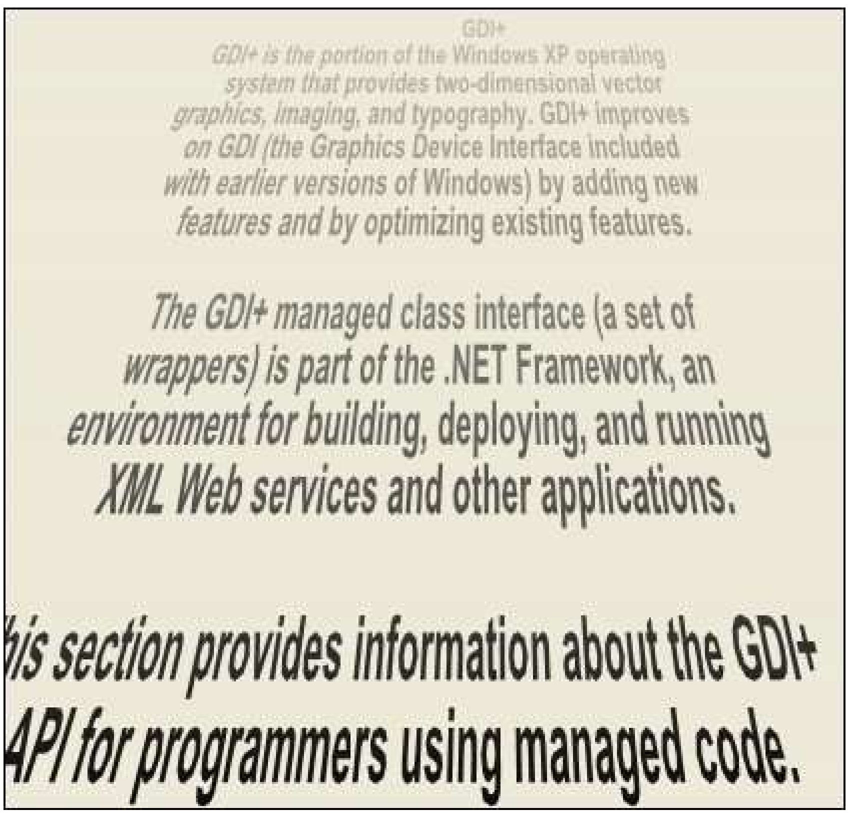
Star Wars style text scroller

alexey N

10 Jul 2007 CPOL

Text scroller control with 3D-look (like in the intro of the Star Wars movies)

- **Download demo - 6.8 KB**
- **Download source - 14.7 KB**



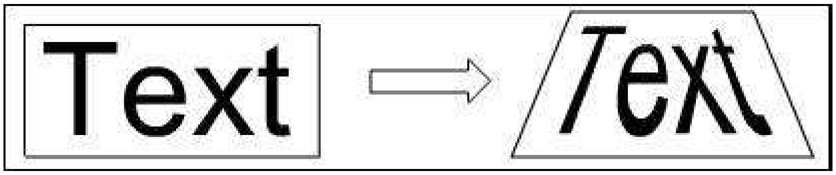
Introduction

You can find text scrollers in many programs, especially in their About dialogs. In most cases, it's a simple colored text that moves up. In this article, I attempted to create something unusual. I decided to write an "outgoing" text component, which looks like a 3D effect such as in the intro to the Star Wars movies.

GDI+ provides many easy-to-use objects and functions. Using these functions, you can do something special without much trouble. Before writing this component with the help of GDI+, I tried to create it with GDI. This required much more time and resulted in ten times more code. It seemed to be a hard task, but later I recreated this same component in GDI+. That was easy. After that, I decided to write this article to demonstrate some features of GDI+.

How we can do this

We can create this "outgoing" effect by transforming all points of text from rectangular to trapezoidal shape:



To animate our transformed text, we'll use an offset variable and timer. On the timer tick, we'll change this variable and repaint the control.

The code

The most useful code is situated in the `Paint` event handler. First of all, we need to enable anti-aliasing for better quality:

```
e.Graphics.SmoothingMode = SmoothingMode.AntiAlias;
```

After that, we erase background to clean the previous frame:

```
e.Graphics.FillRectangle(new SolidBrush(this.BackColor),
    this.ClientRectangle);
```

Then we create a `GraphicsPath` object and fill it with visible lines of text depending on the offset:

```
GraphicsPath path = new GraphicsPath();
for (int i = m_text.Length - 1; i >= 0; i--)
{
    Point pt = new Point((int)
        ((this.Width - e.Graphics.MeasureString(m_text[i],
        this.Font).Width) / 2), (int)(m_scrollingOffset +
        this.Height - (m_text.Length - i) * this.Font.Size));

    // Adds visible lines to path.
    if ((pt.Y + this.Font.Size > 0) && (pt.Y < this.Height))
        path.AddString(m_text[i], new FontFamily("Arial"),
        (int)FontStyle.Bold, this.Font.Size,
        pt, StringFormat.GenericDefault);
}
```

After that, we transform our `GraphicsPath` from rectangle to trapezoid:

```
path.Warp(new PointF[4]
{
    new PointF(this.Width / 3, 0),
    new PointF(this.Width * 2 / 3, 0),
    new PointF(0, this.Height),
    new PointF(this.Width, this.Height)
},
new RectangleF(this.ClientRectangle.X,
this.ClientRectangle.Y, this.ClientRectangle.Width,
this.ClientRectangle.Height),
null, WarpMode.Perspective);
```

The text is now ready. Next, we need to draw it and dispose of the `GraphicsPath` object:

```
// Draws wrapped path.
e.Graphics.FillPath(new SolidBrush(this.ForeColor), path);
path.Dispose();
```

To make control more realistic, we can draw some "fog" using `LinearGradientBrush` with transparent color:

```
// Draws fog effect with help of gradient brush with alpha colors.
using (Brush br = new LinearGradientBrush(new Point(0, 0),
    new Point(0, this.Height),
    Color.FromArgb(255, this.BackColor), Color.FromArgb(0,
    this.BackColor)))
    {
        e.Graphics.FillRectangle(br, this.ClientRectangle);
    }
```

Using the code

The `Scroller` class represents an easy-to-use component with customizable font, background color, text color and, of course, text content. You can simply copy the `Scroller` class to your project to use it. Also, you can create a separate class library for the `Scroller` class.

Properties of the Scroller control

This control has the following properties:

- `TextToScroll` – this text will be separated into lines at the `\n` symbol
- `BackColor` – color of background
- `ForeColor` – color of text
- `Interval` - delay in milliseconds between frames for controlled scrolling speed
- `TextFont` - font that is used to draw; units must be set to `GraphicsUnit.Pixel`
- `TopPartSizePercent` - top part size of text in percent of control width

Methods of the Scroller control

- `Start()` – starts the animation from the beginning
- `Stop()` – stops the animation

Points of interest

When I was creating this control, I noticed that the `GraphicsPath` class can help you in situations where you need some specific transformation of graphics objects, including point transformations.

Disclaimer

You can use this code in any type of project, free or commercial. If you do, please add a link to this article in your code comments or the About dialog box.

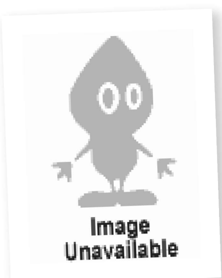
Thanks for reading and thanks to all who have helped me improve this article.

History

- 9 July, 2007 - First updated version posted
  - Fixed bug with text offset: `Font Units` is now set to `GraphicsUnit.Pixel`
  - Added cycle scrolling feature
  - Added `Interval` property
  - Added `TextFont` property
  - Added `TopPartSizePercent` property
- 16 May, 2007 - Original version posted

License

About the Author



alexey N
Software Developer
Russian Federation

No Biography provided

Comments and Discussions

43 messages have been posted for this article Visit **https://www.codeproject.com/Articles/18820/Star-Wars-style-text-scroller** to post and view comments on this article, or click **here** to get a print view with messages.