**Begriffdefinitionen:**

**Opentopodata**: Webseite, die eine API anbietet, um Höhen abzufragen. Diese API ist open-source und kann beliebig auf einem Server reproduziert werden. Höhen werden meistens durch .tif oder .hgt Dateien von Anbietern wie esri bereitgestellt. Diese sind als Rasterdaten gespeichert. Soll heissen, jeder Pixel besteht aus einem Grau-Wert, welcher dann durch diese API abgefragt werden kann. Auch diese Karten kommen in Tiles und werden je nach Koordinaten-Aufruf geladen oder auch nicht. Eine sehr bekannte API ist die Google Elevation API, doch diese kostet pro Aufruf und stimmt somit nicht mit unseren Kriterien überein.

**Tiles**: Ein sehr grosses Bild wird in kleine Teile aufgeteilt. Dies ist gedacht, damit der Client nicht das ganze File runterladen muss, sondern nur genau diesen Abschnitt welcher er braucht. Dies macht ein schnelles und dynamisches Ansehen der Webseite möglich. Diese Tiles werden normalerweise direkt vom Ersteller bereitgestellt, so dass der Client der eigenen Webseite die Tiles direkt von ihm anfordert. Dies spart Speicherplatz und Bandbreite auf dem eigenen/ auch GitHub Server.

**Leafletjs**: Open-source Service welcher ein Ansehen von Karten-Tiles ermöglicht. Zudem ermöglicht er ein Overlay, bei welchem Marker gesetzt, Linien gezeichnet werden und noch vieles mehr gemacht werden kann. Unsere Tiles sind von esri (<https://www.esri.com/de-de/arcgis/about-arcgis/overview>) bereitgestellt und sind die genausten und schnellsten Kartenfragmente, die für die Öffentlichkeit verfügbar gemacht wurden. Jedoch können bei solchen Tiles keine Höhen abgefragt werden. Nicht einmal Leafletjs unterstützt diese Funktionen. Deshalb wird ein Dienst wie Opentopodata benötigt, um diese Höhenkarten auslesen zu können.

**ArcGIS**: Dienst von esri, welcher eine App zum Bearbeiten von Höhenkarten und weiteren Features bereitstellt. Zudem können hochauflösende Höhenkarten heruntergeladen werden. Hierbei handelt es sich um eine Abfragerate von 25m. Die Genauigkeit ist offiziell (+/-) 7m. Europaweit ist das die genauste Karte, die frei erhältlich ist.

**GitHub Pages:** Dies ist ein Dienst von GitHub, welcher ein Repository zu einer Webseite macht, welche über *nutzernamen*.github.io erreichbar ist. Dieser Service ist gratis und ist speziell für Entwickler gedacht. Durch die Schnelligkeit von GitHub kann die Webseite innert Sekunden erneuert werden, obwohl man offline arbeitet. Genau aus diesem Grund habe ich GitHub Pages gewählt. Ich hätte noch eine andere Website, doch dies braucht jedes Mal Stunden, bis der neue Code online ist.

**European Petroleum Survey Group Geodesy (EPSG):** Dies war eine Arbeitsgruppe der europäischen Öl- und Gaserkundungsunternehmen. Sie definierte die EPSG-Kodierungen, welche genutzt werden, um Koordinatenreferenzsysteme zu unterscheiden. Die verschiedenen Höhenkarten sind also in verschiedenen Koordinatenreferenzsysteme verfasst. Glücklicherweise kann die opentopodata API mit verschiedenen Systemen rechnen.

CORS:

**Dokumentation:**

**[26.02.2020 - 8h]:** Ich hatte gestern die Idee einen «War Thunder Grind» Rechner mit GitHub Pages zu erstellen. War Thunder ist ein Computerspiel mit Flugzeugen, Panzer und Schiffen vom 2. WK bis hin in den Kalten Krieg. Doch die Zeit, um weiterzukommen, ist enorm. Deshalb wollte ich einen Rechner erstellen, welcher berechnet, wie lange man etwa noch braucht.

Als erstes musste ich mich Informieren, wie man Karten auf Webseiten anzeigt. Da man für Google Maps oder Google Earth zahlen muss, um sie nutzen zu können, kam dies für uns nicht in Frage. Ich stiess auf Leafletjs mit OpenStreetMap als Kartenanbieter. Durch das Tutorial von Leaflet entstand schnell eine Webseite mit einer Karte, auf der man Marker setzen und zoomen kann. Doch dies Mühsame Arbeit kam, alsbald ich begann, mich um die Höhe zu kümmern. Ich verstand nicht, dass dies nicht so einfach ist und dass in der Tat, wenige Kartendienste Höhendaten liefern. Ich erkannte, dass Leafletjs keine Höhendaten auslesen kann und dass ich einen anderen Weg finden musst. Durch recherchieren, stiess ich auf SRTM und andere Höhenkartenanbieter. SRTM (Definition einfügen). Die Files waren meistens in .tif oder .hgt. Also begann ich nach Lösungen zu suchen, wie ich diese Daten auf der Webseite auswerten kann. Es waren viele open-source Lösungen vorhanden, doch entweder wurden sie schon lange beendet, brauchten einen eigenen Server oder waren einfach nicht geeignet, da sie zum Beispiel nur ein Typ von EPSG unterstützten. Durch einen Stackoverflow Artikel stiess ich auf eine Webseite, opentopodata, die eine API zur Höhenberechnung durch AJAX Abfragen zur Verfügung stellt. Ich schien gerettet. Doch das nächste Problem lief mir schnell über den Weg. Durch CORS Implementation im Browser und Fehlkonfiguration des API-Servers war ich nicht in der Lage, die JSON Rückgabe der API zu lesen. Da ich keinen Zugriff auf den API-Server habe, konnte ich das CORS Problem nicht an der Wurzel packen, sondern musst eine Client Lösung finden. Es gibt zwei Wege. Entweder man deaktiviert CORS im Browser oder man nutzt einen Proxy Server, welcher dieses CORS Problem umgeht. Beide Wege haben ein grosses, jedoch verschiedenes Problem. Bei der ersten Handelt es sich um eine grosse Sicherheitslücke. Zu dem muss der Client ein Plugin herunterladen, um CORS zu deaktivieren. Dies kann man nicht von einem Client verlangen. Die Lösung

USG60W zyxel

[27.02.2020 – 6h]

Was das Programm bis jetzt kann Marker hinzufügen, die Höhe der Punkte in ein Popup schreiben und eine Linie zwischen den beiden Punkten zeichnen. Das Problem hier ist , dass weder die Marker noch die Polylines angesprochen werden können. Somit ist kein verschieben der Marker oder der Polylines gewährleistet.

Ich wollte meinem Dad die Webseite zeigen, doch leider funktionnierte plötzlich nichts. Das problem lag an der opentopodata API. Denn diese lässt nur 100 Anfragen von der gleichen IP Adresse zu. Das Problem liegt darin, dass ich einen Proxy brauche, um CORS zu umgehen. Das heisst, das alle Anfragne auf meiner Webseite durch diesen Proxy eine Anfrage an die API machen. So habe ich dieses Limit schon um 12 Uhr erreicht. Ich suchte die ganze Zeit andere Lösungen um CORS zu Umgehen, ohne proxies. Am Ende das Tages fand ich keine andere Lösung als mehrere Proxies zur Auswahl zu haben, so dass Javasscript automatisch zum nächsten wechselt, wenn einer nicht mehr die API anfragen kann. Ich habe eigene Marker Icons hinzugefügt. Ich habe sogenannte Polyline zu den Markern hinzugefügt. So sieht man besser, wo die Route durchführt. Doch es war alles statisch

[28.02.2020]

Ich musste Javascript neu schreiben, da es mir mit der jetzigen Zusammensetzung des codes nicht gelang, die Polyline und die Marker zu updaten. Jetzt gibt es einen Markerarray und einen position array. Somit werden die Marker mit dem Array referenziert, sprich man kann sie anstueren. Die PositionArray speichert alle x, y, z values von den Marker. So kann man sie später anpassen. Ich musste die drawPolyline function löschen, da ich sie ganz neu machen musste. Das Programm updated bei jedem mouse move die werte der Marker und fragt ihre neue höhe ab.