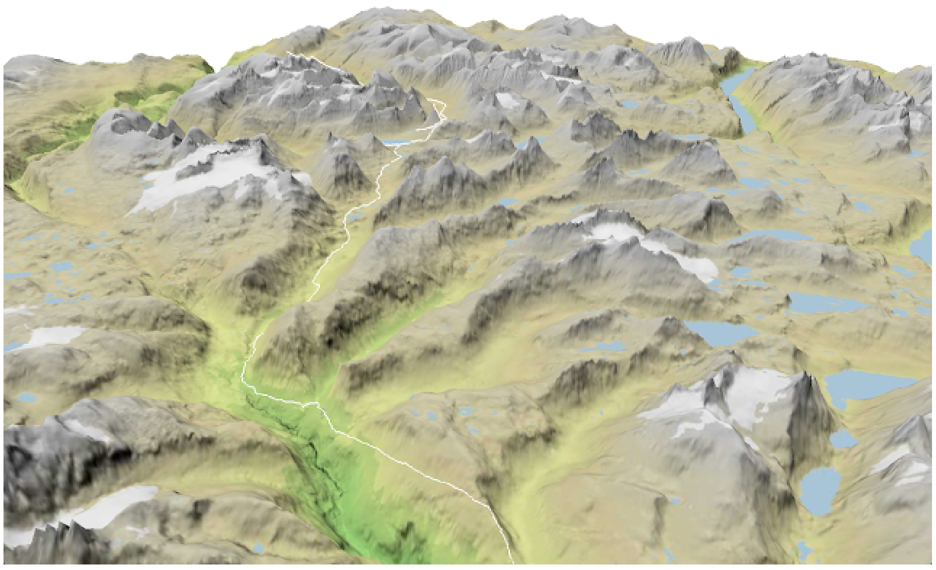
SATURDAY, 2 NOVEMBER 2013

Showing GPS tracks in 3D with three.js and d3.js

Posted by Bjørn Sandvik

How can you visualize a GPS track with three.js? The tricky part was the get the projection right so the GPS track would line up with my terrain map of Jotunheimen. With the help of D3.js, I was able to do what I wanted.

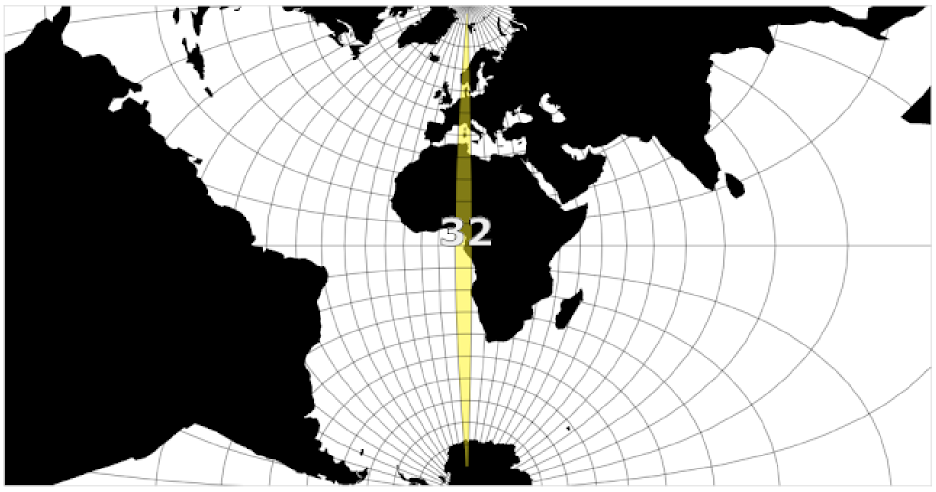


I'm going to use the same GPS track that I've used previously with Leaflet. My plan was to convert this track to GeoJSON, but all the tools I tried didn't include the elevation values for my track. Instead of building my own converter, I decided to parse the GPX file directly in JavaScript. GPX is an XML format, and D3.js supports reading XML files with the d3.xml method:

```
d3.xml('jotunheimen-track.gpx', 'application/xml', gpxParser);
```

The gpxParser function (shown below) is called when the GPX file is loaded, passing in the root element of the XML document. You can parse this document with the JavaScript XML/DOM access facilities.

My GPS track is in geographic coordinates (latitude and longitude) and I need to project the points to be able to show the track on my map of Jotunheimen. Three.js don't have any support for different map projections, but we can use the great projection support of D3.js. My map is in the UTM 32 projection, and I've played with UTM in D3.js previously.



SEARCH THIS BLOG

Search

NEW NAME - SAME CONTENT!

This blog was previously named "Thematic Mapping Blog" :-)

SOCIAL MAPPING

Twitter
Instagram
LinkedIn
Google+

FOLLOW BY EMAIL

Email address... Submit

BLOG ARCHIVE

▶ 2016 (4)
▶ 2015 (7)
▶ 2014 (21)
▼ 2013 (16)
  ▶ December (1)
  ▼ November (1)
    Showing GPS tracks in 3D with three.js and d3.js
  ▶ October (4)
  ▶ September (1)
  ▶ July (1)
  ▶ June (6)
  ▶ April (2)
▶ 2012 (30)
▶ 2011 (3)
▶ 2010 (7)
▶ 2009 (10)
▶ 2008 (48)

LAST COMMENTS

I created a TIN using the default settings, includ... - 4/16/2018 - Anonymous
Is this still working with newer versions of Leafl... - 3/21/2018 - Anonymous
No, I had to cancel it due to external factors. Wi... - 2/19/2018 - Bjørn Sandvik
Hey Karan, just use threejs plane or objects as &amp;q... - 1/15/2018 - skywalker
Hello there, I believe your web site could possibl... - 11/7/2017 - Anonymous

The Universal Transverse Mercator (UTM) projection is based on the cylindrical Transverse Mercator projection, which is supported by D3.js. This is how I define the projection:

```
var terrainSize = 60; // 60 x 60 km

var projection = d3.geo.transverseMercator()
    .translate([terrainSize / 2, terrainSize / 2])
    .scale(terrainSize * 106.4)
    .rotate([-9, 0, 0])
    .center([-0.714, 61.512]);
```

The terrainSize can be any size, but I'm using 60 as the area of Jotunheimen I'm mapping is 60 x 60 km. It took me some time to find the values used in the projection configuration methods:

- translate: The pixel coordinates of the projection's center.
- scale: The scale factor corresponds linearly to the distance between projected points. I figured out that this was terrainSize multiplied by 106.4 for my example, but I don't know why exactly 106.4...
- rotate: I'm rotating the projection minus 9 degrees longitude which corresponds to the central meridian of the UTM 32 zone.
- center: The longitude and latitude of the projection's center. This is the same as the center of my map (8.286, 61.512), except that the longitude position is relative to the central meridian of UTM 32 (8.286 - 9 = -0.714).

With these numbers sorted, my GPS track was lining up correctly on my map. But how to render the track in three.js? I'm adding a vertex for each track point to a THREE.Geometry object. This is the code of my GPX parser:

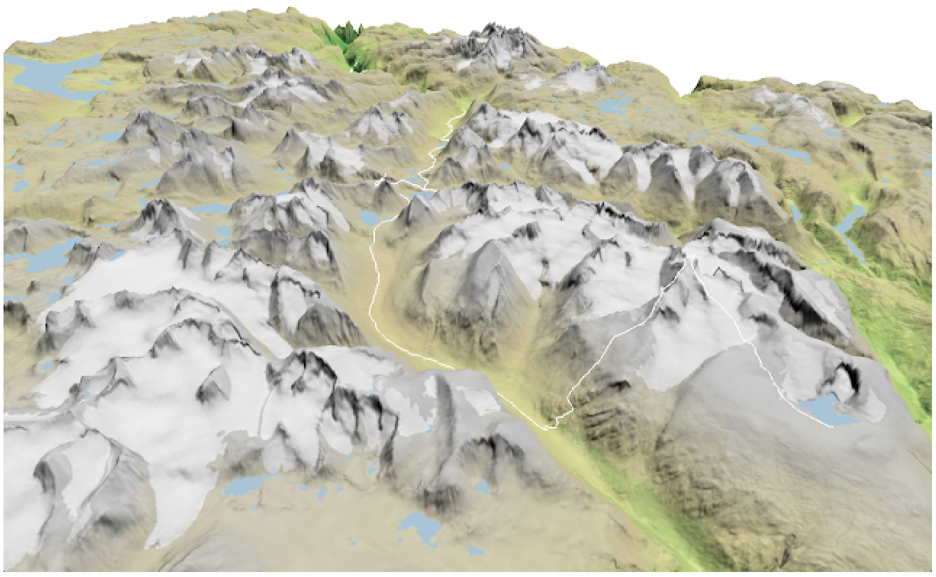
```
function gpxParser(gpx) {
  var tracks = gpx.getElementsByTagName('trk'),
      geometry = new THREE.Geometry();

  for (i = 0; i < tracks.length; i++) {
    var points = tracks[i].getElementsByTagName('trkpt')

    for (x = 0; x < points.length; x++) {
      var point = points[x],
          ele = parseInt(point.getElementsByTagName('ele')
[0].firstChild.nodeValue),
          lat = parseFloat(point.getAttribute('lat')),
          lng = parseFloat(point.getAttribute('lon')),
          coord = translate(projection([lng, lat]));

       geometry.vertices.push(new THREE.Vector3(coord[0], coord[1], (ele
/ 2470 * heightFactor) + (0.01 * heightFactor)));
    }
  }

  var material = new THREE.LineBasicMaterial({
    color: 0xffffff,
    linewidth: 2
  });

  var line = new THREE.Line(geometry, material);
  scene.add(line);
}

function translate(point) {
  return [point[0] - (terrainSize / 2), (terrainSize / 2) - point[1]];
}
```

This function is extracting the elevation, latitude and longitude values from the GPX track and creating a vertex for each point. The coordinate is projected using our D3 projection object and translated to the coordinate space of three.js, as explained in this blog post:

> *In Three.js the coordinate system works as follows. Point (0,0,0) is the center of the world. As you look at your screen, positive x values move to the right and negative x values move to the left. Positive y values move up and negative y values move down. Positive z values move toward you and negative z values move away from you.*

The elevation values are also multiplied by a height factor which is the same I've used for the terrain. In addition I'm adding a small offset, so the track is rendered slightly above the ground.

I'm using THREE.LineBasicMaterial to create the line style, and THREE.Line to put the line geometry and material together before adding it to the scene. You can see the white line on the map (click to see in WebGL):



The code is available on Github. An alternative would be skip the elevation values in the GPS track and instead clamp the track to the terrain, but I haven't found an easy way to do this with three.js.

6 comments:

puzz said...

This is really cool Bjørn.

It would be nice to add some (open source) license in the github repo with your repository. At the moment I'm not sure if I can use parts of your code on my site.

5 November 2013 at 05:18

---

i☆MAT said...

Great. This is what I wanted to make.

5 November 2013 at 20:31

Spencer Haley said...

While still struggling heartily with getting my shapefile to line up properly in conjunction with D3 (projection problems), I took a very cheap shortcut that had decent results; My gpx was converted to a shapefile and layered upon my terrain imagery via LineSymbolizer in mapnik (at the same time I was adding lakes and rivers). After testing on one of my own hikes' tracks, I can attest that it draped quite well over the three.js terrain area right where I was traversing along the ridges. Nowhere near as cool as what you showcased, but I thought I'd throw it out there!

10 November 2013 at 09:24

Anonymous said...

hi
great

how i can calculate distance between tow point???is there any way for it?

11 June 2014 at 21:33

Sebtest said...

Awesome expriment, love playing with the code.

If your code is open source, would you mind adding the open source license on the git hub repo ?

Wouldn't dream of reusing it without the license, got too much respect for fellow coders.

Thanks, and keep on the great job you've been doing all these years...

Sebastien

7 September 2015 at 20:29

Anonymous said...

Hi,
First thanks for the clarity of your post.
A question : what is the value of heightFactor? Thank you

9 December 2015 at 12:20

Post a Comment

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)