =c1FlexGrid1.Cols.Add();
    unboundCol.Name = "SalesV/STarget";
    unboundCol.Caption = "SalesV/STarget";
    unboundCol.AllowEditing = false;

    //FlexGridの追加プロパティを設定します(オプション)
    c1FlexGrid1.Cols.Fixed = 0;
    c1FlexGrid1.AllowEditing = false;
    c1FlexGrid1.Rows[0].StyleNew.Font = new Font(FontFamily.GenericSansSerif, 9f, FontStyle.Bold);
    c1FlexGrid1.Styles.Normal.Font= new Font(FontFamily.GenericSansSerif, 9f, FontStyle.Regular);
    c1FlexGrid1.Rows.DefaultSize = 40;
    c1FlexGrid1.Cols.DefaultSize = 90;
    c1FlexGrid1.ExtendLastCol = true;
    c1FlexGrid1.Cols["Sales"].Caption = "Sales\n(in millions)";
    c1FlexGrid1.Cols["Target"].Caption = "Target\n(in millions)";
    c1FlexGrid1.Cols["Bad"].Caption = "Bad\n(in millions)";
    c1FlexGrid1.Cols["Good"].Caption = "Good\n(in millions)";
    for (int i = 0; i < c1FlexGrid1.Cols.Count; i++)
    {
        c1FlexGrid1.Cols[i].Format = "$ #,##0";
        c1FlexGrid1.Cols[i].TextAlign = TextAlignEnum.CenterCenter;
        c1FlexGrid1.Cols[i].TextAlignFixed = TextAlignEnum.CenterCenter;
    }

    //OwnerDrawCellイベントを購読します
    c1FlexGrid1.DrawMode = C1.Win.C1FlexGrid.DrawModeEnum.OwnerDraw;
    c1FlexGrid1.OwnerDrawCell += C1FlexGrid1_OwnerDrawCell;
}
```

5. Create a method named '**ConfigureBulletGraph**' to specify the general settings for the BulletGraph control. These settings will be applied to all the bullet graphs, which will be rendered in the FlexGrid cells.

- **Visual Basic**

```
Private Sub ConfigureBulletGraph()
    'BulletGraphコントロールを構成します
    _bulletGraph = New C1BulletGraph()
    _bulletGraph.Minimum = 0 '定量的スケールの開始値を設定します
    _bulletGraph.Maximum = 500 '定量的スケールの終了値を設定します
    _bulletGraph.GraphScale.ShowLabels = False 'スケールのラベルを非表示にします
    _bulletGraph.GraphScale.ShowMarks = False 'スケールの目盛りを非表示にします

    'BulletGraphコントロールをスタイルします
    _bulletGraph.BackColor = Color.White
    _bulletGraph.Styles.Ranges.Value.Color = Color.SteelBlue
    _bulletGraph.Styles.Ranges.Bar.Border.Color = Color.Gray
    _bulletGraph.Styles.Ranges.Bar.Border.LineStyle = C1GaugeBorderStyle.Solid
End Sub
```

- **C#**

```
private void ConfigureBulletGraph()
{
    //BulletGraphコントロールを構成します
    _bulletGraph = new C1BulletGraph();
    _bulletGraph.Minimum = 0; //定量的スケールの開始値を設定します
    _bulletGraph.Maximum = 500; //定量的スケールの終了値を設定します
    _bulletGraph.GraphScale.ShowLabels = false; //スケールのラベルを非表示にします
    _bulletGraph.GraphScale.ShowMarks = false; //スケールの目盛りを非表示にします

    //BulletGraphコントロールをスタイルします
    _bulletGraph.BackColor = Color.White;
    _bulletGraph.Styles.Ranges.Value.Color = Color.SteelBlue;
    _bulletGraph.Styles.Ranges.Bar.Border.Color = Color.Gray;
```

```
    _bulletGraph.Styles.Ranges.Bar.Border.LineStyle = C1GaugeBorderStyle.Solid;
}
```

6. To render the BulletGraph control as an image in the 'SalesV/STarget' column of the FlexGrid, use the FlexGrid's **OwnerDrawCell** event.
   In the OwnerDrawCell event handler, specify the values for the comparative measure, featured measure and qualitative ranges of the BulletGraph that need to be drawn. Then, retrieve an image of the BulletGraph control with the help of **GetImage** method of the **C1BulletGraph** class. Finally, draw the retrieved image in the FlexGrid's cells using the **DrawImage** method of Graphics class as shown:
   - **Visual Basic**

```
Private Sub C1FlexGrid1_OwnerDrawCell(ByVal sender As Object, ByVal e As C1.Win.C1FlexGrid.OwnerDrawCellEventArgs)
    If e.Row > 0 AndAlso C1FlexGrid1.Cols(e.Col).Name = "SalesV/STarget" Then
        _bulletGraph.Bounds = e.Bounds
        _bulletGraph.Value = CDbl(_dt.Rows(e.Row - 1)("Sales")) '注目のメジャーの値を設定します
        _bulletGraph.Target = CDbl(_dt.Rows(e.Row - 1)("Target")) '比較メジャーの値を設定します
        _bulletGraph.Bad.To = CDbl(_dt.Rows(e.Row - 1)("Bad"))  'Good範囲の終了値を設定します
        _bulletGraph.Good.To = CDbl(_dt.Rows(e.Row - 1)("Good")) 'Bad範囲の終了値を設定します

        'BulletGraph画像を取得し、取得した画像をFlexGridのセルに描画します
        Dim bulletGraphImg As Image = _bulletGraph.GetImage()
        e.Graphics.DrawImage(bulletGraphImg, e.Bounds)
        e.DrawCell(DrawCellFlags.Border)
        e.Handled = True
    End If
End Sub
End Class
```

   - **C#**

```
private void C1FlexGrid1_OwnerDrawCell(object sender, C1.Win.C1FlexGrid.OwnerDrawCellEventArgs e)
{
    if (e.Row > 0 && c1FlexGrid1.Cols[e.Col].Name== "SalesV/STarget")
    {
        _bulletGraph.Bounds = e.Bounds;
        _bulletGraph.Value = (double)_dt.Rows[e.Row - 1]["Sales"]; //注目のメジャーの値を設定します
        _bulletGraph.Target = (double)_dt.Rows[e.Row - 1]["Target"]; //比較メジャーの値を設定します
        _bulletGraph.Bad.To = (double)_dt.Rows[e.Row - 1]["Bad"]; //Good範囲の終了値を設定します
        _bulletGraph.Good.To = (double)_dt.Rows[e.Row - 1]["Good"]; //Bad範囲の終了値を設定します

        //BulletGraph画像を取得し、取得した画像をFlexGridのセルに描画します
        Image bulletGraphImg = _bulletGraph.GetImage();
        e.Graphics.DrawImage(bulletGraphImg, e.Bounds);
        e.DrawCell(DrawCellFlags.Border);
        e.Handled = true;
    }
}
```

7. Call the **CreateDataTable**, **SetUpGrid** and **ConfigureBulletGraph** methods in the **Form1_Load** event handler.
   - **Visual Basic**

```
Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
    'FlexGridのDataSourceとして使用されるDataTableを作成します
    CreateDataTable()
    '非連結列をFlexGridに追加し、FlexGridプロパティを設定します
    SetUpGrid()
    'BulletGraphコントロールの共通プロパティを設定します
    ConfigureBulletGraph()
End Sub
```

   - **C#**

```
private void Form1_Load(object sender, EventArgs e)
{
    //FlexGridのDataSourceとして使用されるDataTableを作成します
    CreateDataTable();

    //非連結列をFlexGridに追加し、FlexGridプロパティを設定します
    SetUpGrid();

    //BulletGraphコントロールの共通プロパティを設定します
    ConfigureBulletGraph();
}
```

8. Run the application. Observe how the bullet graphs are getting displayed in the 'SalesV/STarget' column of the FlexGrid control.