Cross Media Publishing: Höhenprofil in SVG aus GPX

60600 | Maximilian Schön
60763 | Niklas Amann

Aufgabenstellung

- Geben Sie das Höhe-Zeit-Profil einer GPX-Datei in SVG aus und stellen Sie es im Browser dar.
- Verwenden Sie unterschiedliche Farben für die einzelnen Tracksegmente.
- Die Farbe soll über eine XSLT-2-Funktion bestimmt werden.

GPS Exchange Format (GPX)

- Auf XML-Standard basierender Datenformat zur Speicherung von Geodaten.
- Track Element (<trk>) schließt Trackpoints (<trkpt>) ein, die sich als Linienzug darstellen lassen.
- <trkseg>-Tag fasst Punkte zu Abschnitte zusammen.
- Neues Tracksegment beginnt bpsw. wenn GPS-Empfang verloren geht oder pausiert wird.

```
<trk>
    <name>GraphHopper Track</name>
    <trkseg>
        <trkpt lat="49.933075" lon="11.588903">
            <ele>347.6</ele>
            <time>2018-04-10T13:46:07Z</time>
        </trkpt>
        <trkpt lat="49.93335" lon="11.589356">
            <ele>347.4</ele>
            <time>2018-04-10T13:46:23Z</time>
        </trkpt>
    </trkseg>
</trk>
```

Festlegungen

- X-Achse beschreibt die Zeit
- Y-Achse beschreibt die Höhe
- Anzeige des Koordinatensystems bleibt im Browser unabhängig von der Menge an Werten gleich groß
- Mehr als ein Tracksegment => Tracksegmente werden eingefärbt
- Nur ein Tracksegment => Strecke zwischen Punkten wird eingefärbt

Namensräume

```
<?xml version="1.0" encoding="UTF-8"?>

    <!-- xpath-default-namespace für Default-Namespace der GraphHopper-Datei -->
    <!-- xpath-default-namespace="http://www.topografix.com/GPX/1/1" -->
<xsl:stylesheet version="2.0" exclude-result-prefixes="fn xs math fn_ms"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:fn="http://www.w3.org/2005/xpath-functions"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:math="http://exslt.org/math"
    xmlns:fn_ms="https://www.hs-aalen.de/"
    xmlns="http://www.w3.org/2000/svg"
    xpath-default-namespace="http://www.topografix.com/GPX/1/1"
>
```

Berechnung der Maße

```
<!-- Abmessungen der SVG-Grafik -->
<xsl:variable name="svgWidth" select="1200"/>
<xsl:variable name="svgHeight" select="600"/>

<!-- Abmessungen des Höhenprofils -->
<xsl:variable name="profileWidth" select="$svgWidth - 2 * $padding"/>
<xsl:variable name="profileHeight" select="$svgHeight - 2 * $padding"/>

 <!-- ermittle frühesten und spätesten Zeitpunkt sowie Zeitdifferenz -->
<xsl:variable name="minTime" select="fn:min(//trkpt/xs:dateTime(time))"/>
<xsl:variable name="maxTime" select="fn:max(//trkpt/xs:dateTime(time))"/>
<xsl:variable name="difTime" select="($maxTime - $minTime) div xs:dayTimeDuration('PT1S')"/>

<!-- ermittle minimale und maximale Höhe sowie Höhendifferenz -->
<xsl:variable name="minEle" select="fn:min(//trkpt/ele)"/>
<xsl:variable name="maxEle" select="fn:max(//trkpt/ele)"/>
<xsl:variable name="difEle" select="$maxEle - $minEle"/>

<!-- berechne Maßstab für x- und y-Achse -->
<xsl:variable name="xScale" select="$profileWidth div $difTime"/>
<xsl:variable name="yScale" select="$profileHeight div $difEle"/>

<!-- Abstand für Achsen und Beschriftung -->
<xsl:variable name="padding" select="100"/>
<xsl:variable name="offset" select="10"/>
```

Root-Template

```
<xsl:template match="/">
    <svg>
        <xsl:attribute name="width" select="$svgWidth"/>
        <xsl:attribute name="height" select="$svgHeight"/>
        <!-- Rahmen -->
        <rect x="0" y="0" fill="none" stroke="black">
            <xsl:attribute name="width" select="$svgWidth"/>
            <xsl:attribute name="height" select="$svgHeight"/>
        </rect>
        <!-- erstelle Hilfslinien mit Beschriftung -->
        <xsl:call-template name="labels"/>
        <xsl:call-template name="lines"/>
        <!-- verarbeite Segmente -->
        <xsl:apply-templates select="//trkseg"/>
    </svg>
</xsl:template>
```

Achsenbeschriftungen

```
<!-- erstelle Achsenbeschriftung -->
<xsl:template name="labels">
    <!-- Beschriftung x-Achse (Zeit) -->
    <text text-anchor="middle" dy="1em">
        <xsl:attribute name="x" select="$padding"/>
        <xsl:attribute name="y" select="$svgHeight - $padding + 2 * $offset"/>
        <xsl:value-of select="fn_ms:time($minTime)"/>
    </text>
    <text text-anchor="middle" dy="3em" font-size="0.7em">
        <xsl:attribute name="x" select="$padding"/>
        <xsl:attribute name="y" select="$svgHeight - $padding + 2 * $offset"/>
        <xsl:value-of select="fn_ms:date($minTime)"/>
    </text>
    <text text-anchor="middle" dy="1em">
        <xsl:attribute name="x" select="$svgWidth div 2"/>
        <xsl:attribute name="y" select="$svgHeight - $padding + 2 * $offset"/>
        <xsl:value-of select="fn_ms:time(($maxTime - $minTime) div 2 + $minTime)"/>
    </text>
    <text text-anchor="middle" dy="3em" font-size="0.7em">
        <xsl:attribute name="x" select="$svgWidth div 2"/>
        <xsl:attribute name="y" select="$svgHeight - $padding + 2 * $offset"/>
        <xsl:value-of select="fn_ms:date(($maxTime - $minTime) div 2 + $minTime)"/>
    </text>
```

Ausgabe der Zeit

```
<!-- Zeit im Format hh:mm:ss aus dateTime filtern -->
<xsl:function name="fn_ms:time">
    <xsl:param name="dateTime" as="xs:dateTime"/>
    <!-- ziehe Zeit aus dateTime -->
    <xsl:variable name="time" select="fn:substring-before(fn:substring-after(xs:string($dateTime), 'T'), 'Z')"/>
    <!-- Sekundenbruchteile abschneiden -->
    <xsl:choose>
        <xsl:when test="fn:contains($time, '.')">
            <xsl:sequence select="fn:substring-before($time, '.')"/>
        </xsl:when>
        <xsl:otherwise>
            <xsl:sequence select="$time"/>
        </xsl:otherwise>
    </xsl:choose>
</xsl:function>
```

Ausgabe des Datums

```
<!-- Datum im Format dd.mm.yyyy aus dateTime filtern -->
<xsl:function name="fn_ms:date">
    <xsl:param name="dateTime" as="xs:dateTime"/>
    <!-- ziehe Datum aus dateTime -->
    <xsl:variable name="date" select="fn:substring-before(xs:string($dateTime), 'T')"/>
    <!-- ändere Format -->
    <xsl:sequence select="
            fn:concat (
                fn:substring-after(fn:substring-after($date, '-'), '-'),
                '.',
                fn:substring-before(fn:substring-after($date, '-'), '-'),
                '.',
                fn:substring-before($date, '-')
            )
    "/>
</xsl:function>
```

Zeichnen der Hilfslinien

```
<!-- erstelle Hilfslinien -->
<xsl:template name="lines">
    <xsl:variable name="stroke-width" select="0.5"/>
    <!-- horizontale Hilfslinien -->
    <line fill="none" stroke="black">
        <xsl:attribute name="stroke-width" select="$stroke-width"/>
        <xsl:attribute name="x1" select="$padding - $offset"/>
        <xsl:attribute name="y1" select="$padding"/>
        <xsl:attribute name="x2" select="$svgWidth - $padding + $offset"/>
        <xsl:attribute name="y2" select="$padding"/>
    </line>
    <line fill="none" stroke="black">
        <xsl:attribute name="stroke-width" select="$stroke-width"/>
        <xsl:attribute name="x1" select="$padding - $offset"/>
        <xsl:attribute name="y1" select="$svgHeight div 2"/>
        <xsl:attribute name="x2" select="$svgWidth - $padding + $offset"/>
        <xsl:attribute name="y2" select="$svgHeight div 2"/>
    </line>
    <line fill="none" stroke="black">
        <xsl:attribute name="stroke-width" select="$stroke-width"/>
        <xsl:attribute name="x1" select="$padding - $offset"/>
        <xsl:attribute name="y1" select="$svgHeight - $padding"/>
        <xsl:attribute name="x2" select="$svgWidth - $padding + $offset"/>
        <xsl:attribute name="y2" select="$svgHeight - $padding"/>
    </line>
```

trkseg-Template

```
<xsl:template match="trkseg">
    <xsl:choose>
        <!-- mehrere trkseg - übergebe zufällige Farbe an alle Kindknoten -->
        <xsl:when test="preceding-sibling::trkseg | following-sibling::trkseg">
            <xsl:apply-templates select="child::trkpt[fn:position() != fn:last()]">
                <xsl:with-param name="color" select="fn_ms:random-color()"/>
            </xsl:apply-templates>
        </xsl:when>
        <!-- einzelnes trkseg - übergebe keine Farbe -->
        <xsl:otherwise>
            <xsl:apply-templates select="child::trkpt[fn:position() != fn:last()]"/>
        </xsl:otherwise>
    </xsl:choose>
</xsl:template>
```

trkpt-Template

```
<!-- Darstellung der Strecke mit Linien -->
<xsl:template match="trkpt">
    <!-- wird keine Farbe übergeben, erstelle zufällige Farbe -->
    <xsl:param name="color" select="fn_ms:random-color()"/>
    <line fill="none">
        <xsl:attribute name="stroke" select="$color"/>
        <xsl:attribute name="x1" select="(xs:dateTime(time) - $minTime) div xs:dayTimeDuration('PT1S')
                                         * $xScale + $padding"/>
        <xsl:attribute name="y1" select="$profileHeight - (ele - $minEle) * $yScale  + $padding"/>
        <xsl:attribute name="x2" select="(xs:dateTime(following-sibling::trkpt[1]/time) - $minTime)
                                         div xs:dayTimeDuration('PT1S') * $xScale  + $padding"/>
        <xsl:attribute name="y2" select="$profileHeight - (following-sibling::trkpt[1]/ele - $minEle)
                                         * $yScale  + $padding"/>
    </line>
</xsl:template>
```

Random-Color-Funktion

```
<!-- erstelle zufälligen RGB-Farbcode -->
<xsl:function name="fn_ms:random-color">
    <xsl:sequence select="
        fn:concat (
            'rgb(',
            xs:integer(math:random() * 256) mod 256,
            ', ',
            xs:integer(math:random() * 256) mod 256,
            ', ',
            xs:integer(math:random() * 256) mod 256,
            ')'
        )
    "/>
</xsl:function>
```