Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

IKG
Institute of Cartography
and Geoinformation

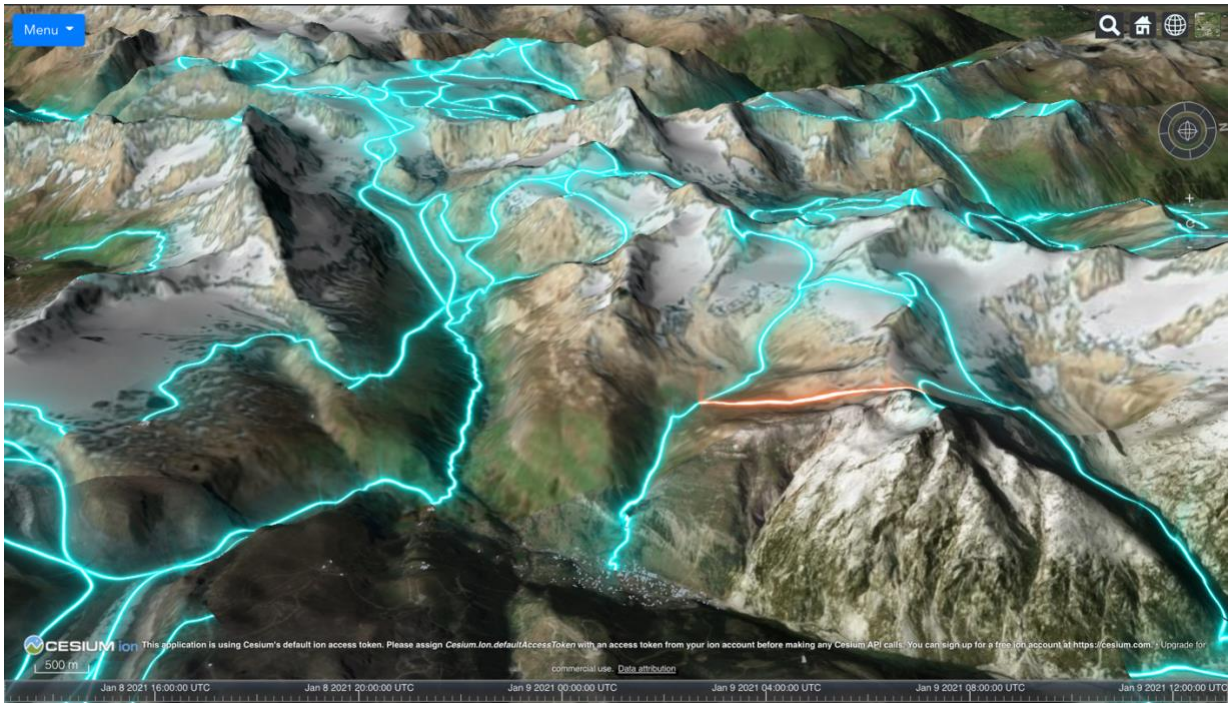
Cartography III (103-0227-00L)

Autumn semester 2020

Course lead: Prof. Dr. Lorenz Hurni

Skiing Zermatt

Project report



Code repository: https://gitlab.ethz.ch/carto3-hs20/Ke

Web application: http://carto3.ethz.ch/project/A5hTCf6z

Author: Bingxin Ke

Study program: M.Sc. Geomatics

bingke@student.ethz.ch

Zurich, 10.01.2021

Introduction

This project, *Skiing Zermatt*, is a 3D web map application aiming at helping young people discover ski trails that suit them in Zermatt. The biggest upgrade is the ski route interaction on a real 3D terrain, compared with other existing ski maps.

This map provides a fancy overview of ski routes. Glowing lines, free perspective of true 3D mountains, skiing animation and a profile map give users an overview of the ski route about the length, location and difficulty. At the same time, real-time weather information is provided.

Considering the needs of young people taking Instagram pictures, this map not only provides ski route information, but also gives photograph spots. They can see pictures uploaded by other users and where they were taken. (In this version, only a small number of pictures are used, and uploading is disabled.)

The UI design takes our target user, young people, into account. Firstly, it's totally compatibility on mobile end. Secondly, the interface is neat, fancy and easy to use.

Data

There are three data assets uploaded onto cesium ion, Cesium OSM Buildings, Cesium World Terrain and ski route GeoJSON data. Real-time weather data are required using API.

Ski routes are downloaded as shapefile. I use QGIS to check and manually select. I use feature selection and remove very short routes at boundaries. In order to import into cesium, I use QGIS to export polyline features as GeoJSON.

Design

These are my key design criteria of a fancy and functionality map: i)Eye-catching and fancy. ii)Simple and easy interactions. iii)Clarity of conveying information. iv)Compatibility on both PC end and mobile end.

Maps

I use *Cesium World Terrain* as my base map. It has mountain terrain details with texture. On the one hand, this gives a very straightforward overview of the skiing environment in Zermatt. On the other hand, 3D ski route polylines that clamp on the terrain would seem more realistic and provide more information than 2D lines.

I clamp the ski routes to ground and assign a fancy style to it. A skiing animation along the route would automatically play when the route is selected.

As for *Instagram Spots*, vertical image billboards are used with a distance-dependent scaling. So that pictures would seem like pins on the mountain, indicating the position.

User interface

I try to make UI clear and simple and try to get less clicks and neat interface.

I use these map UI: i) a navigation compass, to better control the view on PC end, especially for laptop touchpad control that have no middle button. ii) base map switcher, to customize the

base map texture. iii) feature query box, to display information of the selected features, in numbers.

I present an introduction modal window when page is loading, so that users won't get boring when waiting for terrain and data to load.

The most important UI element in this map is the *Menu*. This menu default to collapse and consistent of thematic layer switcher buttons, profile map button, camera control buttons and introduction buttons. Most interactions, except map control, are completed using this menu.

Implementation

I split my project into a main vue app (App.vue) and a CesiumViewer (CesiumViewer.vue) component. Templates are defined in main app. All cesium (map) functionalities are implemented in the cesium component.

I use *Cesium.createWorldTerrain()* to load terrain and use *Cesium.IonResource.fromAssetId()*, together with *Cesium.GeoJsonDataSource()* to load features. Loaded entities are added into viewer using *viewer.dataSources.add()* and *viewer.dataSources.add().*

Weather data are required with *XMLHttpRequest()* from API, then parsed as JSON.

In order to implement 'highlight selected route' style change and 'ski along the route' animation, *Cesium.ScreenSpaceEventHandler()* is used for binding click event with highlight and animation. This is the simplest way that I found to implement a click interaction.

As for UI elements. I use bootstrap plugin. I use a *modal* as introduction window and profile map window. And I choose to use a dropdown button group as menu. In this menu, I use vector icons (svg figure) from this bootstrap icon collection.

For clean code practice, I use human-readable variable names and try to split code into functions (whether reuse or not). I also use loop to reduce redundance. My comments can be found at most important locations. But I usually forget to eliminate redundancies with functions.

My advanced functionality is a calculated profile map. I extract the selected polyline points and then use *Cesium.sampleTerrainMostDetailed()* to get elevation. After that, I use *Echarts* to generate a line chart, and put it into a modal window.

Conclusion

Overall, I'm satisfied with this project, especially the idea and the design. From technical aspect, it's my first front-end project, also first time to use JavaScript, Vue.js, and cesium. Challenges are everywhere for a beginner like me. Here I want to thank all the lecturers for helping.

There are some limitations of my project. Firstly, the *Instagram spot* data is simulated now. It would be better if there is an uploading gateway for people to upload their pictures, we can extract location information from EXIF. Secondly, the 3D terrain render is now computational expensive, which would reduce the battery life in outdoor usage.

One lesson I learned during the project is 'planning before hands-on'. So that one would not waste time doing useless or repeat work.

Appendix

Data

Description	URL
Ski route around Zermatt	https://geovite.ethz.ch/
3D model of OLAF	https://sketchfab.com/3d-models/olaf-the-snowman-frozen-519d3a73886041fdb4fdb6ecc298861e
Cesium OSM Buildings	https://cesium.com/content/cesium-osm-buildings/
Cesium World Terrain	https://cesium.com/content/cesium-world-terrain/
Zermatt pictures	https://www.instagram.com/
Real-time weather information	https://openweathermap.org/api

Table 1: Data sources used in this project

Maps

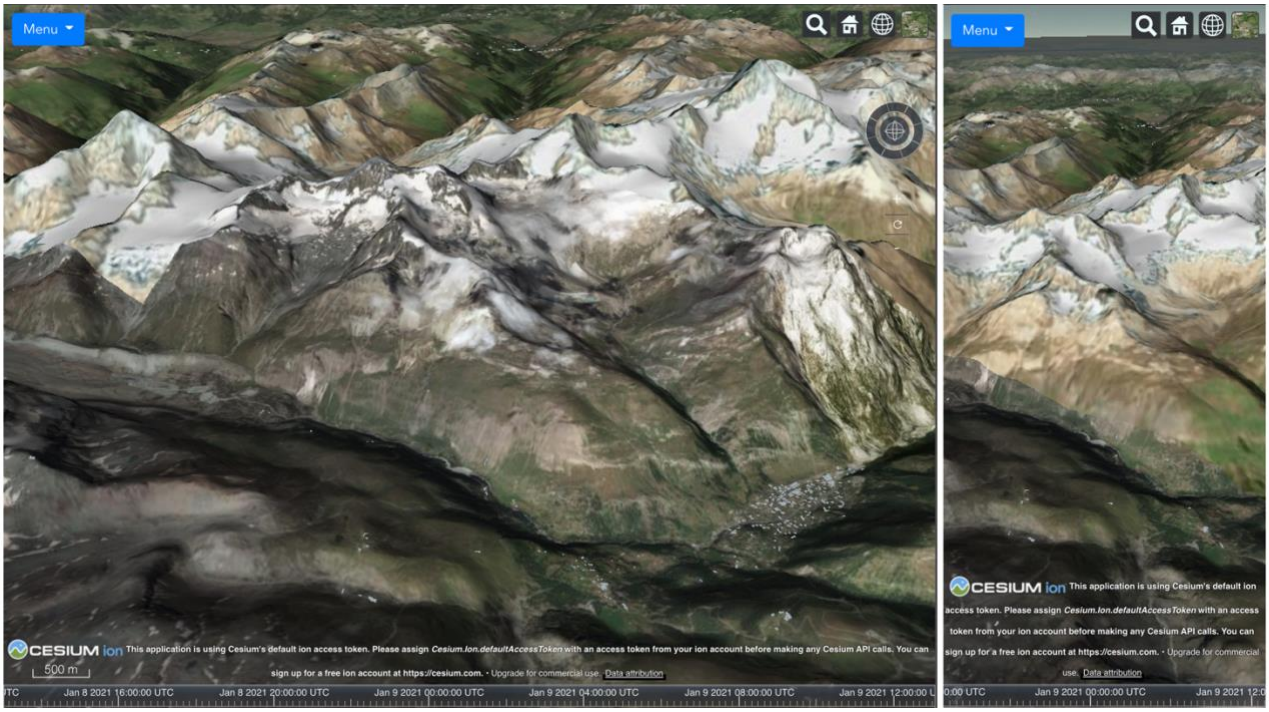


Figure 1: Base terrain layer (L: pc end, R: mobile end)

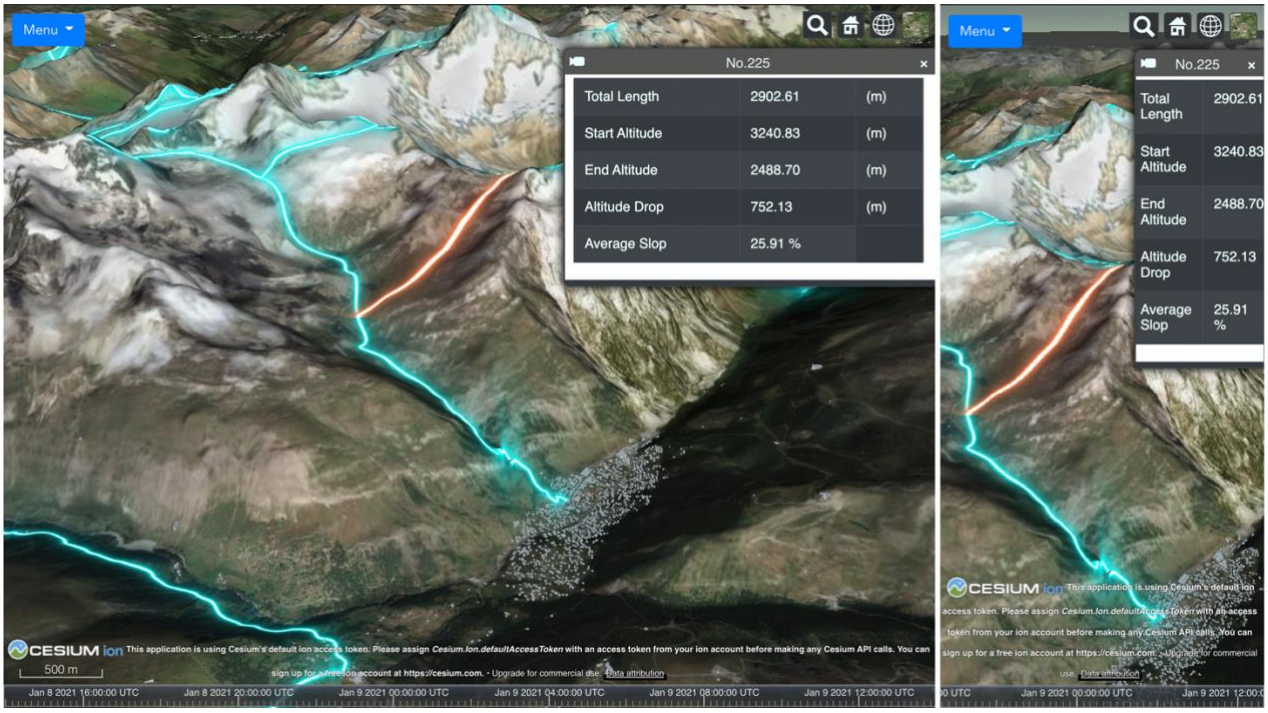


Figure 2: Ski route layer (L: pc end, R: mobile end)

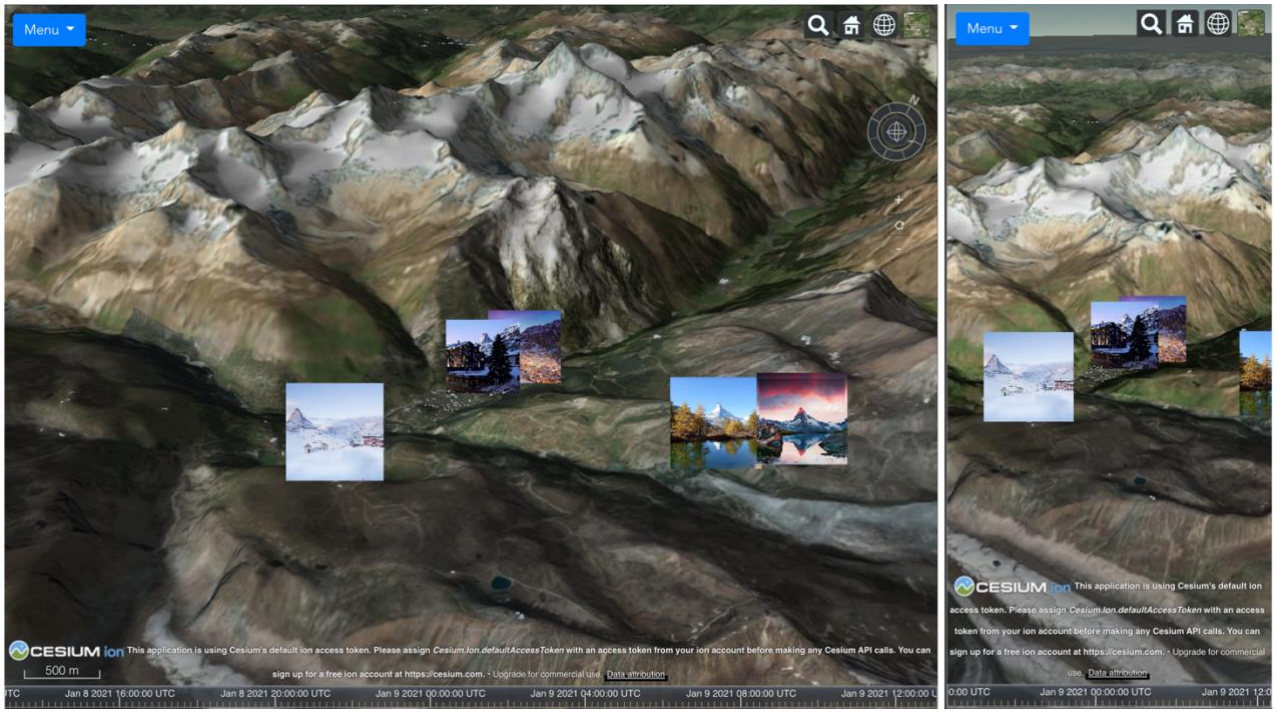


Figure 3: Instagram spot layer (L: pc end, R: mobile end)

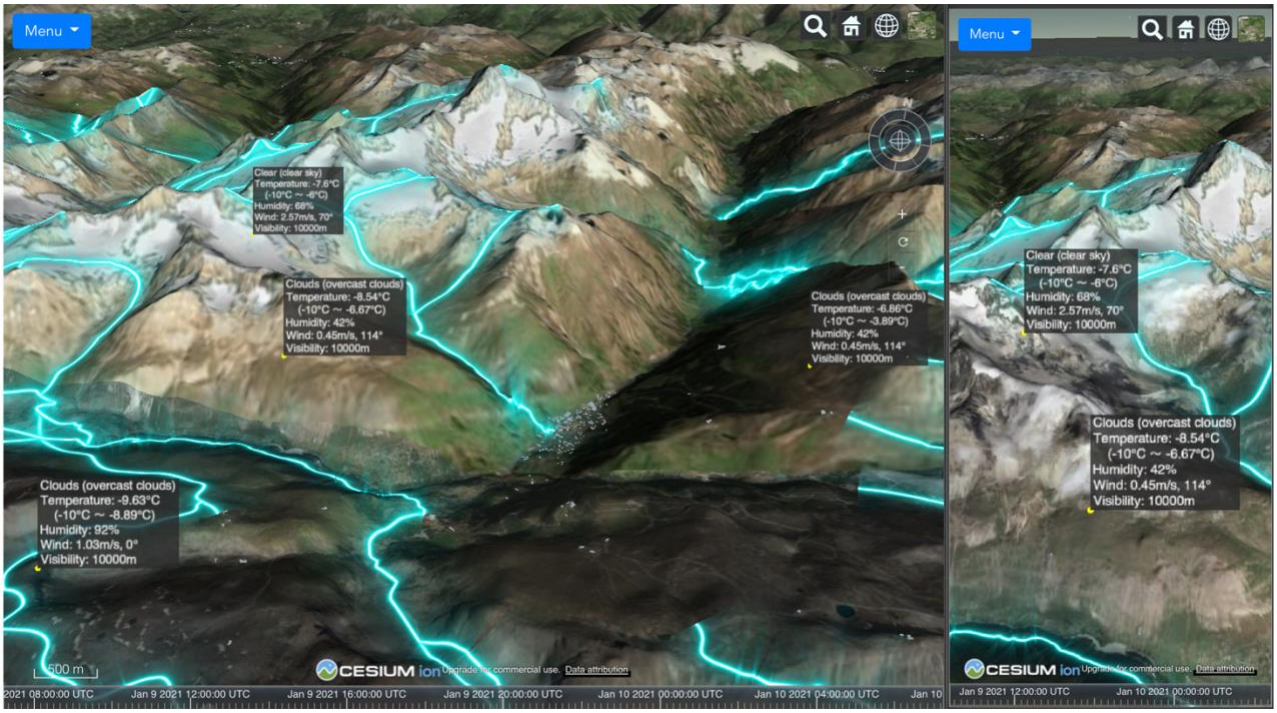


Figure 3-2: Real-time weather layer (L: pc end, R: mobile end)

User interface

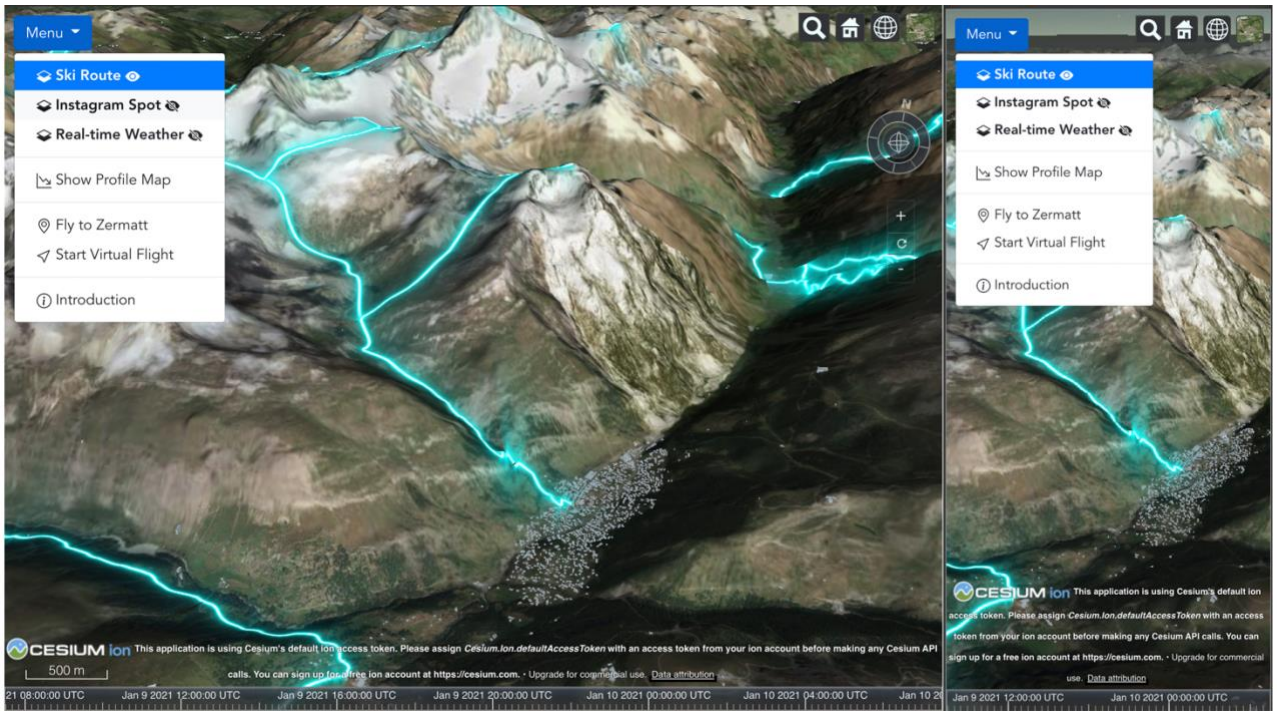


Figure 4: Overall layout with expanded Menu (L: pc end, R: mobile end)

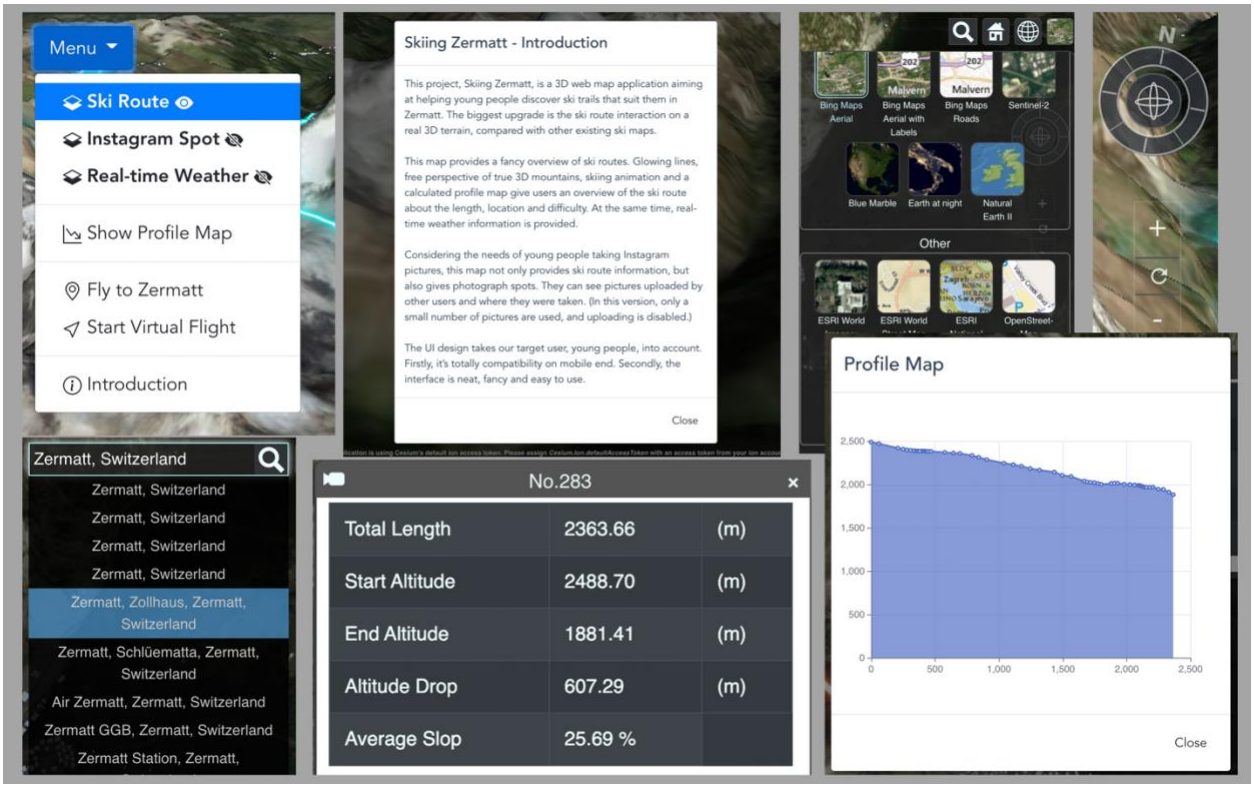


Figure 5: Collage of UI elements

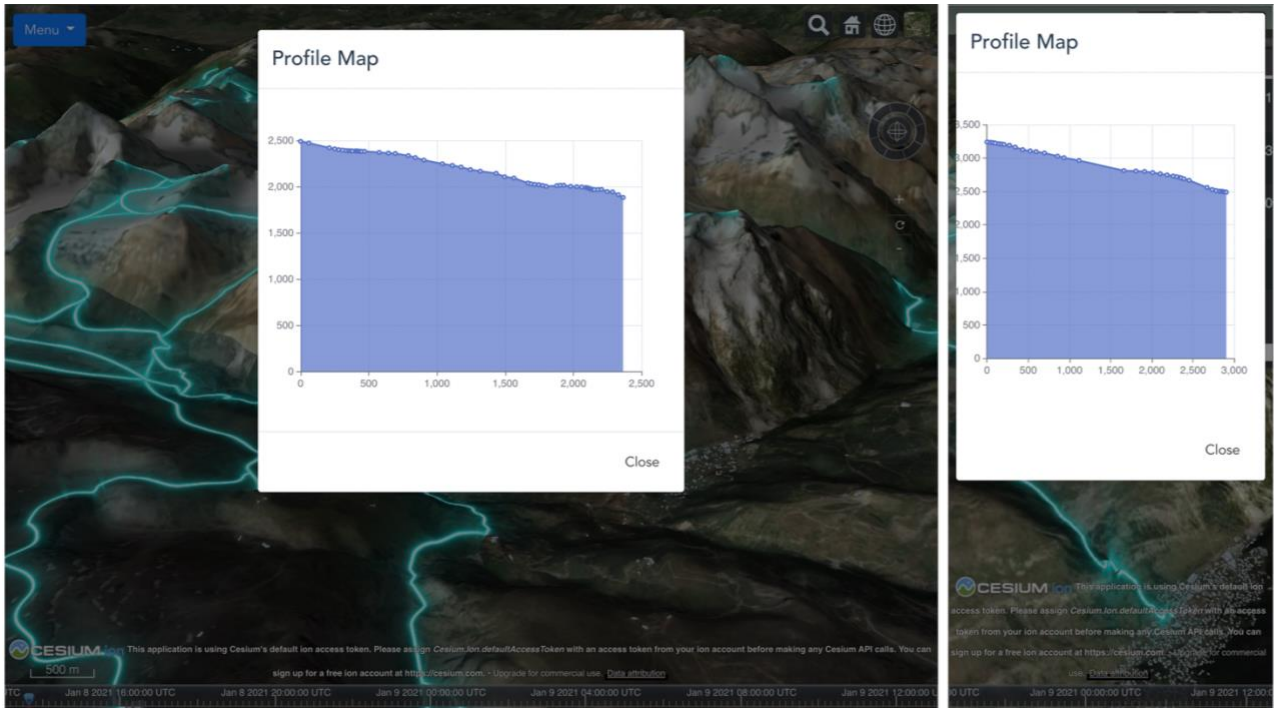


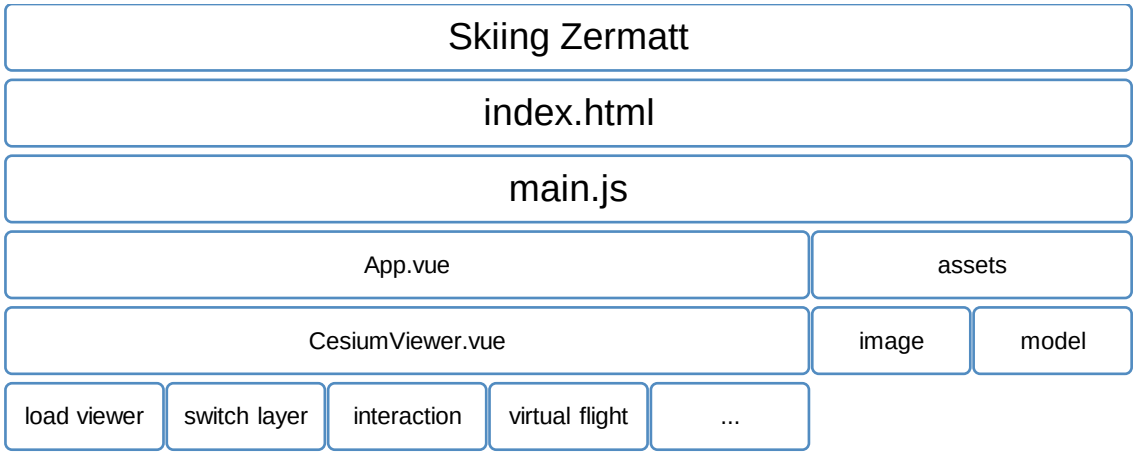
Figure 6: Profile map (L: pc end, R: mobile end)

Software

Name	Description	URL	License
CesiumJS	Virtual globe, 3D maps	https://github.com/CesiumGS/cesium	Apache 2.0
Vue.js	User interface	https://github.com/vuejs/vue	MIT
cesium-navigation-es6	Cesium navigation component plug-in	https://github.com/richard1015/cesium-navigation-es6	MIT
Echarts	Create interactive charts	https://echarts.apache.org/en/index.html	Apache-2.0
Bootstrap	Html UI components	https://getbootstrap.com/	MIT

Table 2: Web frameworks and libraries used in this project

Application architecture



Data processing workflow