# Adventures with Infragistics TreeMap chart

Although there are lots of visualization technics, the treemap chart is a good way to visualize hierarchical data. Let’s say you have the following table:

|  |  |  |
| --- | --- | --- |
| Category | Name | Count |
| Warning | Temperature increased | 30 |
| Warning | Disk space is low | 20 |
| Security Issue | Intrusion detected | 5 |
| Information | Print job completed | 10 |

# Before we start

Create a WinForm project and place an UltraWinChart control onto it. Change the chart type to TreeMap and we’re ready to go.

# Creating the datasource

The above table could be represented with the following function in VB.NET:

Private Function GetData() As DataTable

Dim returnValue As New DataTable()

With returnValue

With .Columns

.Add("Category", GetType(String))

.Add("Name", GetType(String))

.Add("Count", GetType(Double))

End With

With .Rows

.Add(New Object() {"Warning", "Temperature increased", 30})

.Add(New Object() {"Warning", "Disk space is low", 20})

.Add(New Object() {"Security Issue", "Intrusion detected", 5})

.Add(New Object() {"Information", "Print job completed.", 10})

End With

.AcceptChanges()

End With

Return returnValue

End Function

And, in Form load event we could use this as the datasource:

Sub Form1\_Load(sender As System.Object, e As System.EventArgs) Handles MyBase.Load

With Me.UltraChart1

'Set the Data Source of the chart

With .Data

.DataSource = GetData()

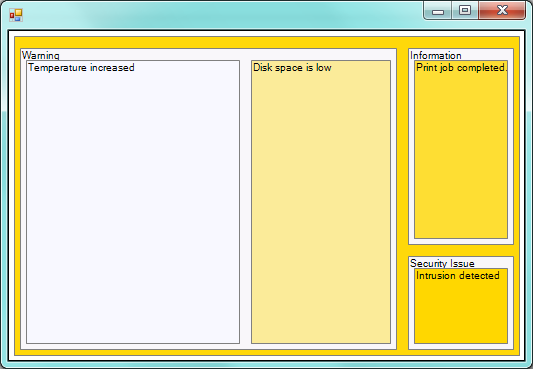
.DataBind()

End With

End With

End Sub

Assuming you have a TreeMap chart on Form, the result will be the following:



So far, so good. We can see all the values. But let’s change the values… For example, let “Print job completed” be 20,000.

Private Function GetData() As DataTable

Dim returnValue As New DataTable()

With returnValue

With .Columns

.Add("Category", GetType(String))

.Add("Name", GetType(String))

.Add("Count", GetType(Double))

End With

With .Rows

.Add(New Object() {"Warning", "Temperature increased", 30})

.Add(New Object() {"Warning", "Disk space is low", 20})

.Add(New Object() {"Security Issue", "Intrusion detected", 5})

.Add(New Object() {"Information", "Print job completed.", 20000})

End With

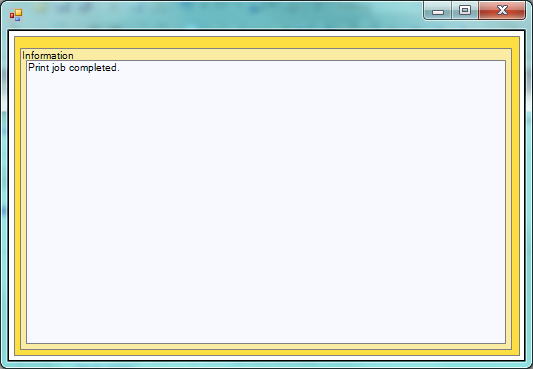
.AcceptChanges()

End With

Return returnValue

End Function

In this case the treemap chart will output the following:



Where are the other categories? Actually it was rendered but since the difference between them is too large, they are not visible.

Other chart types have logarithmic scales but TreeMap is not one of them! Rest of this article will show you how to visualize that kind of data with TreeMap correctly.

# Transforming the data into logarithmic scale

Using the following function will transform the values into logarithmic scale:

Private Function TransformData(input As DataTable) As DataTable

For Each dr As DataRow In input.Rows

dr("Count") = Math.Log10(dr("Count"))

Next

Return input

End Function

And the form\_load will be like this:

Sub Form1\_Load(sender As System.Object, e As System.EventArgs) Handles MyBase.Load

'Get the dummy data

ChartData = GetData()

With Me.UltraChart2

'Set the Data Source of the chart

With .Data

.DataSource = TransformData(ChartData)

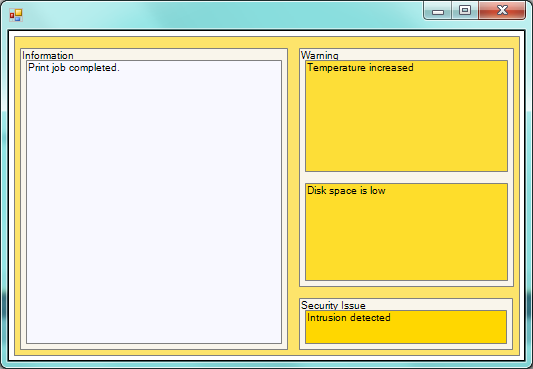
.DataBind()

End With

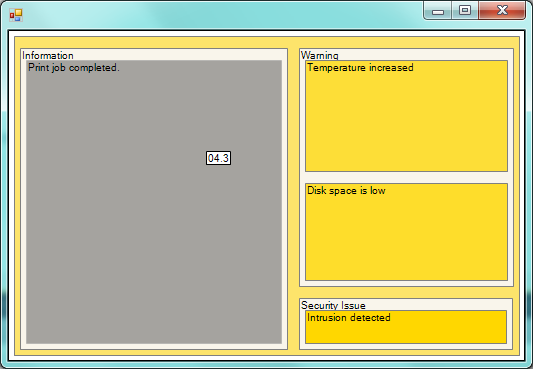
End With

End Sub

When we run the application, it will output this result:



Looks like we’ve done it! But wait… what about the tooltips? Mouse over “Print job completed” and the toolti will display 04.3!



What is 04.3? Well, it is Log10 of 20,000 of course! Hmm… what about categories? Mouse over “Warning” and the tooltip will display 2.78 (sum of “Temperature increased” + “Disk space is low”). This introduces another challenge… Changing the tooltips to display correct values. But this is a little harder than you might think.

# IRenderLabel interface

In order to display custom tooltips, you need to implement an interface called “IRenderLabel.” The following code snippet is an example:

Private Class MyLabelRenderer

Implements Infragistics.UltraChart.Resources.IRenderLabel

Public Overloads Function ToString(

ByVal context As System.Collections.Hashtable) As String Implements

Infragistics.UltraChart.Resources.IRenderLabel.ToString

'Return the custom string here

Return "This is a custom tooltip"

End Function

End Class

In order to utilize this interface, we must tell Ultrachart to change the way it generates the tooltips. This is done as follows:

Sub Form1\_Load(sender As System.Object, e As System.EventArgs) Handles MyBase.Load

With Me.UltraChart1

'Set the Data Source of the chart

With .Data

.DataSource = TransformData(ChartData)

.DataBind()

End With

Dim myLabelHashTable As New Hashtable()

myLabelHashTable.Add("MyLabel", New MyLabelRenderer(ChartData))

'Attach the Hash table to the Chart

.LabelHash = myLabelHashTable

'Assign the custom label to the ToolTip

With .Tooltips

.Format = Infragistics.UltraChart.[Shared].Styles.TooltipStyle.Custom

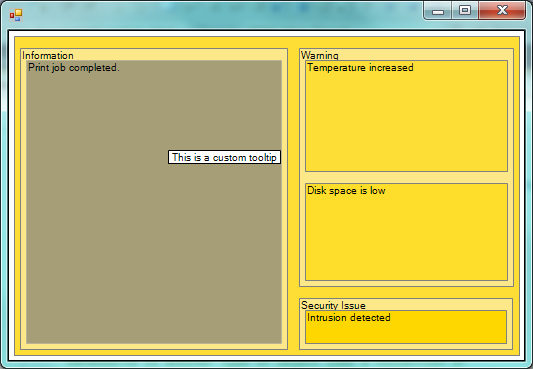
.FormatString = "<MyLabel>"

End With

End With

End Sub

When you run the application, the following will be shown when you mouse over chart tiles:



Well, it worked but it is useless at this point.

# Accessing data values in IRender implementation

In order to access data values and labels we can use the hashtable interface uses. Change MyLabelRenderer class so that it will look like this:

Private Class MyLabelRenderer

Implements Infragistics.UltraChart.Resources.IRenderLabel

Private ReadOnly data As DataTable

Public Sub New(ByVal chartData As DataTable)

data = chartData

End Sub

'The ToString method will take the custom label <MyLabel> and

'return the string to display in the control.

Public Overloads Function ToString(

ByVal context As System.Collections.Hashtable) As String

Implements Infragistics.UltraChart.Resources.IRenderLabel.ToString

Dim seriesLabel As String = context("SERIES\_LABEL")

Dim itemLabel As String = context("ITEM\_LABEL")

Dim dataValue As Double = context("DATA\_VALUE")

'Return the string

Return String.Format("series\_label: {0}" & vbCrLf &

"item\_label: {1}" & vbCrLf &

"data\_value: {2}",

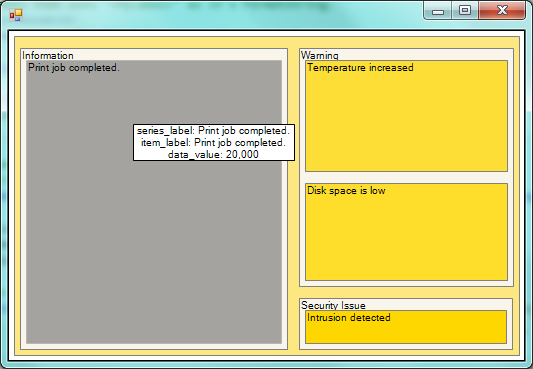
seriesLabel, itemLabel,

Math.Pow(10, dataValue).ToString("#,#"))

End Function

End Class

And run the application. The tooltips will look (like) correct now:



But wait, what happens when you mouse over Warning category? It will display 600! Which is not the sum of 20+30! In order to solve this problem, we need to go back to Math101: Sum of Logarithmic values!

LOG10(20) + LOG10(30) = 2.778

But

POWER10(2.778) = 600

You need to add them up one by one. Maybe there is a better way but my Math 101 knowledge is 20 years old! ☺

# The solution

If you take a close look at the constructor of the MyLabelRenderer class, you’ll notice we pass a datatable into it. Using the datatable, we can access the correct values. Here is the final code to display everything correctly:

Private Class MyLabelRenderer

Implements Infragistics.UltraChart.Resources.IRenderLabel

Private ReadOnly data As DataTable

Private ReadOnly ht As New Hashtable

Public Sub New(ByVal chartData As DataTable)

data = chartData

For Each dr As DataRow In data.Rows

ht(dr("Name").ToString) = dr("Category").ToString

Next

End Sub

Public Overloads Function ToString(

ByVal context As

System.Collections.Hashtable) As String Implements

Infragistics.UltraChart.Resources.IRenderLabel.ToString

Dim seriesLabel As String = context("SERIES\_LABEL")

Dim itemLabel As String = context("ITEM\_LABEL")

Dim dataValue As Double = context("DATA\_VALUE")

If ht.ContainsKey(itemLabel) Then

'we have a leaf node

Return String.Format("series\_label: {0}" & vbCrLf &

"item\_label: {1}" & vbCrLf & "data\_value: {2}",

seriesLabel, itemLabel,

Math.Pow(10, dataValue).ToString("#,#"))

Else

'we have a category

Dim total As Double = 0.0

For Each dr As DataRow In data.Select("Category='" & itemLabel & "'")

total += Math.Pow(10, dr("Count"))

Next

Return String.Format("series\_label: {0}" & vbCrLf &

"item\_label: {1}" & vbCrLf & "data\_value: {2}",

seriesLabel, itemLabel, total.ToString("#,#"))

End If

End Function

End Class

# Final Notes

This article is not a general purpose solution. Using aifferent data shape might need additional steps.