Interpolacijske metode i određivanje optimalnog puta u prostoru

Palinić, Domagoj

Sveučilište u Rijeci – Fakultet informatike i digitalnih tehnologija

Preddiplomski Sveučilišni studij Informatika

Domagoj Palinić

Interpolacijske metode i određivanje optimalnog puta u prostoru

Završni rad

Mentor: dr. sc. Sanda Bujačić Babić

Rijeka, 2022.

Sažetak

U ovom završnom radu bavimo se određivanjem optimalnog puta u prostoru putem web aplikacije koja koristi Google Mapu. U uvodnom dijelu navest ćemo kratku povijest kartografije, GPS sustava i Google Maps platforme te definirati ideju koja stoji iza ovog rada. Definirat ćemo pojmove vektora, vektorskog prostora i Euklidskog prostora te objasniti problematiku rada s vektorskim prostorom koji nije Euklidski prostor. Definirat ćemo pojmove koordinatizacije pravca, ravnine i prostora te u objasniti rad sa sfernim koordinatnim sustavom uz korištenje Haversine formule za određivanje udaljenosti između točaka na sferi. U programskom dijelu objasnit ćemo izvedbu algoritma koji za određeni put definiran u prostoru pomoću Google Mape računa alternativni put kojim se nastoje zaobići prevelike razlike u nadmorskoj visini čineći tako put prohodnim za, primjerice, planinarenje.

Ključne riječi

Kartografija, mapa, karta, atlas, Google Maps platforma, Google Mapa, GPS, satelit, geografska širina, geografska dužina, koordinatni sustav, koordinatizacija, Haversine, alternativni put, optimalni put, dužina, usmjerena dužina, vektor, skalar, vektorski prostor, Euklidski prostor, predstavnik vektora, norma vektora, koordinatizacija pravca, baza prostora, koordinatizacija ravnine, koordinatizacija prostora, zbrajanje vektora, sferni koordinatni sustav, Google Maps API, HTML, JavaScript, CSS, web aplikacija, marker, putanja, algoritam

Sadržaj

1. Uvod

1.1. Povijesni pregled kartografije

Kartografija se definira kao znanost i umjetnost izrade geografskih karata, odnosno grafičkih reprezentacija prostora u različitim mjerilima, nudeći nam tako informacije o prostoru koje možemo dalje koristiti za bolje razumijevanje topografije, meteorologije te kulturnih i ekonomskih utjecaja na društvo. U ovom slučaju, kartografiju koristimo prvenstveno za pronalaženje novih puteva, što je na neki način i njena osnovna svrha.

Neke od najranijih poznatih karata iz 16. st. pr. n. e. prikazuju noćno nebo, odnosno razmještaj zvijezda, Mjeseca i drugih objekata na nebu. Prve karte nađene u ruševinama Babilona, većinom crtane na glinenim pločicama, prikazuju topografske informacije poput crteža planina i dolina. Arheolozi vjeruju da su crtane koristeći veoma točne podatke.[1]

Prve karte s Europskog kontinenta bile su većinom simbolične te su korištene kao ilustracije –one nisu davale točne podatke korisne za navigaciju. U 13. stoljeću Malorkanska kartografska škola sastavljena od židovskih i kršćanskih kartografa postavlja temelje kartografije koja se može koristiti u svrhe navigacije, sastavljajući karte koje sadrže mrežu linija za navigaciju. U doba velikih geografskih otkrića mreža linija se nadograđuje, koristeći mrežu meridijana i paralela zasnovanih na točnijim podacima koji su Zemlju predstavljali kao sferu.[2]

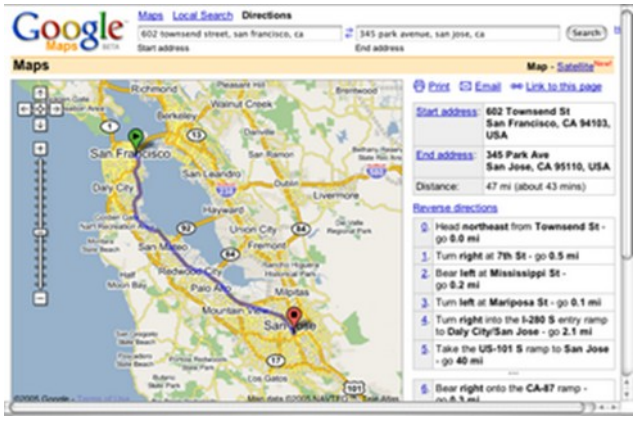
Slika 1: Katalonski atlas, Malorkanska kartografska škola

Tijekom 20. stoljeća, korištenje zrakoplova za fotografiranje dovodi do promjene podataka koji se koriste kao izvori za izradu karata. Nadalje, satelitska tehnologija razvijena krajem stoljeća nudi novu razinu točnosti, prikazujući vrlo velike prostore sa mnoštvom detalja. [3]

1.2. Povijest Google Maps-a

Projekt Google Mape započeo je kao ideja braće Larsa i Jensa Rasmussena, Noela Gordona i Stephena Maa kao program pisan u C++, u okviru australske tvrtke *Where 2 Technologies*. Prvotna ideja bila je da se program koristi offline, ali nakon što je kompanija odlučila započeti suradnju s Googleom, program je prilagođen korištenju u obliku web aplikacije.[4] Nakon kolovoza 2004. Google je preuzeo vodstvo kompanije te je zajedno sa softverom za vizualizaciju geografskog prostora tvrtke Keyhole započeo izgradnju Google Maps-a.[5] U veljači 2005. Google je objavio prvu verziju Google Mapsa na Google Blogu. Prvi test za novu aplikaciju bio je za vrijeme uragana Katrina, kada je Google Maps davao točne podatke i satelitske snimke građanima New Orleansa za praćenje poplave.[6]

Sredinom 2007. Google Maps opremljen je prozorčićima koji daju informacije o određenoj lokaciji („info windows“)[7] te beta verzijom programa za određivanje lokacije korisnika uz pomoć triangulacije signala u odnosu na obližnje odašiljače mobilnog signala i prijavljene Wi-Fi routere.[8]



Slika 2: Beta verzija Google Maps-a

Od 2016., kada je Google dobio mogućnost korištenja satelita Landsat 8, Google Maps koristi satelitske slike u visokoj rezolociji kao izvor podataka za svoje karte. Sam satelit dodao je Google-u preko 700 trilijuna piksela novih podataka.[9] Krajem 2018. Google zamjenjuje Merkatorsku projekciju koja prikazuje svijet kao sferu projiciranu na pravokutnik trodimenzionalim modelom globusa. Ova promjena vidljiva je kada korisnik namjesti zoom postavku na najmanji

broj, odnosno kada se do kraja udalji u smislu pogleda na mapu.[10]

Google Maps API, trenutno nazivan Google Maps Platform, koristi 17 različitih API-ja koji se dijele u tri osnovne kategorije: mape, mjesta i rute. [11] Za korištenje od strane developera na vlastitim stranicama i projektima, Google Maps je bio besplatan do 2018. godine te se nakon 2018. besplatna verzija pojavljuje uz vodeni žig (eng. watermark), a potpuna verzija zahtijeva pretplatu.

Iako je u početku razvijan isključivo kao JavaScript API, kasnije je proširen da bi uključio potporu za Adobe Flash aplikacije te je prilagođen kako bi nudio mnoge usluge poput dohvaćanja statičkih slika karata, izračuna s geografskim koordinatama, generiranja ruta za vožnju te dohvaćanja podataka o nadmorskoj visini pojedinih točaka.

2. Osnovni matematički pojmovi. Koordinatni sustav. Koordinatizacija

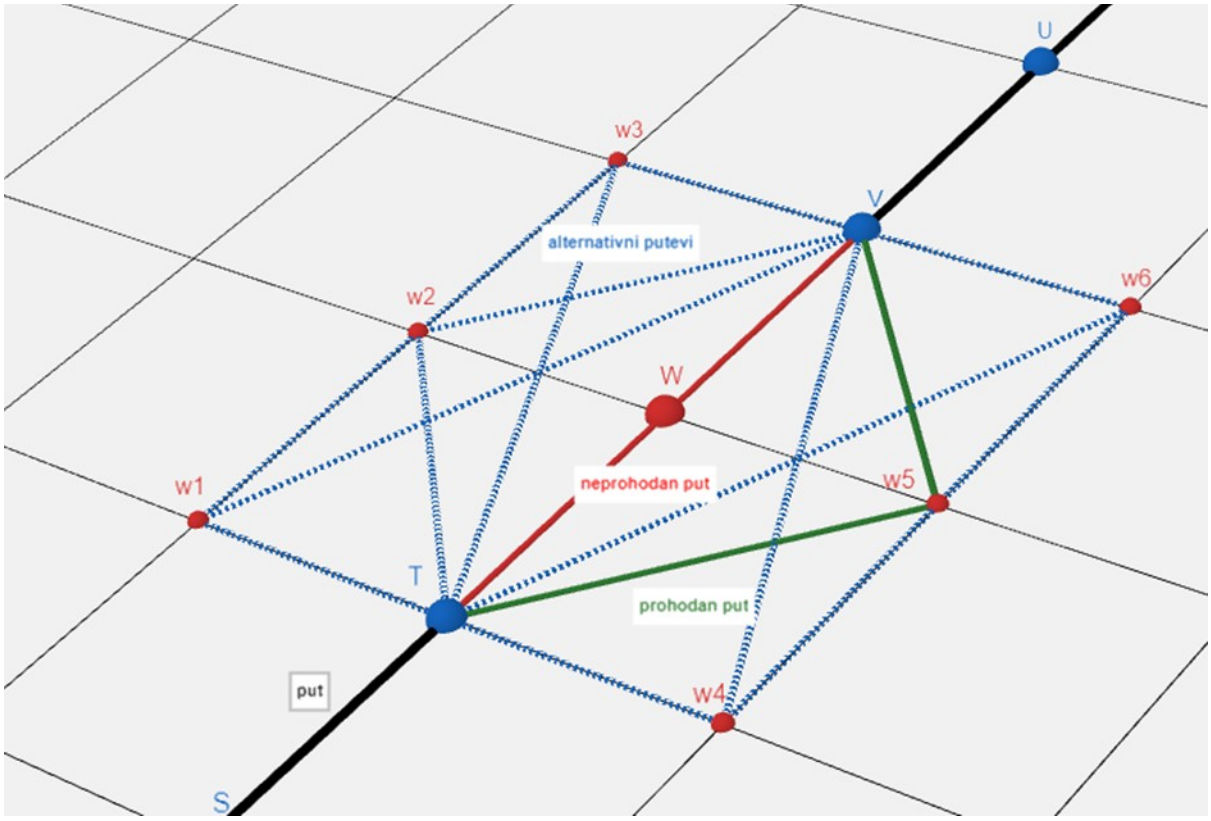
2.1. Osnovna ideja koordinatnog sustava

Geografska dužina i širina su osnovne komponente koje se koriste prilikom kreiranja navigacijskih sustava. Tijekom tog složenog procesa javlja se potreba za apstrahiranjem i matematičkim modeliranjem prostora odgovarajuće dimenzije za što se koristi koordinatni sustav. Koordinatni sustav realizirat ćemo mrežom čvorova koji mogu, ali i ne moraju biti jednako udaljeni jedan od drugoga. U ovom slučaju ta udaljenost mora biti poznata ili zadana nekim unaprijed poznatim parametrom.

U našem slučaju se javlja problem određivanja udaljenosti između dvije točke na sferi, koja se smatra neeuklidskim prostorom. Budući da koristimo Google Maps, koji se od kraja 2018. godine prikazuje na sferi, biti će nam potrebno koristiti drugačije formule od onih koje bismo koristili da je riječ o mapi u ravnini, odnosno u Euklidskom prostoru. Kako bi utvrdili odnose i vrijednosti, koristimo formule koje bismo primjenili u sfernoj geometriji. Stoga zamišljamo koordinatni sustav u kojem za x-os uzimamo geografsku širinu, a za y-os geografsku dužinu, odnosno – radimo sa sfernim koordinatnim sustavom. Za određivanje udaljenosti između točaka tog sustava potrebna nam je formula definirana za sfernu geometriju. U našem slučaju to je Haversine formula koja računa kvadrat sinusa polovičnog kuta (tzv. Haversine) na površini koja je projicirana na sferu, a za to koristi vektore i točke u sfernom koordinatnom sustavu. Ovu ćemo formulu izraziti drugačije, tako da za poznate pozicije točaka u obliku koordinata i poznat Haversine dobijemo

udaljenosti između točaka na sferi te ćemo time omogućiti određivanje udaljenosti između točaka na mapi, kao i crtanje novih točaka razmaknutih na poznatim odnosno unaprijed zadanim udaljenostima.

Direktno iz problema određivanja udaljenosti između dvije točke javlja se i problem određivanja udaljenosti između više točaka. Pretpostavimo da određeni put od vrha planine do njenog podnožja možemo aproksimirati ravnom linijom ukoliko je taj put relativno ravan. Ako put ima oštre zavoje i često mijenja smjer, bit će nam potrebno više točaka te će ukupni put biti jednak sumi puteva između tih točaka. Zbog toga je nužno uvesti koordinatni sustav, pomoću kojeg ćemo promatrati veći broj točaka na mjestima na kojima spomenuti put nije ravan ili je prestrm. Ove točke će biti određene na unaprijed zadanoj udaljenosti od središnje točke koju želimo zamijeniti kao dio našeg puta te ćemo zatim odabrati najkraći put između prethodne i sljedeće točke koji je uz to prohodan ili manje strm od onog početnog. Naravno, budući ne radimo s ravnom plohom već sa sferom, ovaj koncept treba primjeniti unutar sferne geometrije, odnosno na sfernoj ravnini.



Slika 3: Pronalaženje alternativnog puta u ravnini, pri čemu put između točaka T, W, V zamjenjujemo putem između točaka T, w5, V. Točke S, T, W, V, U su zadane točke, dok su točke w1, ..., w6 točke mogućeg alternativnog puta

2.2. Koordinatizacija

2.2.1. Vektori

Definicija 1 (dužina):

Dužina $\overline{AB}$ je skup svih točaka pravca na kojima leže točke *A* i *B* koje se nalaze između točaka *A* i *B*, uključujući točke *A* i *B*.

Definicija 2 (usmjerena dužina)

Usmjerena dužina je dužina $\overline{AB}$kojoj je poznato da je točka *A* njena početna točka, a točka *B* njena krajnja točka. Takvu usmjerenu dužinu označavamo s $\overrightarrow{AB}$.

Definicija 3 (vektor):

Vektor je klasa ekvivalencije usmjerenih dužina. [12]

Napomena 1 (skalar):

Skalar je veličina koja izražava jednim parametrom i prikazuje na jednoj koordinatnoj osi. Njegova je vrijednost realan broj.

Definicija 4 (vektorski prostor):

Vektorski prostor nad nekim poljem *K* je neprazan skup $V\,(V \neq \emptyset)$ za kojeg vrijede svojstva:

1. komutativnost s obzirom na zbrajanje, $v+w=v+w, \forall \quad v, w \in V$,
2. distributivnost s obzirom na zbrajanje, $\lambda(v+w)=\lambda v+\lambda w$,
3. kvaziasocijativnost, $(\lambda\mu)v=\lambda(\mu v)$,
4. svaka jedinica ima svojstvo: $1v=v, \forall \quad v \in V$ [13]

U ovom slučaju se uređena trojka (V, +, ·) naziva *vektorski prostor* ili *linearni prostor nad poljem* K. Ako je K = R, onda govorimo o *realnom vektorskom prostoru*, a ako je K = C onda govorimo o *kompleksnom vektorskom prostoru*. [14]

Definicija 5 (Euklidski prostor):

Euklidski prostor E je realan vektorski prostor na kojem je definirano preslikavanje $\varphi: V \times V \to R$ sa svojstvima:

1. $v \cdot w = w \cdot v$
2. $(u+v) \cdot w = u \cdot w + v \cdot w$
3. $\alpha(v \cdot w) = (\alpha v) \cdot w = (\alpha w) \cdot v$
4. $w \cdot w > 0$, *ako i samo ako vrijedi* $w \neq 0$ [14]

Vezu između usmjerenih dužina i vektora dobivamo iz osnovnog svojstva Euklidskog prostora.

Napomena 2 (predstavnik vektora):

Ako je $A \in E$ proizvoljna točka te $\vec{a}$ zadani vektor, tada postoji jedinstvena točka B takva da A i B čine usmjerenu dužinu $\overrightarrow{AB}$, koja je predstavnik vektora $\vec{a}$. Pišemo:

$$\vec{a} = \overrightarrow{AB}, \vec{b} = \overrightarrow{BC}, \vec{c} = \overrightarrow{CD}, \ldots$$

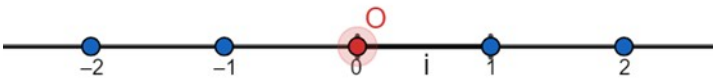
Iako ovaj zapis nije u potpunosti precizan, jer je vektor klasa ekvivalencije usmjerenih dužina, a usmjerena dužina je samo jedan njegov predstavnik, dogovorno u praksi ne radimo razliku između vektora i njegovog predstavnika. [15]

2.2.2. Koordinatizacija pravca

Dva ili više vektora su međusobno linearno nezavisni ako se ni nedan od njih ne može izraziti kao linearna kombinacija konačnog broja vektora u skupu u kojem su određeni. Linearnu nezavisnost definiramo preko linearne zavisnosti. Ako postoji konačan skup vektora $v_1, v_2, \ldots, v_n$, te ako postoji konačan skup skalara $a_1, a_2, \ldots, a_n$, $a_i \neq 0, i = 1, 2, \ldots, n$, za koje vrijedi:

$$a_1 v_1 + a_2 v_2 + \ldots + a_n v_n = \vec{0},$$

pri čemu je $\vec{0}$ oznaka za nul-vektor, tada kažemo da su vektori $v_1, v_2, \ldots, v_n$ linearno zavisni. Ako ne postoji takav skup skalara, odnosno kada gornji izraz vrijedi samo za slučaj kada su $a_i = 0, i = 1, 2, \ldots, n$, onda kažemo da su vektori $(v_1, v_2, \ldots, v_n)$ međusobno linearno nezavisni. [16]



Slika 4: Koordinatizacija pravca

Definicija 7 (baza prostora):

Neka je $\{a_1, a_2, \ldots, a_n\}$ skup linearno nezavisnih vektora. Taj skup je baza *n*-dimenzionalnog prostora *E* ukoliko je tim skupom određen *n*-dimenzionalni koordinatni sustav ($O;$ $a_1, a_2, \ldots, a_n$).

Svaki vektor *b* iz prostora *E* može se jednoznačno prikazati kao linearna kombinacija vektora baze, odnosno:

$$b = \alpha_1 \cdot a_1 + \alpha_2 \cdot a_2 + \ldots + \alpha_n \cdot a_n.$$

Napomena 3:

U ovom radu bavit ćemo se slučajevima koji postoje u dvodimenzionalnom, kao i u trodimenzionalnom prostoru.

Definirajmo koordinatizaciju pravca. Odabiremo pravac *p* kroz točku $O \in E$. Definirajmo jedinični vektor $\vec{i}$ tako da je on predstavnik usmjerene dužine $\overrightarrow{OI}$, pri čemu je točka *I* na mjestu na kojem je na brojevnom pravcu broj 1. Time je vektor $\vec{i}$ jednoznačno određen te pišemo:

$$\vec{i} = \overrightarrow{OI}, |\overrightarrow{OI}| = 1,$$

pri čemu je $|\overrightarrow{OI}|$ oznaka za duljinu usmjerene dužine $\overrightarrow{AI}$.

Na ovaj način smo na pravcu *p* zadali koordinatni sustav (O; *i*).

Svakoj točki *T* koja leži na pravcu *p* jednoznačno pridružujemo *skalar* α i vektor $\overrightarrow{OT}$. Vektor $\overrightarrow{OT}$ možemo izraziti kao umnožak skalara α i jediničnog vektora $\vec{i}$, pri čemu α nazivamo *skalarnom komponentom* vektora $\overrightarrow{OT}$:

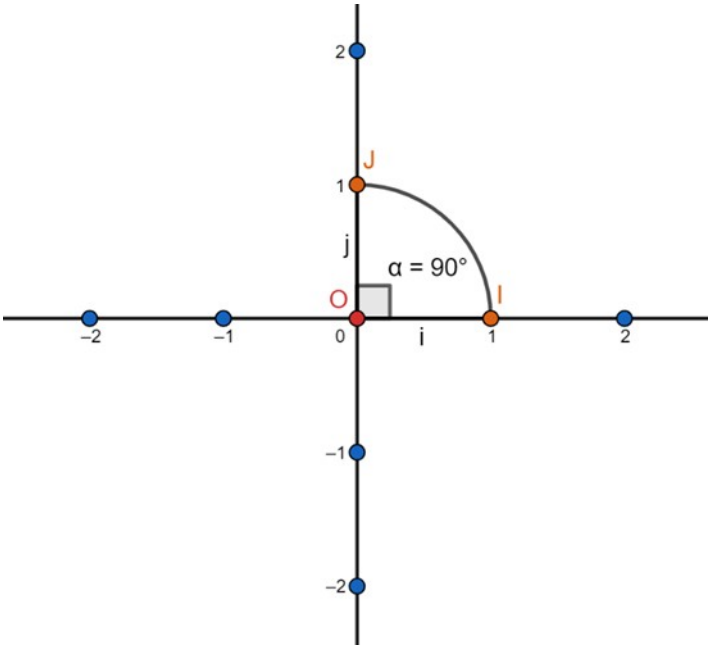
$$\overrightarrow{OT} = \alpha \cdot \overrightarrow{OI} = \alpha \cdot i$$ [17]

2.2.3. Koordinatizacija ravnine

U ravnini p koja se nalazi u prostoru E odabiremo točku O kao ishodište. Odabiremo zatim međusobno okomite pravce a i b koji leže u ravnini p i prolaze kroz točku O. Definiramo koordinatne sustave $(O; \vec{i})$ i $(O; \vec{j})$ na pravcima a i b:

$$i = \overrightarrow{AI}, j = \overrightarrow{AJ}, |i| = |j| = 1.$$

Točke I i J odabiremo tako da točka I rotacijom oko ishodišta za kut od 90° ili $\pi/2$ u pozitivnom smjeru prelazi u točku J. Time smo u ravnini p zadali *desni ili ortogonalni koordinatni sustav* $(O; \vec{i}, \vec{j})$. [18]



Slika 5: Koordinatizacija ravnine

2.2.4. Koordinatizacija prostora

Kod koordinatizacije trodimenzionalnog prostora slučaj je sličan kao kod koordinatizacije pravca i ravnine. Najprije odaberemo točku *O* kao ishodište. Zatim odredimo međusobno okomite pravce *p*, *q* i *r* koji prolaze kroz točku *O*. U ravnini koju određuju pravci *p* i *q* odredimo desni pravokutni koordinatni sustav $(O; \vec{i}, \vec{j})$ analogno tome kako smo odredili koordinatni sustav ranije. Zatim na pravcu *r* odredimo koordinatni sustav $(O; \vec{k})$, tako da su vektori $\vec{i}, \vec{j}, \vec{k}$ međusobno okomiti jedan na drugog. Ovime smo definirali desni pravokutni koordinatni sustav $(O; \vec{i}, \vec{j}, \vec{k})$ u prostoru *E* Vrijedi:

$$\vec{i} = \overrightarrow{AI}, \vec{j} = \overrightarrow{AJ}, \vec{k} = \overrightarrow{AK}, |i| = |j| = |k| = 1.$$

Brojevne osi koje smo na ovaj način kreirali nazivamo *apscisa*, *ordinata* i *aplikata*. Ove tri osi definiraju tri koordinatne ravnine (*xy*, *yz*, *zx*) te one dijele prostor na osam *oktanata*. Svaka proizvoljna točka *T* ima koordinate u koordinatnim sustavima $(O; \vec{i})$, $(O; \vec{j})$, $(O; \vec{k})$. Ove koordinate u praksi označavamo sa *x*, *y* i *z*. One predstavljaju skalarne komponente vektora $a = \overrightarrow{AT}$ u sustavu $(O; \vec{i}, \vec{j}, \vec{k})$.

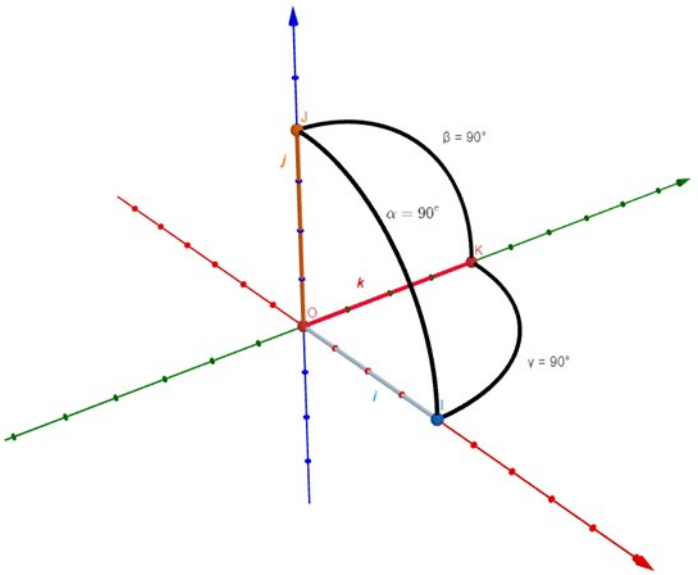
Prema pravilu za zbrajanje vektora vrijedi:

$$\vec{a} = \overrightarrow{OT} = x \cdot \overrightarrow{OI} + y \cdot \overrightarrow{OJ} + z \cdot \overrightarrow{OK},$$

$$\vec{a} = x \cdot i + y \cdot j + z \cdot k.$$

Napomena 4:

Kod mnogih slučajeva u dvije ili više dimenzija primjereno je uzimati predstavnike vektora koji svi imaju istu početnu točku. Odaberimo točku *O* u prostoru *E*. Svakoj točki *G* pripada jednoznačno određen vektor $\overrightarrow{OG}$. U tom slučaju, vektor $\overrightarrow{OG}$ je radijus-vektor ili vektor položaja točke *G* u odnosu na točku *O*.



Slika 6: Koordinatizacija prostora

2.3. Norma vektora

Normu vektora definiramo kao duljinu vektora. Neka su zadane dvije točke, T_1 i T_2, za određivanje udaljenosti između njih možemo povući pravce usporedne s osima *x* i *y* te na taj način dobiti pravokutni trokut. U ovom slučaju gledamo pravokutni trokut kojemu su vrhovi točke T_1, T_2 i sjecište povučenih pravaca. Stranice ovog trokuta su označene sa *a*, *b* i *c*, pri čemu je *c* hipotenuza čiju duljinu želimo odrediti. S obzirom da je $a=\left(y_{T_1}-y_{T_2}\right)$, $b=\left(x_{T_2}-x_{T_1}\right)$, primjenom Pitagorinog poučka lako dobivamo:

$$c=\sqrt{\left(y_{T_1}-y_{T_2}\right)^2+\left(x_{T_2}-x_{T_1}\right)^2}.$$

2.4. Sferni koordinatni sustav

U našem slučaju koristimo sferni koordinatni sustav za računanje koordinata na mapi koja se projicira na sferu. Potrebne su nam koordinate točaka koje će korisnik proizvoljno unijeti na samoj mapi, odnosno startna i krajnja pozicija puta. Nadalje, tražimo koordinate točaka koje naš program automatski dodaje na mapu kako bi prikazao alternativni put.

Sferni koordinatni sustav određen je ishodištem *O* i međusobno okomitim pravcima *p* i z. Nekoj proizvoljnoj točki *P* pridružujemo koordinate *(r, φ, θ)*, gdje je *r* koordinata jednaka udaljenosti od ishodišta do točke *P*, koordinata *φ* je kut koji vektor $\overrightarrow{OP}$ zatvara s poluosi *z*. Sve su

točke prostora jednoznačno određene ukoliko su vrijednosti sfernih koordinata ograničene,

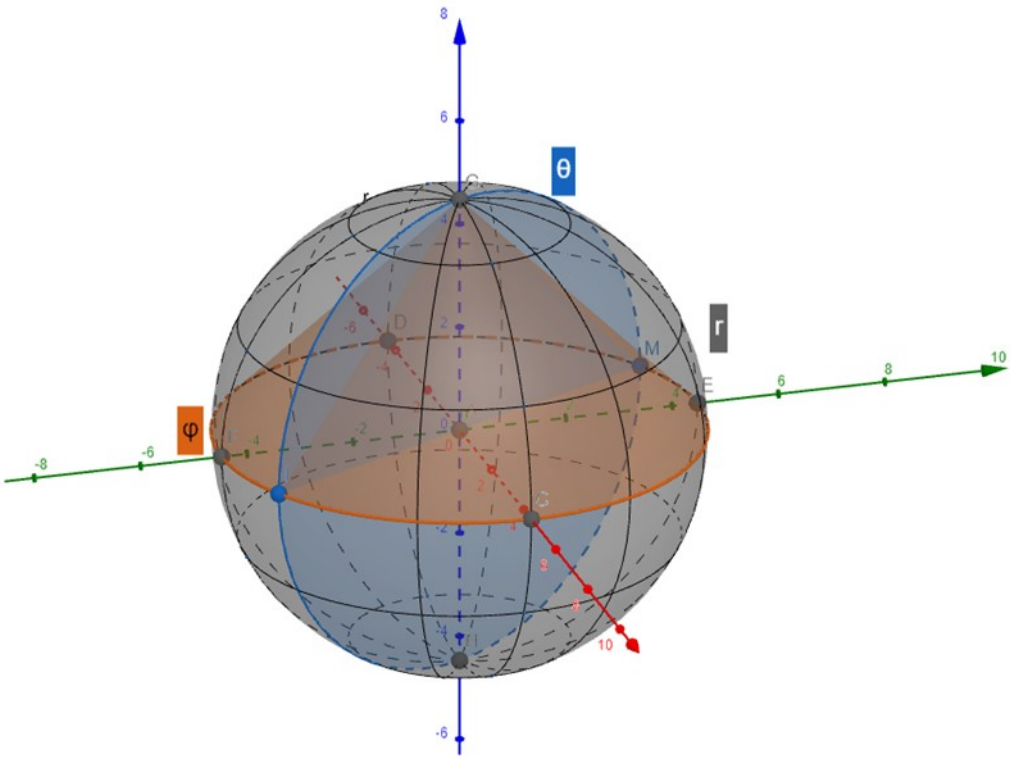
$$0<p<\infty,$$
$$-\pi<\varphi<\pi,$$
$$0<\theta<\pi.$$

Preciznije, u našem slučaju radimo s geografskom dužinom i geografskom širinom, odnosno *kordinatnim plohama* u sfernom koordinatnom sustavu.

Zamislimo sferu sa plohama r, φ, θ:

- ploha r je sfera sa središtem u ishodištu,
- plohe φ je poluravnine s rubom u z-osi,
- plohe θ je plaštevi jednoplošnih stožaca s vrhom u ishodištu i središnjom osi z.

Primjetimo da, fiksiramo li sferu r i presječemo je preostalim koordinatnim plohama,



Slika 7: Sferni koordinatni sustav

2.5. Haversine formula

Haversine formulu možemo definirati kao formulu za računanje kvadrata sinusa polovičnog kuta na sferi, pri čemu kut možemo definirati kao kvocijent dužine na sferi i njenog radijusa:

$$\theta = \frac{c}{r},$$

pri čemu je *c* sferna distanca, odnosno duljina dužine na površini sfere, a *r* radijus sfere.

Haversine formula nam omogućuje da *hav(θ)*, odnosno $sin^2(\theta/2)$ prikažemo direktno iz geografske širine i dužine. Označimo geografsku širinu s *φ*, a geografsku dužinu s *λ*. U našem slučaju, možemo os *y* uzeti kao geografsku širinu, a *x* kao geografsku dužinu. Budući da radimo s udaljenostima, određujemo razliku geografske širine i dužine:

$$hav(\theta) = hav(\varphi_2 - \varphi_1 + \cos(\varphi_1)\cdot\cos(\varphi_2)\cdot hav(\lambda_2 - \lambda_1)).$$

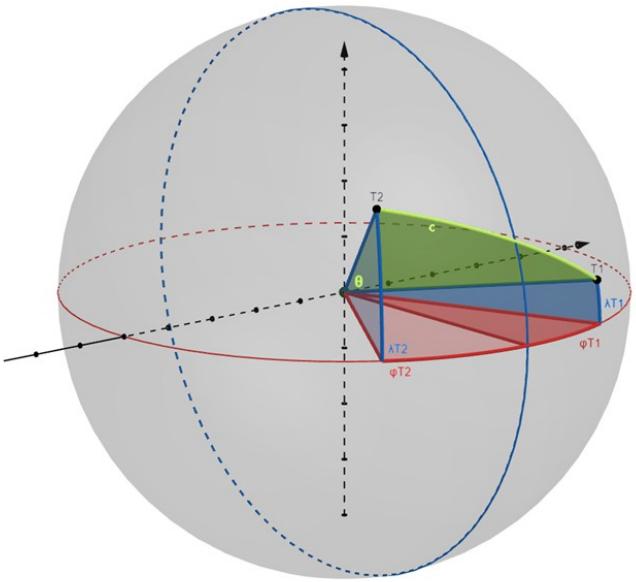
Nama je potrebna udaljenost *c*. Izrazimo je preko formule:

$$c = 2\,archav(h) = 2\,archav(\sqrt{h})$$

$$c = 2r\cdot\arcsin(\sqrt{hav(\varphi_2-\varphi_1)+(1-hav(\varphi_1-\varphi_2)-hav(\varphi_1+\varphi_2))\cdot hav(\lambda_2-\lambda_1)})$$

$$c = 2r\cdot\arcsin\left(\sqrt{\sin^2\left(\frac{\varphi_2-\varphi_1}{2}\right)+\left(1-\sin^2\left(\frac{\varphi_1-\varphi_2}{2}\right)-\sin^2\left(\frac{\varphi_1+\varphi_2}{2}\right)\right)\cdot\sin^2\left(\frac{\lambda_2-\lambda_1}{2}\right)}\right)$$

$$c = 2r\cdot\arcsin\left(\sqrt{\sin^2\left(\frac{\varphi_2-\varphi_1}{2}\right)+\cos(\varphi_1)\cdot\cos(\varphi_2)\cdot\sin^2\left(\frac{\lambda_2-\lambda_1}{2}\right)}\right)$$



Slika 8: Sferna distanca, označena zelenom linijom c

2.6. Izračun optimalnog puta

Definirali smo sferni koordinatni sustav te način na koji računamo udaljenosti između točaka u tom sustavu. Koristit ćemo Haversine formulu da bismo odredili duljinu određenog puta na mapi. U našem slučaju, gledat ćemo površinu Zemlje kao ravninu na zakrivljenoj plohi, s geografskom širinom i geografskom dužinom koje predstavljaju *x*-os i *y*-os te ćemo joj pridružiti nadmorsku visinu kao reprezentaciju *z*-osi.

Kad budemo imali reprezentaciju puta u trodimenzionalnom prostoru, korisniku ćemo ponuditi prikaz zadanog puta kao funkcije u dvodimenzionalnom kooridnatnom sustavu, kod kojeg će *x*-os prikazivati odabrani put, a *y*-os nadmorsku visinu. Budući korisnik unosi put proizvljno, točke kroz koje taj put prolazi mogu se veoma lako naći na mjestima koja su u stvarnosti neprohodna. Zato ćemo omogućiti korisniku pronalaženje optimalnijeg puta koji će biti automatski generiran te kojeg reprezentira funkcija s manjim nagibima na svojim podsegmentima.

3. Web aplikacija

3.1. Struktura HTML stranice

Program koji prati ovaj rad realiziran je u obliku aplikacije za web, koja se sastoji od tri datoteke pisane u HTML-u, CSS-u i JavaScript-u, pri čemu je realizacija matematičkog dijela odnosno glavni dio aplikacije pisan u JavaScript-u. U HTML datoteci *index.html* nalazi se kod podijeljen na pet odjeljaka: header, map, commands, display i footer. Header i footer sadrže nefunkcionalne dijelove stranice poput naslova i potpisa.

```
<!DOCTYPE html>

<html>
    <head>
        <link rel="stylesheet" href="style.css">

        <meta charset="utf-8">
        <title>Mapa</title>
    </head>
    <body>

        <div class="container">
            <div class="header">
                <h1>Numeričke metode i određivanje optimalnog puta u prostoru</h1>
            </div>
                <div id="map"></div>
                <div id="commands">
                    <button id="btn_1" onclick="addPath()">Zadaj putanju</button>
                    <button id="btn_2" onclick="display1()">Izračunaj duljinu zadane putanje</button>
                    <button id="btn_3">Stvori novi set markera</button>
                    <button id="btn_4">Stvori novu putanju</button>
                    <button id="btn_5" onclick="display2()">Izračunaj duljinu nove putanje</button>
                    <button id="btn_7" onclick="clearMarkers()">Prikaži samo novu putanju</button>
                    <button id="btn_6" onclick="refresh()">Ponovo učitaj stranicu</button>
                </div>
                <div id="display1"></div>
                <div id="display2"></div>

                <div id="error_display"></div>
            <div class="footer">
                Domagoj Palinić | Završni rad | Fakultet informatike i digitalnih tehnologija, 2022.
            </div>
        </div>

        <script async defer src="https://maps.googleapis.com/maps/api/js?key=AIzaSyDWL_qYAhwStzWG-IOOR_IK5lekQ3JM8os&callback=initMap"></script>
        <script src="script.js"></script>
    </body>
</html>
```

Slika 9: HTML kod

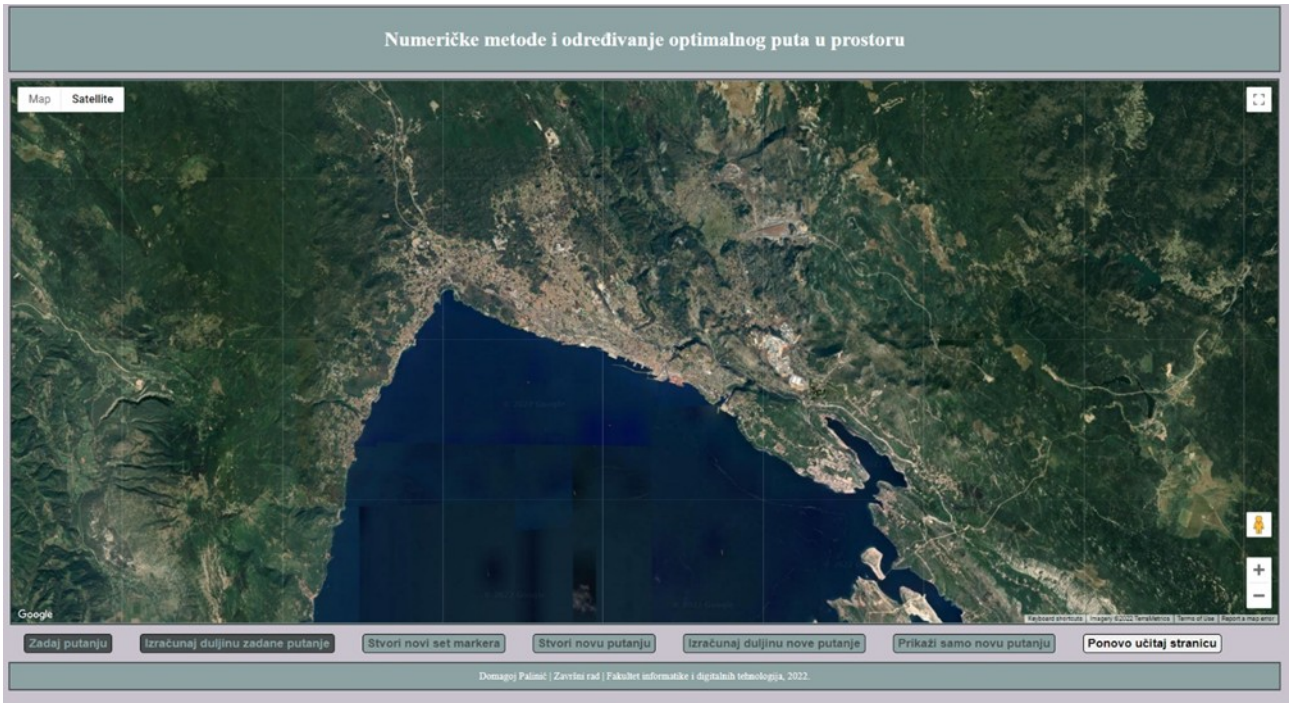
Google mapa koja je središnja komponenta ovog rada nalazi se u odjeljku „map“ te je putem poveznice <script> povezana sa JavaScript datotekom u kojoj se nalazi programski kod te API ključem potrebnim za učitavanje mape.

Odjeljak „commands“ sadrži sedam gumba, odnosno komponenata <button></button> koje predstavljaju sedam radnji koje su korisniku omogućene:

- zadavanje proizvoljne putanje dodavanjem markera klikom na Google mapi,

- izračun duljine zadane putanje i ispis te duljine u metrima,
- stvaranje novog skupa markera koji se koristi da bi se izračunao alternativni odnosno optimalniji put od unesenog,
- stvaranje nove putanje uz pomoć prethodno stvorenih markera,
- izračun duljine nove putanje te ispis duljine u metrima,
- sakrivanje zadane putanje te pomoćnih markera, pri čemu se prikazuje samo nova putanja,
- ponovo učitavanje stranice.

Ispod Google mape i odjeljka s gumbima nalazi se prostor za ispis duljine putanje koju je unio korisnik te prostor za ispis duljine izračunate putanje, odnosno optimalnog puta.



Slika 10: Prikaz stranice sa Google Mapom i gumbima

3.2. Funkcija initMap, postavljanje Google Mape

Google Maps API funkcionira na način da Google mapa koju prikazujemo poziva funkciju kojoj je unaprijed zadan naziv „initMap“. Prema tome, glavnu funkciju u kojoj definiramo sve radnje koje izravno koriste Google mapu u našoj JavaScript datoteci nazivamo „initMap“.

Da bismo uopće prikazali mapu na našoj stranici, potrebno je najprije stvoriti instancu mape u obliku varijable s određenim opcijama. Ovo realiziramo tako da prvo definiramo varijablu *options*, koja je u stvari struktura sastavljena od tri varijable:

- zoom – numerička vrijednost koja zadaje početno uvećanje prikaza mape,
- center – struktura koja se sastoji od parametara *lat* i *lng*, koji predstavljaju geografsku širinu i geografsku dužinu točke na koju je mapa centrirana u trenutku njenog učitavanja na stranici,
- mapTypeId – parametar kojem dodjeljujemo vrijednost ovisno o vrsti mape koju želimo prikazati, u našem slučaju prikazujemo satelitsku sliku.

Nakon što smo definirali varijablu *options*, mapu instanciramo novom varijablom tipa google.maps.Map koju nazivamo proizvoljno, u našem slučaju to je „map“. Pomoću naredbe document.getElementById povezujemo ju s lokacijom odjeljka map u našoj HTML datoteci te varijabli proslijeđujemo varijablu *options*, kako bi se prethodno definirane opcije primjenile na našu mapu.

```
12  function initMap(){
13      var options = {
14          zoom: 12,
15          center: { lat: 45.3322013185171, lng: 14.436759450945985 },
16          mapTypeId: 'satellite',
17      }
18
19      var map = new google.maps.Map(document.getElementById('map'), options);
```

Slika 11: Funkcija initMap, instanciranje Google mape

3.3. Instanciranje novih markera, funkcije addMarker i getElevation

Instanciranje novih markera na mapi vrši se pozivom funkcije *addMarker* koja prima parametar *e*, koji označuje *event*, odnosno klik na nekom mjestu na mapi. Definicija ove funkcije nalazi se unutar funkcije *initMap*, budući prilikom dodavanja novih markera radimo izravno s Google mapom.

Novi marker instanciramo tako što definiramo novu varijablu tipa *google.maps.Marker*, naziva *marker*, s parametrima:

- position – pozicija markera na mapi, sadrži strukturu e.coords odnosno koordinate točke na koju se dogodio klik na mapi,
- map – referenca na prethodno definiranu instancu mape,
- icon – ikona markera koja može biti unaprijed zadana ili posebno definirana.

```
43    async function addMarker(e){
44
45        var marker = new google.maps.Marker({
46            position: e.coords,
47            map: map,
48            icon: iconBlue
49        })
50        all_markers.push(marker);
51
52        var lat = marker.getPosition().lat();
53        var lng = marker.getPosition().lng();
54        var elev = await getElevation(marker);
55
56        var obj = {
57            lat: lat,
58            lng: lng,
59            elev: elev
60        }
61        var polyline_obj = {
62            lat: lat,
63            lng: lng
64        }
65
66        markers.push(obj);
67        polylinePath.push(polyline_obj);
68    }
69
70    var polylineOptions = {
71        path: polylinePath,
72        strokeColor: "blue",
73    }
74    var polyline = new google.maps.Polyline(polylineOptions);
75    polyline.setMap(map);
```

Slika 12: Funkcija addMarker, varijabla polylineOptions, varijabla polyline

Nakon što je instanciran novi marker, dodajemo ga u listu *all_markers* pomoću naredbe *push*.

Definiramo tri nove varijable: *lat*, *lng* i *elev*, koje će nam biti potrebne za stvaranje objekata koje dalje dodajemo u liste *markers* i *polylinePath*. Varijablama *lat* i *lng* dodajemo vrijednosti geografske dužine i širine za varijablu *marker* koju smo upravo stvorili, dok varijabli *elev* dodajemo vrijednost nadmorske visine izračunate pomoću funkcije *getElevation*.

Funkcija *getElevation* je asinkrona funkcija koja putem URL-a šalje zahtjev serveru da bi vratila iznos nadmorske visine za neku određenu točku na mapi.

```
206  async function getElevation(marker) {
207      var lat = marker.getPosition().lat();
208      var lng = marker.getPosition().lng();
209      var url = 'https://maps.googleapis.com/maps/api/elevation/json?locations=' + lat + '%2C' + lng + '&key=AIzaSyDWL_qYAhwStzWG-IOOR_IK5lekQ3JM8os';
210      var response = await fetch(url);
211      var elevation_data = await response.json();
212
213      return new Promise((accept, reject) => {
214          accept(elevation_data.results[0].elevation)
215      })
216  }
```

Slika 13: Funkcija getElevation

URL koji koristimo da bi poslali zahtjev serveru moramo dinamički stvoriti unutar same funkcije, kao varijablu *url*. Ona se sastoji od nekoliko dijelova:

- fiksnog dijela URL-a koji se ne mijenja ovisno o točki za koju tražimo visinu,
- varijabli *lat* i *lng* koje popunjavamo tako što putem funkcija *getPosition().lat()* i *getPosition().lng()* dohvaćamo geografsku širinu i dužinu za određeni marker na mapi,
- stringa *'%2C'* koji odvaja geografsku širinu i dužinu,
- API ključa, kojeg koristimo za instanciranje naše Google mape.

Nakon što smo stvorili URL, pomoću naredbe *await fetch()* dohvaćamo odgovor servera i spremamo ga u varijablu *response*, koju zatim pretvaramo u json oblik i njenu vrijednost spremamo u varijablu *elevation_data*. Zahtjev će vratiti objekt tipa *Promise* koji sadrži rezultat u json obliku. Na kraju, vraćamo vrijednost nadmorske visine za točku na mapi, koja se u json objektu *elevation_data* nalazi na lokaciji *.results[0].elevation*.

```
{
   "results" : [
      {
         "elevation" : 45.96683120727539,
         "location" : {
            "lat" : 45.3322013185171,
            "lng" : 14.43675945094598
         },
         "resolution" : 610.8129272460938
      }
   ],
   "status" : "OK"
}
```


Slika 14: Primjer json objekta vraćenog u funkciji getElevation

Kada smo dohvatili vrijednost za varijablu *elev* pomoću funkcije getElevation, imamo sve tri varijable koje su nam potrebne za stvaranje objekata *obj* i *polyline_obj* koje dalje koristimo za popunjavanje listi *markers* i *polylinePath*.

Nakon što smo definirali novi marker, instanciramo ga pomoću naredbe *google.maps.event.addListener* kojoj proslijeđujemo referencu na našu instancu mape, parametar *'click'* kojim specificiramo da očekujemo klik na mapi te ukoliko varijabla *add_path* ima vrijednost *true* pozivamo prethodno definiranu funkciju *addMarker*.

```
77      google.maps.event.addListener(map, 'click', function(e){
78          if(add_path){
79              addMarker({coords: e.latLng});
80
81              var currentPath = polyline.getPath();
82              currentPath.push(e.latLng);
83          }
84      })
```

Slika 15: Dodavanje novog markera pozivom funkcije addMarker, ažuriranje putanje za polyline

Na kraju, potrebno je na našoj mapi ucrtati liniju koja spaja dodane markere kako bi se jasno prikazalo da je riječ o putanji. Na Slici 12 prikazana je definicija objekta tipa google.maps.Polyline, odnosno linije koja spaja prethodno unesene markere.

Najprije definiramo varijablu *polylineOptions*, koja je struktura sa parametrima:

- *path* – putanja definirana kao lista objekata koji sadrže geografsku širinu i dužinu za neku točku na mapi, između kojih crtamo liniju,
- *strokeColor* – boja linije koju crtamo.

Dodatno, varijabla *polylineOptions* može sadržavati i parametar strokeWeight koji određuje debljinu linije, no u ovom slučaju ga ne navodimo.

Kada smo definirali opcije, instanciramo novi objekt tipa google.maps.Polyline sa parametrom *polylineOptions* te ga postavljamo na mapu naredbom .setMap(). Prilikom dodavanja svakog novog markera, dohvaćamo trenutnu putanju za naš google.maps.Polyline objekt te joj dodajemo poziciju upravo stvorenog markera, kako je prikazano na Slici 12.

Slika 16: Prikaz nove putanje, zadane između dva dodana markera, koji su obilježeni plavim točkama

3.4. Mjerenje udaljenosti, Haversine formula

Mjerenje udaljenosti, odnosno duljine putanje koju je na mapi zadao korisnik izvodimo pomoću tri funkcije. Prva funkcija koju koristimo jest funkcija *haversine*, kojom realiziramo Haversine formulu.

```
222  function haversine(marker1, marker2) {
223
224      var lat1 = marker1.lat;
225      var lng1 = marker1.lng;
226      var lat2 = marker2.lat;
227      var lng2 = marker2.lng;
228
229      var pi_x = Math.PI / 180;
230      var dlng = (lng2 * pi_x) - (lng1 * pi_x);
231      var dlat = (lat2 * pi_x) - (lat2 * pi_x);
232
233      var a = (Math.sin(dlat / 2)) * (Math.sin(dlat / 2)) + (Math.cos(lat1 * pi_x)) * (Math.cos(lat2 * pi_x)) * (Math.sin(dlng / 2)) * (Math.sin(dlng / 2));
234      var c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));
235      var r = 6378.137;
236      var d = c * r
237      return d;
238  }
```

Slika 17: Funkcija Haversine

Funkcija prima dva argumenta, *marker1* i *marker2*, s obzirom da njome računamo udaljenost između dvije točke na sferi, odnosno dva markera na Google mapi. Najprije od dva markera dohvaćamo vrijednosti njihove geografske širine i geografske dužine. Budući da u ovom slučaju ne radimo s objektima samih markera, već s objektima koje smo proslijedili u listu *markers*, nije potrebno koristiti funkcije *getPosition().lat()* i *getPosition().lng()*, već jednostavno uzimamo već spremljene vrijednosti u listi.

Varijabla *pi_x* ima vrijednost *π/100* te je koristimo kao pomoćnu varijablu prilikom izračuna, kao i varijalbu *d* čija je vrijednost 6378.137. Funkcijama sinus i kosinus, kao i vrijednosti *π* pristupamo pomoću ugrađene biblioteke Math.

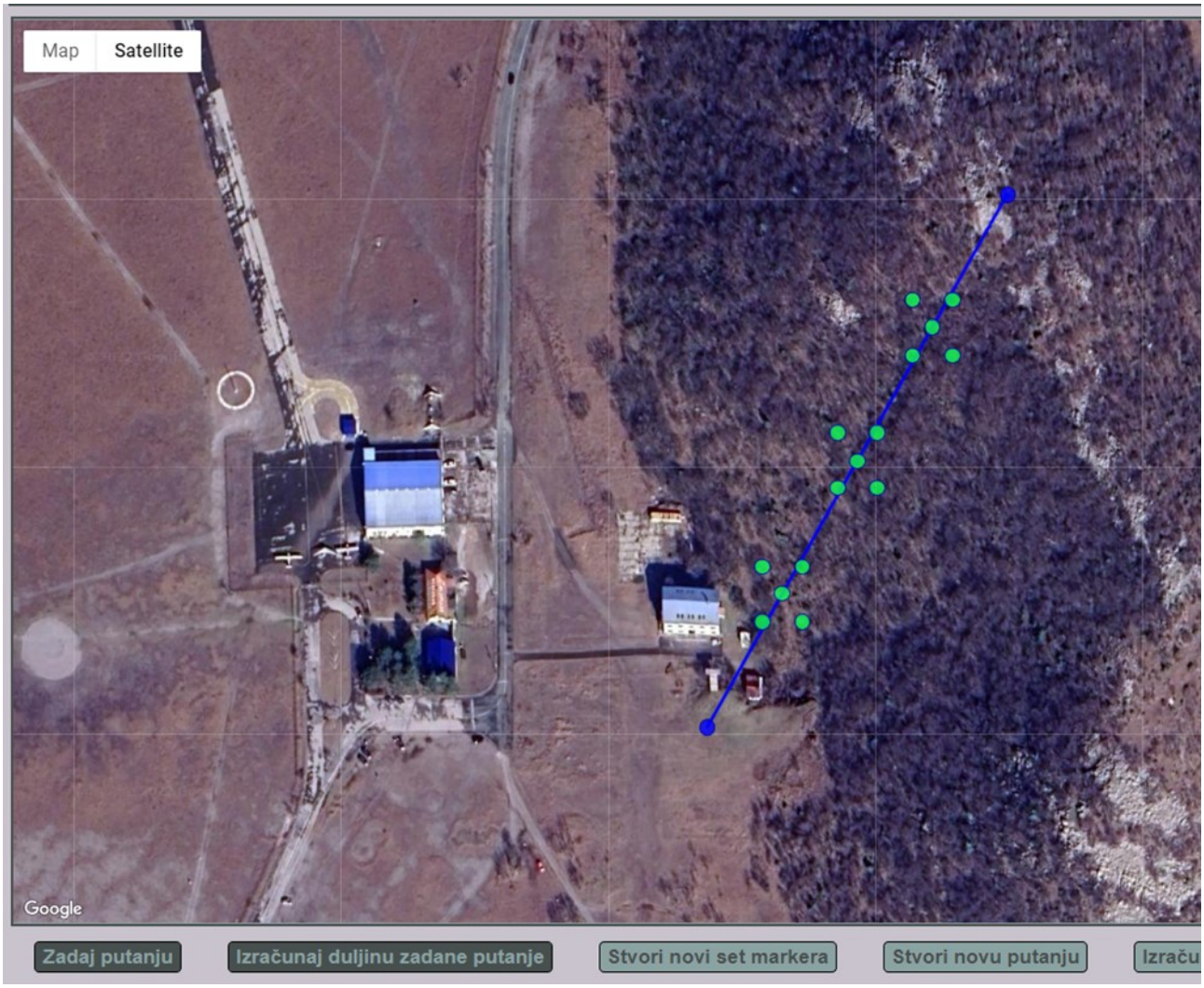
Na kraju izračuna vraćamo iznos duljine između dvije točke na sferi. Budući većina razmaknutih točaka na Zemlji nije na istoj nadmorskoj visini, potreban nam je način na koji ćemo izračunati udaljenost između njih, za što koristimo Pitagorin poučak.

```
240  function pythagoras(marker1, marker2){
241      var h1 = marker1.elev;
242      var h2 = marker2.elev;
243      var d = haversine(marker1, marker2)*1000;
244      var h3 = Math.abs(h1-h2);
245
246      var c = Math.sqrt((h3*h3) + (d*d));
247
248      return c;
249  }
```

Slika 18: Pitagorin poučak

Dohvaćajući visinu za dva markera između kojih mjerimo udaljenost, možemo izračunati razliku u visini između ta dva markera. Varijablom h_3 predstavljamo apsolutnu vrijednost razlike u visini, odnosno jednu katetu pravokutnog trokuta. Drugu katetu dobivamo pomoću funkcije *haversine*. Na kraju, primjenom Pitagorinog poučka vraćamo duljinu hipotenuze.

Na kraju, potrebno je izračunati duljinu čitave putanje, ukoliko je unesena putanja koja se sastoji od više od dva markera. Navedeno realiziramo jednostavnom funkcijom *measure*, kod koje pomoću *for* petlje prolazimo kroz listu koja sadrži sve markere u listi *markers* te prilikom svakog inkrementa povećavamo vrijednost varijable *dd* koja predstavlja zbroj svih udaljenosti između markera. Konačno, kada smo zbrojili iznose udaljenosti za sve segmente, taj zbroj vraćamo i prikazujemo na stranici pomoću naredbe *innerHTML*, prikazan u metrima i zaokružen na tri decimale, što se vidi na mjestu na kojem se nalazi odjeljak *display1* u HTML datoteci.



Slika 19: Segment putanje sa skupom novih markera, označenih zelenom bojom

3.5. Stvaranje novog skupa markera za izračun nove putanje

Putanja koju korisnik unosi tako što dodaje markere na Google mapu u većini slučajeva neće biti optimalna. Drugim riječima, ta putanja može prelaziti preko dijelova koji su neprohodni ili imati velike razlike u visini između pojedinih markera. Klikom na gumb *Stvori novi set markera* pokreće se dio programa koji stvara skup markera koji služe za stvaranje optimalnijeg puta između prve i zadnje točke na zadanom putu.

Ideja je sljedeća: stvorimo tri nova markera na svakom podsegmentu, jednako udaljena jedan od drugoga i od početnog i završnog te četiri markera blizu svakog od tri nova. Time dobivamo na svakom podsegmentu putanje po petnaest novih markera, ili tri grupe po pet markera, koje dalje koristimo da bismo izračunali optimalniji put.

Da bismo dodali novi skup markera na svakom podsegmentu, potrebno je najprije podijeliti postojeći segment na četiri nova podsegmenta, odnosno dodati svakom od tri skupa markera poziciju na koju stavljamo središnji od pet markera. Za ovu svrhu definiramo funkciju *addOffsetMarker*, koja je u stvari modifikacija već postojeće funkcije *addMarker i* koja prima argumente:

- *lat* – geografska širina novog markera,
- *lng* – geografska dužina novog markera,
- *lat_offset* – razlika u geografskoj širini u odnosu na parametar *lat*,
- *lng_offset* – razlika u geografskoj dužini u odnosu na parametar *lng*,
- *group* – lista koja predstavlja jednu od tri grupe markera na segmentu.

```
86    async function addOffSetMarker(lat, lng, lat_offset, lng_offset, group){
87        var new_marker = new google.maps.Marker({
88            position: {
89                lat: lat + (lat_offset),
90                lng: lng + (lng_offset)
91            },
92            map: map,
93            icon: iconGreen
94        })
95        all_markers.push(new_marker);
96        var new_elev = await getElevation(new_marker);
97        var obj = {
98            lat: new_marker.getPosition().lat(),
99            lng: new_marker.getPosition().lng(),
100           elev: new_elev
101       };
102       group.push(obj);
103   }
```

Slika 20: Funkcija addOffsetMarker

Novi marker stavljamo na poziciju koja je određena zbrojem parametara *lat* i *lat_offset* te zbrojem parametara *lng* i *lng_offset*. Ostatak funkcije je isti kao kod funkcije *addMarker*, uz dvije iznimke: ovoga puta ne popunjavamo listu za *polyline* te stvoreni objekt koji sadrži geografsku širinu, geografsku dužinu i nadmorsku visniu ubacujemo u listu *group*.

Da bi svaki od tri nova skupa markera bio na istoj udaljenosti, računamo poziciju točke koja se nalazi na pola puta između dva markera koji određuju segment. Opisano radimo tako da zbrojimo geografsku širinu ili geografsku dužinu dva markera i podijelimo je s 2. Na taj način smo dobili vrijednosti za varijable *mid_lat* i *mid_lng*. Nadalje, uzimamo njihove vrijednosti kao početak ili kraj segmenta te ponavljamo postupak još dva puta te time dobivamo vrijednosti za varijable *l_lat*, *l_lng* i *r_lat*, *r_lng*. Konačno, dobili smo geografsku širinu i geografsku dužinu za tri nova markera, jednako udaljena jedan od drugoga koji dijele zadani segment na četiri nova podsegmenta jednake duljine. Grupe podsegmenata nazvane su *left_group*, *mid_group* i *right_group* samo radi jednostavnosti. Nije nužno da je jedna grupa na lijevoj, a druga na desnoj strani mape.

Ostala četiri markera, koji okružuju svaki od tri nova markera dodaju se postavljanjem parametara *lat_offset* i *lng_offset*. U ovom slučaju zadani su na vrijednosti reda veličine 0.0001, iako ta vrijednost u stvari može biti i proizvoljna. Na kraju funkcije dodajemo sve tri grupe markera

u listu *new_route*.

```
105    async function calculateNewMarkers(list, i1, i2){
106
107        // middle point coords
108        var mid_lat = (list[i1].lat + list[i2].lat)/2;
109        var mid_lng = (list[i1].lng + list[i2].lng)/2;
110
111        // middle point left coords
112        var l_lat = (list[i1].lat + mid_lat)/2;
113        var l_lng = (list[i1].lng + mid_lng)/2;
114
115        // middle point right coords
116        var r_lat = (mid_lat + list[i2].lat)/2;
117        var r_lng = (mid_lng + list[i2].lng)/2;
118
119        // marker groups
120        var left_group = [];
121        var mid_group = [];
122        var right_group = [];
123
124        // create new markers for left_group
125        addOffSetMarker(l_lat, l_lng, 0, 0, left_group);
126        addOffSetMarker(l_lat, l_lng, 0.0001, 0.0001, left_group);
127        addOffSetMarker(l_lat, l_lng, -0.0001, -0.0001, left_group);
128        addOffSetMarker(l_lat, l_lng, 0.0001, -0.0001, left_group);
129        addOffSetMarker(l_lat, l_lng, -0.0001, 0.0001, left_group);
130
131        // create new markers for mid_group
132        addOffSetMarker(mid_lat, mid_lng, 0, 0, mid_group);
133        addOffSetMarker(mid_lat, mid_lng, 0.0001, 0.0001, mid_group);
134        addOffSetMarker(mid_lat, mid_lng, -0.0001, -0.0001, mid_group);
135        addOffSetMarker(mid_lat, mid_lng, 0.0001, -0.0001, mid_group);
136        addOffSetMarker(mid_lat, mid_lng, -0.0001, 0.0001, mid_group);
137
138        // create new markers for right_group
139        addOffSetMarker(r_lat, r_lng, 0, 0, right_group);
140        addOffSetMarker(r_lat, r_lng, 0.0001, 0.0001, right_group);
141        addOffSetMarker(r_lat, r_lng, -0.0001, -0.0001, right_group);
142        addOffSetMarker(r_lat, r_lng, 0.0001, -0.0001, right_group);
143        addOffSetMarker(r_lat, r_lng, -0.0001, 0.0001, right_group);
144
145        // push all
146        new_route.push(left_group);
147        new_route.push(mid_group);
148        new_route.push(right_group);
149    }
```

Slika 21: Dodavanje novih markera pomoću funkcije calculateNewMarkers

3.6. Račun i crtanje nove putanje

Jednom kad smo zadali novi skup markera za svaki podsegment zadane putanje, potrebno je stvoriti novu putanju koja će predstavljati optimalniji put između početne i završne točke na putanji koju je zadao korisnik.

Najprije u listu *new_route_markers* dodajemo prvi element liste *markers*, čime osiguravamo da je početna točka nove putanje ista kao i početna točka zadane putanje. Zatim pomoću dvije ugnježđene *for* petlje prolazimo kroz novi skup markera. Vanjska *for* petlja prolazi kroz sve elemente liste *new_route*, dok unutarnja petlja prolazi kroz svakih pet elemenata. Drugim riječima, svaka iteracija vanjske petlje je grupa koja sadrži pet markera (*left_group, mid_group, right_group*), a svaka iteracija unutarnje petlje pojedinačni marker u svakoj od tri grupe.

Prije prolaska kroz svaku od tri grupe definiramo varijablu *pyth_min* kojoj dodijeljujemo vrijednost udaljenosti između prethodnog markera i prvog markera u pojedinoj grupi. Zatim prolazimo kroz pozicije ostala četiri markera te uspoređujemo udaljenost između svakog od ta četiri markera i posljednjeg markera kojeg smo dodali u listu *new_route_markers*. Koristimo pomoćnu varijablu *pyth_index*, početne vrijednosti 0, kako bismo spremili indeks onog markera unutar grupe čija je udaljenost od prethodne točke najmanja. Ako je neka od tih udaljenosti manja od prethodno definirane udaljenosti *pyth_min*, vrijednost varijable *pyth_index* zamjenjujemo indeksom tog određenog markera. Kada unutarnja *for* petlja prođe kroz svih pet svojih iteracija, u listu *new_route_markers* dodajemo marker s indeksima *i*, *pyth_index*.

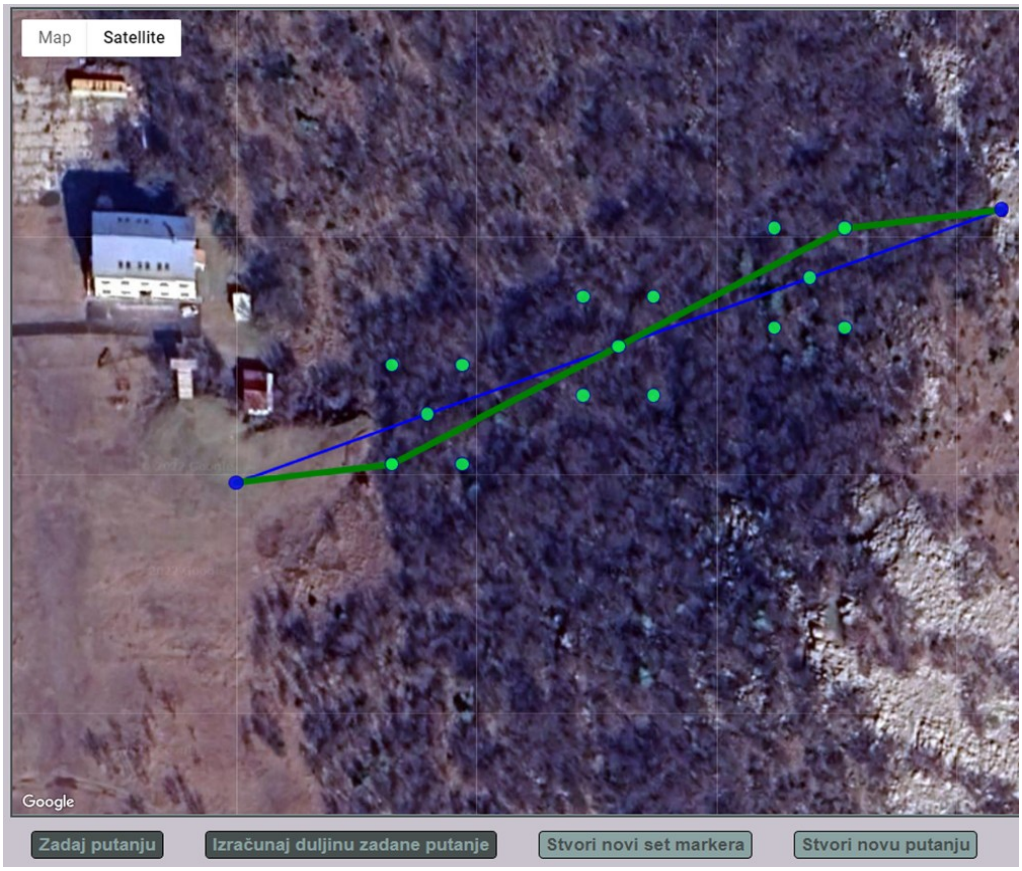
```
159     document.getElementById("btn_4").addEventListener('click', function(){
160         new_route_markers.push(markers[0]);
161         for (var i=0; i<new_route.length; i++){
162             var pyth_min = pythagoras(new_route_markers[i], new_route[i][0]);
163             for(var j=0; j<5; j++){
164                 var pyth = pythagoras(new_route_markers[i], new_route[i][j]);
165                 var pyth_index = 0;
166                 if (pyth < pyth_min){
167                     pyth_index = j;
168                 }
169             }
170             new_route_markers.push(new_route[i][pyth_index]);
171         }
172         var markers_limit = markers.length-1;
173         new_route_markers.push(markers[markers_limit]);
174 
175         for(var i=0; i<new_route_markers.length; i++){
176             var new_polylineObj = {
177                 lat: new_route_markers[i].lat,
178                 lng: new_route_markers[i].lng
179             };
180             new_polylinePath.push(new_polylineObj);
181         }
182 
183         var new_polylineOptions = {
184             path: new_polylinePath,
185             strokeColor: "green",
186             strokeWeight: 7
187         }
188         var new_polyline = new google.maps.Polyline(new_polylineOptions);
189         new_polyline.setMap(map);
190     })
191 
192     document.getElementById("btn_7").addEventListener('click', function(){
193         for(var i=0; i<all_markers.length; i++){
194             all_markers[i].setMap(null);
195             polyline.setMap(null);
196         }
197     })
198 }
```

Slika 22: Kreiranje nove putanje

Jednom kada obje petlje završe s radom, potrebno je dodati u listu *new_route_markers* i posljednju točku putanje koju je zadao korisnik, što radimo tako što najprije dohvaćamo duljinu liste *markers* umanjenu za 1 te je spremamo u pomoćnu varijablu *markers_limit*, koja predstavlja indeks posljednjeg elementa u listi *markers*. Zatim u listu *new_route_markers* dodajemo posljednji element te liste. Na ovaj način popunili smo listu *new_route_markers* markerima koji određuju optimalniji put od zadanog.

Naposljetku, potrebno je nacrtati novu putanju, koristeći objekt tipa *google.maps.Polyline*. Pomoću *for* petlje prolazimo kroz sve elemente liste *new_route_markers* te za svaki element stvaramo novi objekt koji sadrži geografsku širinu i geografsku dužinu te ga dodajemo u listu *new_polylinePath*. Crtanje nove linije koja će predstavljati izračunatu putanju analogno je crtanju linije za putanju koju je zadao korisnik, uz dvije iznimke: nova putanja označena je drugom bojom te kod definiranja varijable *new_polylineOptions* ovaj put koristimo i parametar *strokeWeight*, a kako bismo bolje istaknuli novu putanju u odnosu na staru.

Na kraju, korisniku je omogućeno sakrivanje stare putanje, te svih pomoćnih markera, kako bi se na Google mapi prikazala samo nova putanja.



Slika 23: Prikaz nove putanje

4. Dodatni materijali – programski kod

4.1. Datoteka index.html

```
<!DOCTYPE html>

<html>
  <head>
    <link rel="stylesheet" href="style.css">

    <meta charset="utf-8">
    <title>Mapa</title>
  </head>
  <body>

    <div class="container">
      <div class="header">
        <h1>Numeričke metode i određivanje optimalnog puta u prostoru</h1>
      </div>
        <div id="map"></div>
        <div id="commands">
          <button id="btn_1" onclick="addPath()">Zadaj putanju</button>
          <button id="btn_2" onclick="display1()">Izračunaj duljinu zadane
putanje</button>
          <button id="btn_3">Stvori novi set markera</button>
          <button id="btn_4">Stvori novu putanju</button>
          <button id="btn_5" onclick="display2()">Izračunaj duljinu nove
putanje</button>
          <button id="btn_7" onclick="clearMarkers()">Prikaži samo novu
putanju</button>
          <button id="btn_6" onclick="refresh()">Ponovo učitaj
stranicu</button>
        </div>
        <div id="display1"></div>
        <div id="display2"></div>

        <div id="error_display"></div>
      <div class="footer">
        Domagoj Palinić | Završni rad | Fakultet informatike i digitalnih
tehnologija, 2022.
      </div>
    </div>

    <script async defer src="https://maps.googleapis.com/maps/api/js?
key=AIzaSyDWL_qYAhwStzWG-IOOR_IK5lekQ3JM8os&callback=initMap"></script>
```

```
    <script src="script.js"></script>
  </body>
</html>
```

4.2. Datoteka script.js

```
var markers = [];
var polylinePath = [];

var add_path = false;

var new_route = [];
var new_route_markers = [];
var new_polylinePath = [];

var all_markers = [];

function initMap(){
  var options = {
    zoom: 12,
    center: { lat: 45.3322013185171, lng: 14.436759450945985 },
    mapTypeId: 'satellite',
  }

  var map = new google.maps.Map(document.getElementById('map'), options);

  var iconBlue = {
    path: google.maps.SymbolPath.CIRCLE,
    fillColor: '#00F',
    fillOpacity: 0.8,
    strokeColor: '#00A',
    strokeOpacity: 0.9,
    strokeWeight: 1,
    scale: 7
  }

  var iconGreen = {
    path: google.maps.SymbolPath.CIRCLE,
    fillColor: ' #0FFF50',
    fillOpacity: 0.8,
    strokeColor: '#00A',
    strokeOpacity: 0.9,
```

```
  strokeWeight: 1,
  scale: 7
}



async function addMarker(e){

  var marker = new google.maps.Marker({
    position: e.coords,
    map: map,
    icon: iconBlue
  })
  all_markers.push(marker);

  var lat = marker.getPosition().lat();
  var lng = marker.getPosition().lng();
  var elev = await getElevation(marker);

  var obj = {
    lat: lat,
    lng: lng,
    elev: elev
  }
  var polyline_obj = {
    lat: lat,
    lng: lng
  }

  markers.push(obj);
  polylinePath.push(polyline_obj);
}

var polylineOptions = {
  path: polylinePath,
  strokeColor: "blue",
}
var polyline = new google.maps.Polyline(polylineOptions);
polyline.setMap(map);

google.maps.event.addListener(map, 'click', function(e){
  if(add_path){
    addMarker({coords: e.latLng});

    var currentPath = polyline.getPath();
    currentPath.push(e.latLng);
  }
})
```

```
async function addOffSetMarker(lat, lng, lat_offset, lng_offset, group){
  var new_marker = new google.maps.Marker({
    position: {
      lat: lat + (lat_offset),
      lng: lng + (lng_offset)
    },
    map: map,
    icon: iconGreen
  })
  all_markers.push(new_marker);
  var new_elev = await getElevation(new_marker);
  var obj = {
    lat: new_marker.getPosition().lat(),
    lng: new_marker.getPosition().lng(),
    elev: new_elev
  };
  group.push(obj);
}

async function calculateNewMarkers(list, i1, i2){

  // middle point coords
  var mid_lat = (list[i1].lat + list[i2].lat)/2;
  var mid_lng = (list[i1].lng + list[i2].lng)/2;

  // middle point left coords
  var l_lat = (list[i1].lat + mid_lat)/2;
  var l_lng = (list[i1].lng + mid_lng)/2;

  // middle point right coords
  var r_lat = (mid_lat + list[i2].lat)/2;
  var r_lng = (mid_lng + list[i2].lng)/2;

  // marker groups
  var left_group = [];
  var mid_group = [];
  var right_group = [];

  // create new markers for left_group
  addOffSetMarker(l_lat, l_lng, 0, 0, left_group);
  addOffSetMarker(l_lat, l_lng, 0.0001, 0.0001, left_group);
  addOffSetMarker(l_lat, l_lng, -0.0001, -0.0001, left_group);
  addOffSetMarker(l_lat, l_lng, 0.0001, -0.0001, left_group);
  addOffSetMarker(l_lat, l_lng, -0.0001, 0.0001, left_group);

  // create new markers for mid_group
  addOffSetMarker(mid_lat, mid_lng, 0, 0, mid_group);
  addOffSetMarker(mid_lat, mid_lng, 0.0001, 0.0001, mid_group);
  addOffSetMarker(mid_lat, mid_lng, -0.0001, -0.0001, mid_group);
```

```
    addOffSetMarker(mid_lat, mid_lng, 0.0001, -0.0001, mid_group);
    addOffSetMarker(mid_lat, mid_lng, -0.0001, 0.0001, mid_group);

    // create new markers for right_group
    addOffSetMarker(r_lat, r_lng, 0, 0, right_group);
    addOffSetMarker(r_lat, r_lng, 0.0001, 0.0001, right_group);
    addOffSetMarker(r_lat, r_lng, -0.0001, -0.0001, right_group);
    addOffSetMarker(r_lat, r_lng, 0.0001, -0.0001, right_group);
    addOffSetMarker(r_lat, r_lng, -0.0001, 0.0001, right_group);

    // push all
    new_route.push(left_group);
    new_route.push(mid_group);
    new_route.push(right_group);
  }

  document.getElementById("btn_3").addEventListener('click', async
function(){
      for (var i=0; i<(markers.length-1); i++){
        var j = i+1;
        await calculateNewMarkers(markers, i, j);
      }

  })

  document.getElementById("btn_4").addEventListener('click', function(){
    new_route_markers.push(markers[0]);
    for (var i=0; i<new_route.length; i++){
      var pyth_min = pythagoras(new_route_markers[i], new_route[i][0]);
      for(var j=0; j<5; j++){
        var pyth = pythagoras(new_route_markers[i], new_route[i][j]);
        var pyth_index = 0;
        if (pyth < pyth_min){
          pyth_index = j;
        }
      }
      new_route_markers.push(new_route[i][pyth_index]);
    }
    var markers_limit = markers.length-1;
    new_route_markers.push(markers[markers_limit]);

    for(var i=0; i<new_route_markers.length; i++){
      var new_polylineObj = {
        lat: new_route_markers[i].lat,
        lng: new_route_markers[i].lng
      };
      new_polylinePath.push(new_polylineObj);
    }
```

```
    var new_polylineOptions = {
      path: new_polylinePath,
      strokeColor: "green",
      strokeWeight: 7
    }
    var new_polyline = new google.maps.Polyline(new_polylineOptions);
    new_polyline.setMap(map);
  })

  document.getElementById("btn_7").addEventListener('click', function(){
    for(var i=0; i<all_markers.length; i++){
      all_markers[i].setMap(null);
      polyline.setMap(null);
    }
  })
}

function compareElevations(elev1, elev2, treshold){
  if (Math.abs(elev1 - elev2) > treshold) {
    return true;
  }
}

async function getElevation(marker) {
  var lat = marker.getPosition().lat();
  var lng = marker.getPosition().lng();
  var url = 'https://maps.googleapis.com/maps/api/elevation/json?locations='
+ lat + '%2C' + lng + '&key=AIzaSyDWL_qYAhwStzWG-IOOR_IK5lekQ3JM8os';
  var response = await fetch(url);
  var elevation_data = await response.json();

  return new Promise((accept, reject) => {
    accept(elevation_data.results[0].elevation)
  })
}

function addPath(){
  add_path = true;
}

function haversine(marker1, marker2) {

  var lat1 = marker1.lat;
  var lng1 = marker1.lng;
  var lat2 = marker2.lat;
  var lng2 = marker2.lng;

  var pi_x = Math.PI / 180;
  var dlng = (lng2 * pi_x) - (lng1 * pi_x);
```

```
  var dlat = (lat2 * pi_x) - (lat2 * pi_x);

  var a = (Math.sin(dlat / 2)) * (Math.sin(dlat / 2)) + (Math.cos(lat1 *
pi_x)) * (Math.cos(lat2 * pi_x)) * (Math.sin(dlng / 2)) * (Math.sin(dlng /
2));
  var c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));
  var r = 6378.137;
  var d = c * r
  return d;
}

function pythagoras(marker1, marker2){
  var h1 = marker1.elev;
  var h2 = marker2.elev;
  var d = haversine(marker1, marker2)*1000;
  var h3 = Math.abs(h1-h2);

  var c = Math.sqrt((h3*h3) + (d*d));

  return c;
}

function measure(m) {
  var dd = 0
  for (var i = 0; i < (m.length - 1); i++) {
    dd += pythagoras(m[i], m[i + 1]);
  }
  return dd;
}

function refresh() {
  window.location.reload();
}

function display1(){
  document.getElementById("display1").innerHTML = "Duljina unesenog puta: "
+ measure(markers).toFixed(3) + " m";
}

function display2(){
  document.getElementById("display2").innerHTML = "Duljina alternativnog
puta: " + measure(new_route_markers).toFixed(3) + " m";
}
```

4.3. Datoteka style.css

```
body {
  background-color: #c8c3cc;
}

#map {
  display: block;
  margin-left: auto;
  margin-right: auto;
  width: 99%;
  height: 865px;
  border: 3px solid  #484f4f;
  background-color: #8ca3a3;

}

#commands{
  display :block;
  width: 99%;
  margin-left: auto;
  margin-right: auto;
  height: 10%;
  margin-top: 10px;
  margin-bottom: 10px;
}
#commands button {
  display: inline-block;
  margin-top: 5px;
  margin-bottom: 5px;
  margin-left: 20px;
  margin-right: 20px;
  font-size: larger;
  font-weight: bold;
  border-radius: 5px;
}
#btn_1 {
  background-color: #484f4f;
  color: #8ca3a3;
}
#btn_2 {
  background-color: #484f4f;
  color: #8ca3a3;
}
#btn_3 {
  background-color: #8ca3a3;
  color: #484f4f;
}
#btn_4 {
```

```
  background-color: #8ca3a3;
  color: #484f4f;
}
#btn_5 {
  background-color: #8ca3a3;
  color: #484f4f;
}
#btn_7 {
  background-color: #8ca3a3;
  color: #484f4f;
}
.header {
  background-color: #8ca3a3;
  display: block;
  margin-left: 5px;
  margin-right: 5px;
  margin-top: 10px;
  margin-bottom: 10px;
  padding-top: 10px;
  padding-bottom: 10px;
  color: white;
  border: 3px solid #484f4f;
  text-align: center;
}
.footer {
  background-color: #8ca3a3;
  display: block;
  margin-left: 5px;
  margin-right: 5px;
  margin-top: 10px;
  margin-bottom: 10px;
  padding-top: 10px;
  padding-bottom: 10px;
  color: white;
  border: 3px solid #484f4f;
  text-align: center;
}
```

Popis slika

Literatura

[1] Amanda Briney, The history of cartography, ThoughtCo (2019), pristup 28. kolovoza 2022., https://www.thoughtco.com/the-history-of-cartography-1435696

[2] Wikipedia, Majorcan cartographic school (2020), pristup 29. kolovoza 2022., https://en.wikipedia.org/wiki/Majorcan_cartographic_school#cite_ref-Riddle_1-0

[3] Lardinois, Frederic, pristup 28. kolovoza 2022., "For the Love of Mapping Data", TechCrunch (2014), https://techcrunch.com/2014/08/09/for-the-love-of-open-mapping-data/?guccounter=1

[4] "Google mapper: Take browsers to the limit". CNET. Archived from the original on October 26, 2012. Retrieved January 3, 2013.

[5] Orlowski, Andrew, "Google buys CIA-backed mapping startup",The Register, Archived (2014), pristup 30. kolovoza 2022., https://www.theregister.com/2004/10/28/google_buys_keyhole/

[6] Frank Tylor, Google Shows Pre-Katrina Photos for New Orleans, Google Earth Blog, 2007, pristup 10. rujna 2022., https://www.gearthblog.com/blog/archives/2007/03/google_shows_prekatr.html

[7] Google Maps User Guide, Internet Archive Way Back Machine, 2007, pristup 10. rujna 2022., https://web.archive.org/web/20071105004439/http:/maps.google.com/support/bin/answer.py?answer=68259#overview

[8] Google Announces Launch of Google Maps for Mobile With “My Location” Technology, Google Press, Blogspot, 2007, pristup 10. rujna 2022., https://googlepress.blogspot.com/2007/11/google-announces-launch-of-google-maps_28.html

[9] Robinson Meyer, Google’s Satellite Map Gets a 700 – Trillion-Pixel Makeover, The Atlantic, 2016, pristup 10. rujna 2022, https://www.theatlantic.com/technology/archive/2016/06/google-maps-gets-a-satellite-makeover-mosaic-700-trillion/488939/

[10] Andrew Liptak, Google Maps now depicts the Earth as a globe, The Verge, 2018, pristup 10. rujna 2022., https://www.theverge.com/2018/8/5/17653122/google-maps-update-mercator-projection-earth-isnt-flat

[11] Google, Google Maps Platform Documentation, Google Maps Platform, pristup 10. rujna 2022., https://developers.google.com/maps/documentation

[12] Ivan Slapničar, poglavlje „Vektorska algebra i analitička geometrija”, potpoglavlje „Vektori” u knjizi Matematika 1, Fakultet elektrotehnike, strojarstva i brodogradnje, Split, 2002

[13] Dino Marinčić, poglavlje „Svojstva vektora u Euklidskom prostoru” u završnom radu „Euklidski prostori”, Sveučilište J. J. Strossmayera u Osijeku, Odjel za matematiku, Osijek,

2015.

[14] Zrinka Franušić, Juraj Šiftar, poglavlje „Definicija i osnovna svojstva vektorskog prostora” u skripti „Linearna algebra 1”, skripta za nastavničke studije na PMF-MO, Prirodoslovno matematički fakultet – Matematički odsjek, Sveučilište u Zagrebu

[15] R. Manger, Sferni sustav i primjene, Građevinski fakultet, Sveučilište u Zagrebu, http://www.grad.unizg.hr/_download/repository/Sferni__sustav_i_primjene.pdf