Windows Forms – UserControl, Drawing, Drag and Drop, Printing

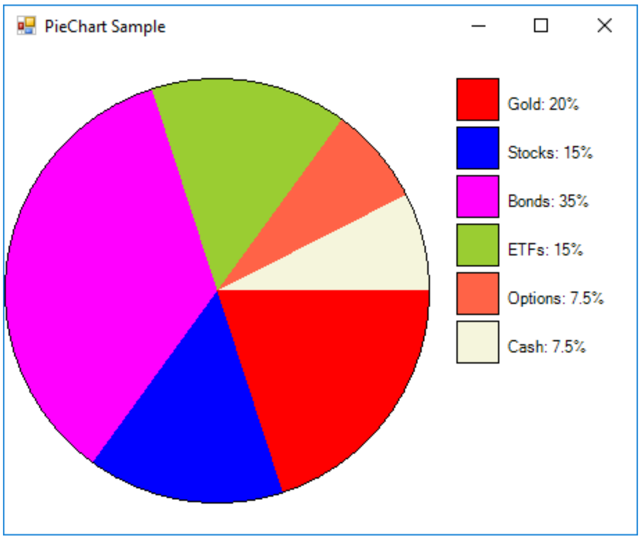
Contents

1. Chart Control

2. UserControl

Activity

C# Sample code available at http://online.ase.ro – “PieChartGraphicsSample” Sample



1. Create a new project with the name “PieChartGraphicsSample”
2. Add a new class “PieChartCategory”, defined as follows

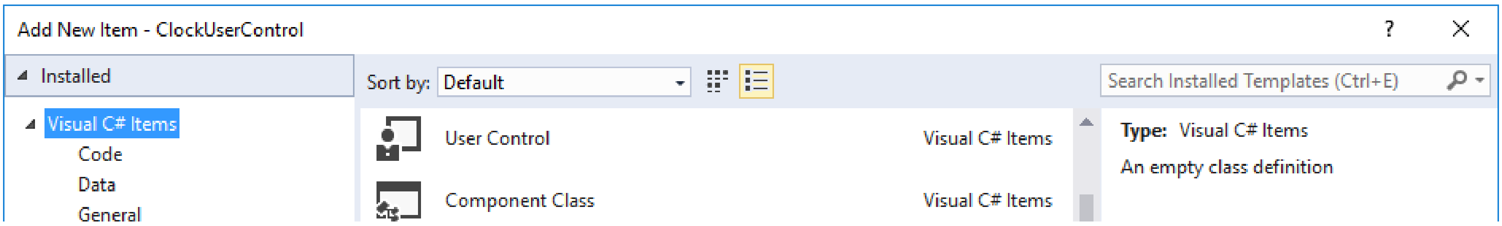
```
internal class PieChartCategory
{
      public string Description { get; set; }

      public float Percentage { get; set; }
```

```
        public Color Color { get; set; }

        public PieChartCategory(string description, float percent, Color color)
        {
                Description = description;
                Percentage = percent;
                Color = color;
        }
}
```

1. Add a new UserControl and name it “PieChartControl”



Add New Item - ClockUserControl

Sort by: Default

Search Installed Templates (Ctrl+E)

Installed

Visual C# Items
    Code
    Data
    General

User Control — Visual C# Items

Component Class — Visual C# Items

Type: Visual C# Items

An empty class definition

2. Add the “Data “ property in the “PieChartControl” class

```
private PieChartCategory[] _data;
public PieChartCategory[] Data {
        get { return _data; }
        set
        {
                if(_data == value)
                        return;

                _data = value;

                //trigger the Paint event
                Invalidate();
        }
}
```

2. Drawing

3. The Graphics class provides methods for drawing objects to the display device.

Activity

4. Modify the constructor of the “PieChartControl” class as follows

```
public PieChartControl()
{
        InitializeComponent();

        //redraws if resized
        ResizeRedraw = true;

        //Default data
        Data = new[]
        {
                new PieChartCategory("Category 1", 20, Color.Red),
            new PieChartCategory("Category 2", 80, Color.Blue)
        };
}
```

5. Handle the “Paint” event for the “PieChartControl” as follows

```
private void PieChartControl_Paint(object sender, PaintEventArgs e)
{
        //width reserved for displaying the legend
        int legendWidth = 150;

        //get the drawing context
        Graphics graphics = e.Graphics;
        //get the drqwing area
        Rectangle clipRectangle = e.ClipRectangle;

        //compute the maximum radius
        float radius = Math.Min(clipRectangle.Height, clipRectangle.Width - legendWidth) /
(float)2;

        //determine the center of the pie
        int xCenter = (clipRectangle.Width - legendWidth) / 2;
        int yCenter = clipRectangle.Height / 2;

        //determine the x and y coordinate of the pie
        float x = xCenter - radius;
        float y = yCenter - radius;

        //determine the width and the height
        float width = radius * 2;
        float height = radius * 2;

        //draw the pie sectors
        float percent1 = 0;
        float percent2 = 0;
        for (int i = 0; i < Data.Length; i++)
        {
                if (i >= 1)
                        percent1 += Data[i - 1].Percentage;

                percent2 += Data[i].Percentage;

                float angle1 = percent1 / 100 * 360;
                float angle2 = percent2 / 100 * 360;

                Brush b = new SolidBrush(Data[i].Color);

                graphics.FillPie(b, x, y, width, height, angle1, angle2 - angle1);
        }

        //draw the pie contour
        Pen pen = new Pen(Color.Black);
        graphics.DrawEllipse(pen, x, y, width, height);

        //draw the chart legend
        float xpos = x + width + 20;
        float ypos = y;
        for (int i = 0; i < Data.Length; i++)
        {
                Brush b = new SolidBrush(Data[i].Color);
```

```
            graphics.FillRectangle(b, xpos, ypos, 30, 30);
            graphics.DrawRectangle(pen, xpos, ypos, 30, 30);

            Brush b2 = new SolidBrush(Color.Black);

            graphics.DrawString(Data[i].Description + ": " + Data[i].Percentage + "%", 
Font, b2,
                  xpos + 35, ypos + 12);

            ypos += 35;
      }
}
```

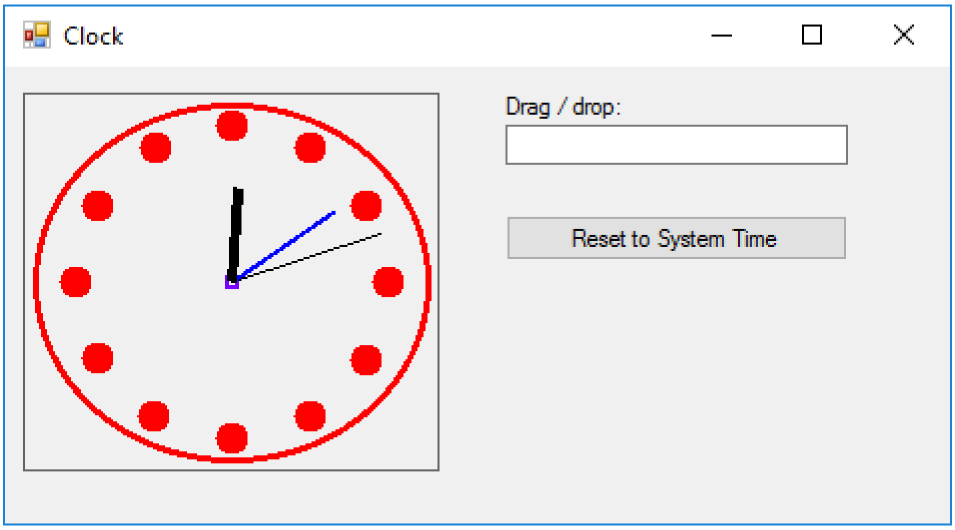
6. Add the "PieChartControl" to the "MainForm" (using the Toolbox)
7. Handle the "Load" event for the "MainForm" as follows

```
private void MainForm_Load(object sender, System.EventArgs e)
{
      PieChartCategory[] pieCategories = {
            new PieChartCategory("Gold", 20, Color.Red),
            new PieChartCategory("Stocks", 15, Color.Blue),
            new PieChartCategory("Bonds", 35, Color.Magenta),
            new PieChartCategory("ETFs", 15, Color.YellowGreen),
            new PieChartCategory("Options", (float) 7.5, Color.Tomato),
            new PieChartCategory("Cash", (float) 7.5, Color.Beige)
      };

      pieChartControl1.Data = pieCategories;
}
```

Activity

C# Sample code available at http://online.ase.ro – "ClockUserControlSample" Sample



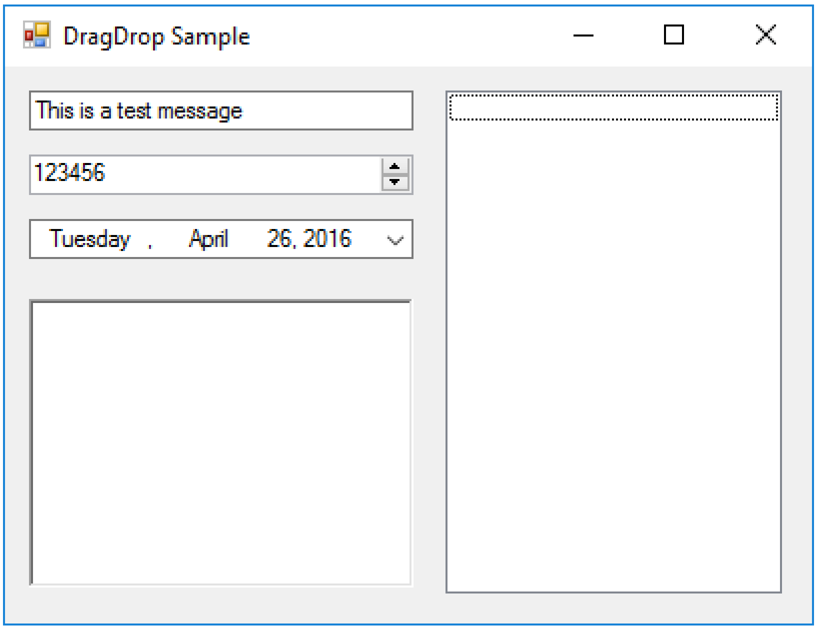
3. Drag and Drop

- Further reading: https://msdn.microsoft.com/en-us/library/aa984430%28v=vs.71%29.aspx

Activity

C# Sample code available at http://online.ase.ro – "DragDropSample" Sample

1. Create a new project with the name "DragDropSample".
2. Create the UI shown bellow.

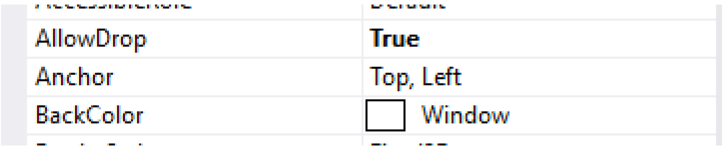


This is a test message

123456

Tuesday , April 26, 2016

3. Set the "AllowDrop" property of the ListView to true.



AllowDrop	**True**
Anchor	Top, Left
BackColor	□ Window

4. Handle the "MouseDown" event for the TextBox as follows.

```
textBox1.DoDragDrop(textBox1.Text, DragDropEffects.Copy);
```

5. Handle the "DragEnter" event for the ListView as follows.

```
 // Display some information about the DragDrop information in the
// richTextBox1 control to show some of the information available.
richTextBox1.Text = "Allowed Effect: " + e.AllowedEffect +
"\r\nAvailable Formats:\r\n";

// Data may be available in more than one format, so loop through
// all available formats and display them in richTextBox1.
foreach (string availableFormat in e.Data.GetFormats(true))
{
      richTextBox1.Text += "\t" + availableFormat + "\r\n";
}

// This control will use any dropped data to add items to the listbox.
// Therefore, only data in a text format will be allowed.  Setting the
// autoConvert parameter to true specifies that any data that can be
// converted to a text format is also acceptable.
if (e.Data.GetDataPresent(DataFormats.Text, true))
{
      // Some controls in this sample allow both Copy and Move effects.
      // If a Move effect is allowed, this implementation assumes a Move
      // effect unless the CTRL key was pressed, in which case a Copy
      // effect is assumed.  This follows standard DragDrop conventions.
      if ((e.AllowedEffect & DragDropEffects.Move) == DragDropEffects.Move &&
(e.KeyState & CtrlKey) != CtrlKey)
      {
            // Show the standard Move icon.
            e.Effect = DragDropEffects.Move;
      }
      else
      {
            // Show the standard Copy icon.
            e.Effect = DragDropEffects.Copy;
      }
}
```

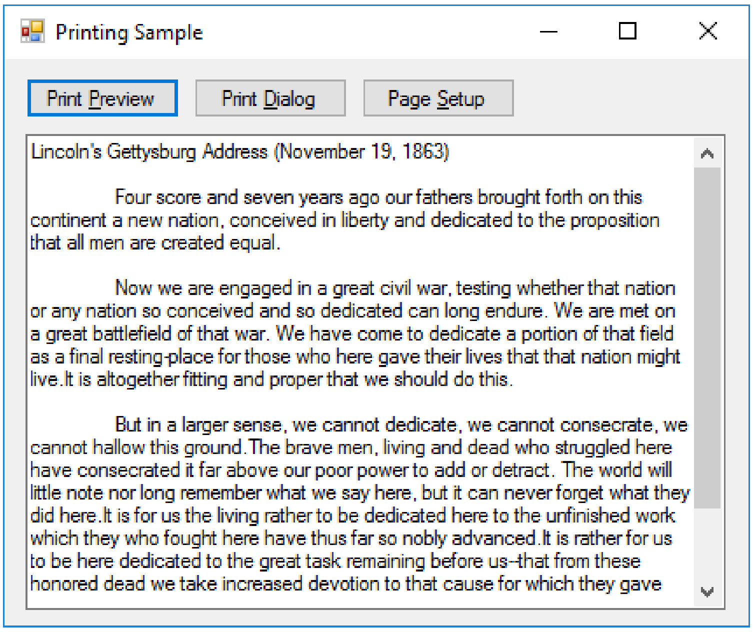
6. Handle the “DragDrop” event for the ListView as follows.

```
 /// <summary>
/// The DragDrop event of the target control fires when a drop actually occurs over
/// the target control.  This is where the data being dragged is actually processed.
///
/// This event will fire only if the AllowDrop property of the target control has
/// been set to true.
/// </summary>
/// <param name="sender">The source of the event.</param>
/// <param name="e">A DragEventArgs that contains the event data.</param>
private void listBox1_DragDrop(object sender, DragEventArgs e)
{
      if (e.Data.GetDataPresent(DataFormats.Text, true))
      {
            // Create the list item using the data provided by the source control.
            listBox1.Items.Add(e.Data.GetData(DataFormats.Text));
      }
}
```

4. Printing

Activity

C# Sample code available at http://online.ase.ro – “PrintingSample” Sample



Lincoln's Gettysburg Address (November 19, 1863)

Four score and seven years ago our fathers brought forth on this continent a new nation, conceived in liberty and dedicated to the proposition that all men are created equal.

Now we are engaged in a great civil war, testing whether that nation or any nation so conceived and so dedicated can long endure. We are met on a great battlefield of that war. We have come to dedicate a portion of that field as a final resting-place for those who here gave their lives that that nation might live.It is altogether fitting and proper that we should do this.

But in a larger sense, we cannot dedicate, we cannot consecrate, we cannot hallow this ground.The brave men, living and dead who struggled here have consecrated it far above our poor power to add or detract. The world will little note nor long remember what we say here, but it can never forget what they did here.It is for us the living rather to be dedicated here to the unfinished work which they who fought here have thus far so nobly advanced.It is rather for us to be here dedicated to the great task remaining before us-that from these honored dead we take increased devotion to that cause for which they gave

5. Further reading

5.1. Unit Testing

Further reading: link