Render Color Emoji in a NinjaScript Indicator

Generated by Gemini 2.5 Pro Ai

Yes, it is absolutely possible to render full-color emoji characters within a NinjaTrader 8 environment using methods like Draw.Text() and Draw.TextFixed() inside your OnRender() loop.

The reason this works so well in NinjaTrader 8 is that the platform is built on **Windows Presentation Foundation (WPF)**. WPF's rendering engine uses DirectX and is fully capable of handling modern font technologies, including the embedded color bitmaps and vector graphics used by color emojis.

This is a significant advantage over older platforms like NinjaTrader 7, which was based on WinForms and would have struggled with this, likely only rendering the black-and-white outline of the emoji.

How to Render Color Emoji in a NinjaScript Indicator

Here is a practical example of how to draw color emojis on your chart within a custom indicator. The key is to use a font that contains color glyphs. On any modern Windows system, the "Segoe UI Emoji" font is the standard for this.

You can place this code inside a custom indicator's OnRender() method.

C#

```
// This code goes inside your custom NinjaScript indicator class

private SharpDX.DirectWrite.TextFormat a_custom_font;
private SharpDX.Direct2D1.Brush a_text_color;

protected override void OnStateChange()
{
    if (State == State.SetDefaults)
    {
```

```
        // ... your other defaults
        Name = "Emoji Drawing Example";
        IsOverlay = true;
    }
    else if (State == State.Configure)
    {
        // Pre-configure the resources we will use in OnRender
        // This is much more efficient than creating them in the OnRender loop
    }
}

protected override void OnRender(ChartControl chartControl, ChartScale chartScale)
{
    // The text containing emoji characters
    string myEmojiText = "Signal: 🚀 ▲";
    string anotherEmojiText = "Target Hit: ✅ 🎯 ";

    // --- Method 1: Let WPF's Font Fallback Do the Work (Usually Works) ---
    // Here we use a standard font like Arial. WPF is smart enough to find the
    // emoji characters in a different font (like Segoe UI Emoji) if needed.
    // This is convenient but less explicit.
    SimpleFont standardFont = new SimpleFont("Arial", 16);

    Draw.TextFixed(this, "TagA", myEmojiText, TextPosition.TopLeft,
        ChartControl.Properties.ChartText, // a standard color
        standardFont,
        Brushes.Transparent, Brushes.Transparent, 0);


    // --- Method 2: Explicitly Use an Emoji Font (Most Reliable Method) ---
    // This is the best practice for guaranteed color emoji rendering.
    SimpleFont emojiFont = new SimpleFont("Segoe UI Emoji", 16);

    Draw.TextFixed(this, "TagB", anotherEmojiText, TextPosition.TopRight,
        ChartControl.Properties.ChartText,
        emojiFont, // Explicitly provide the emoji font
        Brushes.Transparent, Brushes.Transparent, 0);
```

```
}
```

Key Considerations in the NinjaTrader Environment

1. **NinjaTrader 8 vs. NinjaTrader 7:** This capability is specific to NinjaTrader 8 and later versions due to the move to the WPF platform. Attempting this in NT7 would not yield color results.
2. **Font Selection is Key:** While WPF's font fallback is very good, the most reliable method is to explicitly create a SimpleFont object with "Segoe UI Emoji" as the font family. This removes any ambiguity for the renderer.
3. **Performance:** Creating new SimpleFont() inside the OnRender() loop is inefficient as OnRender is called very frequently. While the example above does this for simplicity, in a real-world, performance-sensitive indicator, you should instantiate your SimpleFont objects once in OnStateChange() under State.Configure or State.DataLoaded and reuse them in OnRender.
4. **No Special Libraries Needed:** You do not need any third-party libraries like SkiaSharp for this. NinjaTrader's built-in drawing tools are sufficient because they are wrappers around the powerful WPF rendering engine.

In summary: Yes, you can. Use Draw.Text() or Draw.TextFixed() and ensure you are either relying on WPF's robust font fallback or, for best results, explicitly specifying "Segoe UI Emoji" in your SimpleFont object.