



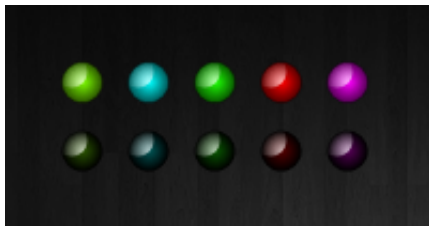
A Simple Vector-Based LED User Control

Steve Marsh, 6 Apr 2013 [CPOL](#)

★★★★★ 4.92 (63 votes)

The LEDBulb is a .NET user control for Windows Forms that emulates an LED light. Its purpose is to provide a sleek looking representation of an LED light that is sizable, has a transparent background and can be set to different colors.

[Download source - 245.91 KB](#)



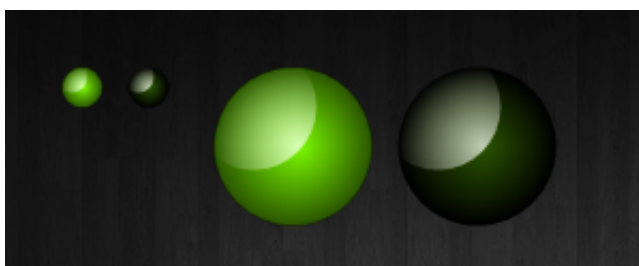
Introduction

The **LEDBulb** is a .NET user control for Windows Forms that emulates an LED light with two states On and Off. The purpose of the control is to provide a sleek looking representation of an LED light that is sizable, has a transparent background and can be set to different colors.

```
LedBulb bulb = new LedBulb();  
bulb.Size = new Size(25, 25);  
bulb.Color = Color.LawnGreen;  
bulb.On = true;  
this.Controls.Add(bulb);
```

Sizeable

I recently needed to add a custom user control to a Window Forms project of mine that would represent an LED bulb commonly seen on household electronics. After a quick search, most of the existing controls I found were ugly and used static images, which means they wouldn't scale well. This control uses the **System.Drawing.Drawing2D** namespace to draw a vector image so that it not only looks clean but will also scale to any size without affecting the image quality.



Rendering the control starts with drawing a solid circle with our background color. Then we draw a radial gradient over it with our highlight color, transitioning to transparent. To draw the reflection, we draw another radial gradient, from white to transparent, but in a smaller rectangle shifted up and left. To prevent the white gradient from showing up outside the bulb, we set the clip parameter to the bounds of our original circle.

```
// Fill in the background circle
g.FillEllipse(new SolidBrush(darkColor), drawRectangle);

// Draw the glow gradient
GraphicsPath path = new GraphicsPath();
path.AddEllipse(drawRectangle);
PathGradientBrush pathBrush = new PathGradientBrush(path);
pathBrush.CenterColor = lightColor;
pathBrush.SurroundColors = new Color[] { Color.FromArgb(0, lightColor) };
g.FillEllipse(pathBrush, drawRectangle);

// Set the clip boundary to the edge of the ellipse
GraphicsPath gp = new GraphicsPath();
gp.AddEllipse(drawRectangle);
g.SetClip(gp);

// Draw the white reflection gradient
GraphicsPath path1 = new GraphicsPath();
path1.AddEllipse(whiteRectangle); // a smaller rectangle set to the top left
PathGradientBrush pathBrush1 = new PathGradientBrush(path);
pathBrush1.CenterColor = Color.FromArgb(180, 255, 255, 255);
pathBrush1.SurroundColors = new Color[] { Color.FromArgb(0, 255, 255, 255) };
g.FillEllipse(pathBrush1, whiteRect);
```

Transparent Background

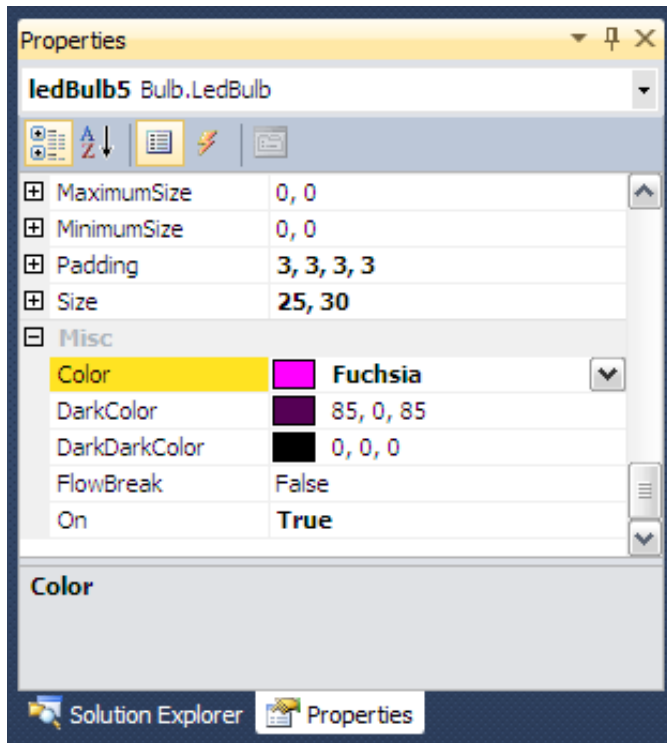
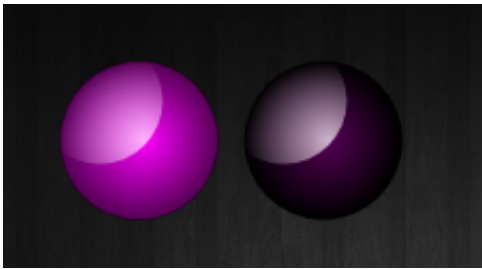
It is important that this control has a transparent background. To do this, we need to add a command to the constructor.

```
SetStyle(ControlStyles.DoubleBuffer
    | ControlStyles.AllPaintingInWmPaint
    | ControlStyles.ResizeRedraw
    | ControlStyles.UserPaint
    | ControlStyles.SupportsTransparentBackColor, true
);
```

Customizable Colors

You can customize the color of the LED display by setting the **Color** property. Setting this property also automatically calculates two other Properties **ColorDark** and **ColorDarkDark** which are used for gradients when the control is drawn. These colors are calculated using the **ControlPaint** class, a **Control** helper class in the **System.Windows.Forms** namespace.

```
this.DarkColor = ControlPaint.Dark(_this.Color);
this.DarkDarkColor = ControlPaint.DarkDark(_this.Color);
```



Blink

The control has a blink method that will make the control start blinking. You pass the number of milliseconds between blinks. To turn the blinking off just call the blink function again passing 0 as the argument.

```
led.Blink(2000); // Slow blink  
led.Blink(500);  // Fast blink  
led.Blink(0);    // Turn off blinking
```



Points of Interest

This control uses double-buffering by drawing its image to an off-screen bitmap first, then sending the final rendering to the client. Double buffering creates a smoother drawing of the control and avoids flicker when moving or re-sizing. The code to accomplish this is below.

```
protected override void OnPaint(PaintEventArgs e){  
    // Create an offscreen graphics object for double buffering  
    Bitmap offScreenBmp = new Bitmap(this.ClientRectangle.Width,  
                                     this.ClientRectangle.Height);  
    System.Drawing.Graphics g = Graphics.FromImage(offScreenBmp);  
    g.SmoothingMode = SmoothingMode.HighQuality;  
  
    // Render the control to the off-screen bitmap  
    drawControl(g);  
  
    // Draw the image to the screen  
    e.Graphics.DrawImageUnscaled(offScreenBmp, 0, 0);  
}
```

Links

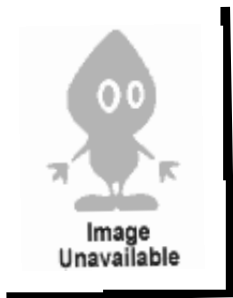
- [Double Buffering](#)
- [Lighten and Darken Colors in .NET](#)

License

This article, along with any associated source code and files, is licensed under [The Code Project Open License \(CPOL\)](#)

Share

About the Author



Steve Marsh

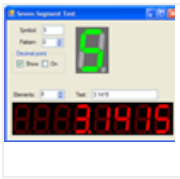
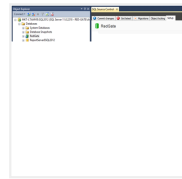
Software Developer (Senior)
United States 

Stephen Marsh has over 10 years of experience developing enterprise applications built on the .Net framework. He specializes in building expert systems that serve the financial industry.

You may also be interested in...




LED vu Meter User Control

Learn to Better Monetize
Apps by Rewarding UsersSeven-segment LED Control
for .NETPutting databases under
version control with Redgate's
SQL Source Control

A C# LED matrix display

Beyond RDBMS: A Guide To
NoSQL Databases

Comments and Discussions

 **29 messages** have been posted for this article Visit <http://www.codeproject.com/Articles/114122/A-Simple-Vector-Based-LED-User-Control> to post and view comments on this article, or click [here](#) to get a print view with messages.

[Permalink](#) | [Advertise](#) | [Privacy](#) | [Terms of Use](#) | [Mobile](#)
Web03 | 2.8.150819.1 | Last Updated 5 Apr 2013

Select Language | ▼

Article Copyright 2010 by Steve Marsh
Everything else Copyright © [CodeProject](#), 1999-2015