Gewöhnliche Tools für ungewöhnliche Aufgaben

Oder: Canvas kann was

1 Einleitung

Füllstand anzeigen, Bewegungen und Positionen darstellen – wenn kein passendes Steuerelement in der WPF-Werkzeugkiste vorhanden ist, ist Kreativität gefragt. Bevor ihr nach komplizierten Lösungen sucht, können Euch folgende Tipps Anregungen liefern, Standardwerkzeuge nützlich „umzufunktionieren".

2 Slider

Das Einfachste voran: Der simple *Slider* kann sehr gut zur Livedarstellung veränderlicher Parameter verwendet werden.

Als Beispiel soll folgende Problemstellung dienen: ein Spindelmotor bewegt ein Target. Bewegung und aktuelle Position des Targets sollen live dargestellt werden. Gegeben sind der Nullpunkt, der maximale Spindelweg und die aktuelle Position. Diese Aufgabe lässt sich ganz einfach mit einem Slider lösen:

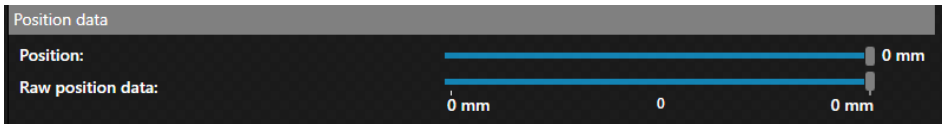

Abbildung 1

Slider zur Livedarstellung von Parametern mit XAML-Quellcode

```
XAML:
<Slider
       x:Name="PositionSlider"
       IsHitTestVisible="False"
       Maximum="{Binding MaxPosition}"
       Minimum="{Binding NullPosition}"
       Value="{Binding CurrentPosition, Mode=OneWay}"
/>
<Label
       Content="{Binding Value, ElementName=PositionSlider}"
       ContentStringFormat="{}{0} mm"
/>
```

Daraus ergibt sich sowohl eine graphische Anzeige der Position über den Slider, als auch eine nummerische Anzeige über das Label. Da dies aber ein reines Anzeigeelement werden soll, wird dem Slider seine Interaktivität über `IsHitTestVisible="False"` genommen. Diese Eigenschaft regelt die mausbezogenen Ereignisse und macht das Steuerelement beim Setzen des Wertes auf „False" unempfindlich gegenüber den Mauseingaben.

3 ProgressBar

Die simple *ProgressBar* ist nicht nur als Fortschrittsanzeige zu gebrauchen, sondern auch für die Darstellungen von Füllständen gut geeignet. Fast ohne Modifikationen lässt sich eine ProgressBar dafür einsetzen, z.B. so, wie nachfolgend dargestellt.

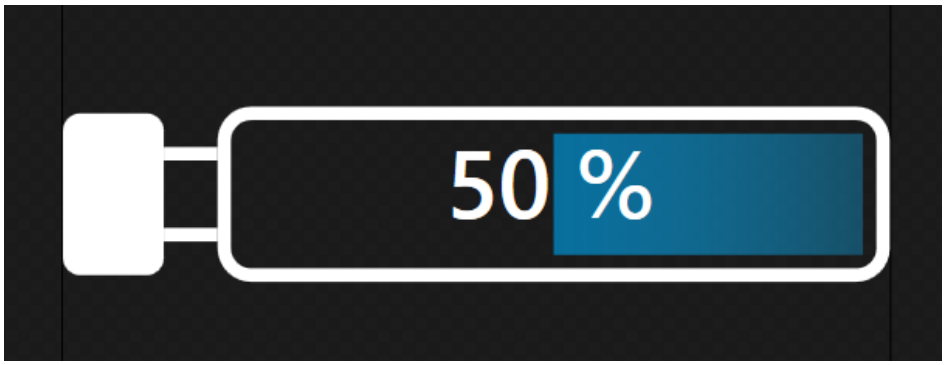

Abbildung 2

ProgressBar zur Fortschrittsanzeige mit XAML-Quellcode

XAML:

```
<ProgressBar
       x:Name="LevelProgress"
       Background="{x:Null}"
       BorderBrush="{x:Null}"
       Value="{Binding FillLevel}"
       RenderTransformOrigin="0.5,0.5">
    <ProgressBar.RenderTransform>
        <TransformGroup>
           <RotateTransform Angle="180"/>
        </TransformGroup>
    </ProgressBar.RenderTransform>
</ProgressBar>
```

Dabei wird die *ProgressBar* mit `RotateTransform Angle="180"` um 180° gedreht.

Mit etwas mehr Aufwand lassen sich auch komplexere, animierte Füllstandanzeigen kreieren.

4 Canvas

An sich ist das Canvas ein eher weniger spektakuläres Panel in WPF. Es macht praktisch nichts allein und lässt dem Anwender die volle Kontrolle über die Position der zugehörigen Elemente. Die Positionierung innerhalb des Canvas erfolgt über das XY-Koordinatensystem.

Als letztes Beispiel für heute soll eine Joystickanzeige für eine Kalibrierung dienen. Gegeben sind die A/D-Counts für die XYZ-Joystickachsen, Maximum, Minimum und der Nullpunkt. Nun ist die Zeit für den Super-Canvas gekommen.

Die XYZ-Joystickwerte können in das Canvas-Koordinatensystem transformiert werden, wie das untenstehende Beispiel mit Quellcode zeigt.

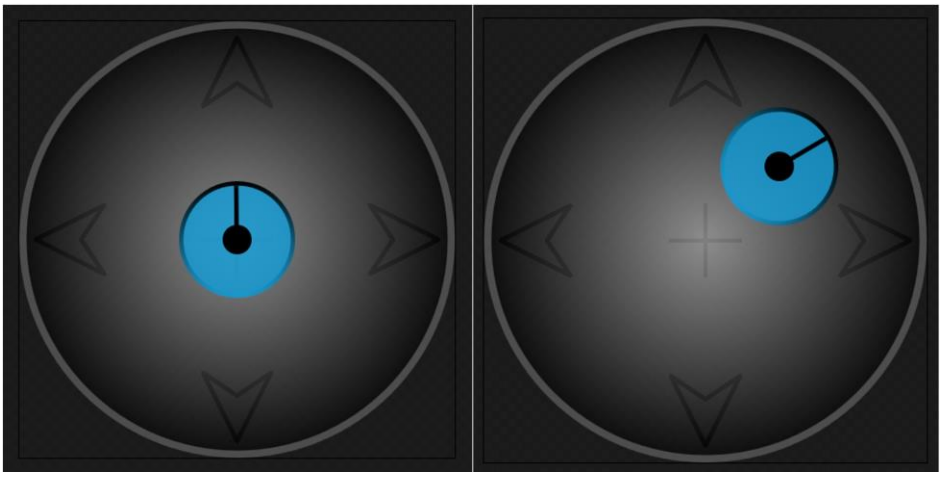

Abbildung 3

Nutzung des Canvas-Elements zur Joystickanzeige mit C#-Quellcode

```
/// <summary>
/// Transform the joystick counts to the canvas coordinates
/// </summary>
/// <param name="val">Current value</param>
/// <param name="min">Min. value for joystick</param>
/// <param name="max">Max. value for joystick</param>
/// <param name="mid">Zero value for joystick</param>
/// <returns>coordinates between -100 and 100</returns>
public static double JoyTransform(double val, uint min, uint max, uint mid)
{
    double rv = 0;
    const double percent = 100;
    uint pRange = max - mid;            // positiv range
    uint nRange = mid - min;            // negativ range

    if ((val >= mid) && (pRange != 0))      // check pRange for division
    {
        rv = ((val - mid) / pRange) * percent;
    }
    else if ((val < mid) && (nRange != 0))  // check nRange for division
    {
        rv = ((mid - val) / nRange) * percent * (-1);
    }
    else
    {
        rv = 0;
    }

    return Math.Round(rv,0);
}
```

Die resultierenden XY-Koordinaten können dann über die Canvas-Eigenschaften `Canvas.Left` und `Canvas.Top` gesetzt werden. Die Z-Achse lässt sich mit Hilfe der `RotateTransform`-Operation darstellen:

```
<Canvas.RenderTransform>
    <TransformGroup>
        <RotateTransform Angle="{Binding ZAxsis}"/>
    </TransformGroup>
</Canvas.RenderTransform>
```

Abbildung 4

XAML-Quellcode für die Joystickanzeige

So, das war’s mit unseren Tipps zur Nutzung der WPF-Elemente für heute.

Dmitri Batan
Junior-Softwareentwickler der CogniMed GmbH