SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 6647

MODEL DIJELA GRADA ZAGREBA TEMELJEN NA REALNIM PODACIMA

Luka Mesarić

Zagreb, lipanj 2020.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 6647

MODEL DIJELA GRADA ZAGREBA TEMELJEN NA REALNIM PODACIMA

Luka Mesarić

Zagreb, lipanj 2020.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 13. ožujka 2020.

ZAVRŠNI ZADATAK br. 6647

Pristupnik:	**Luka Mesarić (0036505985)**
Studij:	Računarstvo
Modul:	Računarska znanost
Mentor:	prof. dr. sc. Željka Mihajlović

Zadatak: **Model dijela grada Zagreba temeljen na realnim podacima**

Opis zadatka:

Proučiti izvore javno dostupnih prostornih kartografskih informacija vezanih uz područje grada Zagreba. Razraditi mogućnosti korištenja tih informacija u izradi trodimenzionalnog modela proizvoljnog dijela grada. Ostvariti prikaze dobivenih rezultata. Diskutirati utjecaj različitih parametara. Načiniti ocjenu rezultata i implementiranih algoritama. Izraditi odgovarajući programski proizvod. Koristiti grafički programski pogon Unreal Engine za prikaz rezultata. Rezultate rada načiniti dostupne putem Interneta. Radu priložiti algoritme, izvorne kodove i rezultate uz potrebna objašnjenja i dokumentaciju. Citirati korištenu literaturu i navesti dobivenu pomoć.

Rok za predaju rada: 12. lipnja 2020.

SADRŽAJ

1. Uvod

Izrada trodimenzionalnih modela gradova svoju primjenu nalaze u mnogim područjima ljudske djelatnosti. U eri svekolike digitalizacije i nagle urbanizacije mapiranje terena omogućuje učinkovito urbanističko planiranje s ciljem boljeg razumijevanja i upravljanja urbanim ekosustavima u smjeru razvoja pametnih gradova.

Inteligentna rješenja transportnih i evakuacijskih puteva u slučaju prirodnih katastrofa, organizacija cestovnog i zračnog prometa, raspoređivanje telekomunikacijskih odašiljača i slično samo su neka od područja primjene modela gradova. Također se koriste u vizualizaciji postupka kontrole prometa postavljanjem nadzornih kamerama, planiranju osvijetljenosti cesta kao i cijelih stambenih područja [1].

Ne manje važna je i mogućnost raspoređivanja zelenih i vodenih površina, prostorna analiza i simulacija insolacije radi optimalnog pozicioniranja solarnih kolektora, praćenje izmjene perioda zasjenjenosti i osunčanosti zgrada s ciljem upravljanja potrebama za energentima.

U seizmički aktivnim područjima važna je projekcija oštećenja zgrada od potresa kao i učinak rušenja na okolne objekte, a u regijama s jakim vjetrovima trodimenzionalni modeli od velike su pomoći u arhitekturi visokih zgrada.

U velikim i gusto naseljenim područjima od osobite je važnosti mogućnost praćenja urbanih zagađivača, primjerice isijavanje topline s pločnika i kolnika u međuprostorima zgrada, analiza količine urbane buke i vidljivosti neba između nebodera.

Vojni i sigurnosni segmenti također su zastupljeni u području primjene trodimenzionalnih modela pri čemu modele koriste za obuku kadra i simulaciju kretanja.

U multimediji trodimenzionalne simulacije gradova nude brojne mogućnosti u dizajnu video igara, filmskoj industriji i oglašavanju.

U turističkom sektoru i zaštiti kulturne baštine modeli gradova služe boljoj vizualnoj prepoznatljivosti lokacije i kreiranju virtualnih razgleda.

U ovom radu razmatra se postupak automatskog generiranja trodimenzionalnog modela proizvoljnog dijela grada Zagreba temeljenog na realnim i javno dostupnim podatcima.

Model se izrađuje u omjeru $1:1$ te se realistično prikazuje teren, zgrade i stabla. Područje za koje se planira generirati trodimenzionalni model zadaje se geografskim koordinatama donjeg lijevog i gornjeg desnog ugla. Posljedično, područje mora biti pravokutnog oblika.

Za izradu modela koristi se dodatak za alat *Blender* napisan u programskom jeziku *Python*.

2. Korištene tehnologije i alati

2.1. Blender

Konačno programsko rješenje razvijeno u ovom radu jest dodatak (engl. *add-on*) za *Blender*,[1] profesionalni programski alat otvorenog kôda namijenjen za 3D računalnu grafiku. Između ostalog, alat *Blender* koristi se za 3D modeliranje, animiranje objekata i stvaranje animiranih filmova, izradu vizualnih efekata, simuliranje fluida i čestičnih sustava te izrađivanje prikaza 3D scena (engl. *rendering*).

Za razvoj programskog rješenja korišten je *Blender* verzije 2.82 te je ista ili novija verzija potrebna za instalaciju i pokretanje dodatka.

2.2. Python

Programsko rješenje razvijeno je koristeći programski jezik *Python*.[2] *Blender* verzije 2.82 uključuje interpreter za *Python* verzije $3.7.4$, te je stoga programski kôd pisan u istoj verziji.

Programsko obavljanje operacija u *Blenderu* omogućeno je kroz *Python* API i posebne module, primarno `bpy`, `bmesh` i `mathutils`.[3]

2.3. Dodatne biblioteke

Osim biblioteka uključenih u standardnu instalaciju *Pythona*, prilikom razvoja programskog rješenja upotrijebljeno je nekoliko javno dostupnih biblioteka trećih strana.

Biblioteka *Requests: HTTP for Humans*™ služi vrlo jednostavnom slanju HTTP zahtjeva.[4] Za razvoj se koristi stabilna verzija $2.23.0$.

[1] `https://www.blender.org/`
[2] `https://www.python.org/`
[3] `https://docs.blender.org/api/current/`
[4] `https://requests.readthedocs.io/`

Biblioteka *Pillow* proširenje je biblioteke PIL (*Python Imaging Library*) te služi za obradu slika.[5] Prilikom razvoja korištena je verzija 7.1.2.

Biblioteka *pyproj* sučelje je za programski jezik *Python* prema biblioteci PROJ za kartografske projekcije i transformacije koordinata.[6] Korištena je verzija 2.6.1.

2.4. Unreal Engine

Unreal Engine grafički je programski pogon otvorenog kôda pisan u programskom jeziku *C++*.[7] Razvija ga tvrtka *Epic Games*. Za prikaz modela kojeg generira *Blender* dodatak koristi se *Unreal Engine* verzije 4.22.3.

Iako je programsko rješenje za izradu modela moglo biti izvedeno i kao dodatak za *Unreal Engine*, ipak je odabran *Blender* radi lakšeg razvoja. Naime, za razliku od programskog jezika *C++*, jezik *Python* pogodniji je za mnoge operacije koje su ključne pri izgradnji modela, poput mrežne komunikacije. S druge strane, program napisan u jeziku *C++* nedvojbeno bi se brže izvodio. Međutim, to nije presudno jer se model grada izrađuje samo jednom nakon čega se može neograničeno koristiti s obzirom na to da se zapravo radi samo o jednostavnom skupu objekata.

[5] `https://pillow.readthedocs.io/`
[6] `https://pyproj4.github.io/pyproj/`
[7] `https://www.unrealengine.com/`

3. Instalacija i korištenje razvijenog dodatka

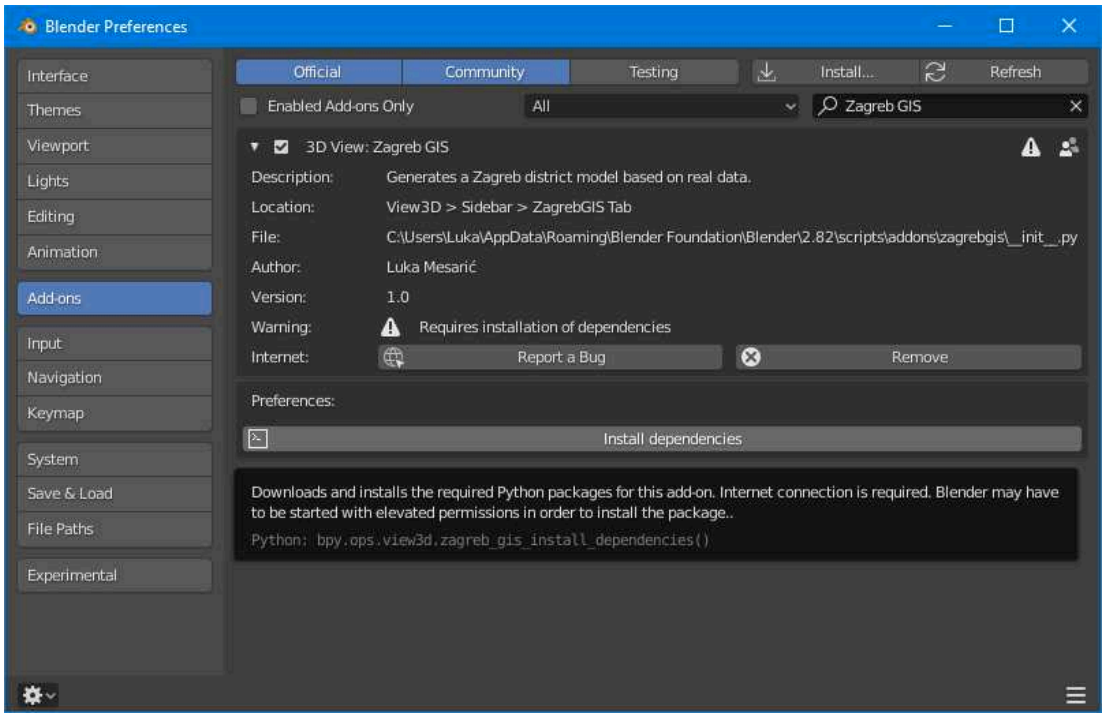
Razvijeno programsko rješenje prije korištenja potrebno je instalirati kao dodatak u alatu *Blender*. Sav izvorni kôd sprema se u ZIP arhivu iz koje *Blender* stvara i registrira novi dodatak.

U alatu *Blender* slijedom klikova na gumbe `Edit > Preferences > Add-ons > Install` otvara se preglednik datoteka na računalu. Nakon lociranja ZIP arhive odabire se "*Install Add-on*" i označuje kućica (engl. *checkbox*) pored imena dodatka u njegovim postavkama (slika 3.1).[8] Time je dodatak instaliran i registriran te postaje aktivan u grafičkom sučelju.

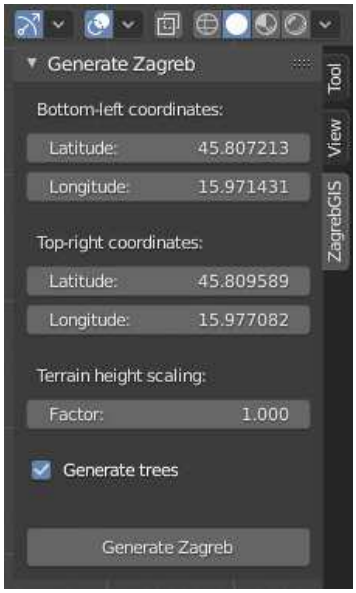
Blender koristi vlastitu instalaciju *Python* interpretera i pripadnih biblioteka te je u skriptama i dodatcima moguće koristiti jedino biblioteke koje se nalaze unutar te instalacije. Radi olakšanja korištenja dodatka, biblioteke opisane u poglavlju 2.3 o kojima dodatak ovisi automatski se preuzimaju i instaliraju unutar *Blenderove* instalacije *Pythona* pritiskom na gumb "*Install dependencies*" u postavkama dodataka (slika 3.1). Za razvoj te funkcionalnosti modificirano je postojeće programsko rješenje [2].

Jednom kada je instalacija dovršena dodatak je moguće koristiti pritiskom na tipku "`N`" na tipkovnici čime se otvara bočna izborna traka (engl. *sidebar*) u kojoj se nalazi nova kartica (engl. *tab*) pod imenom ZagrebGIS. Izgled grafičkog sučelja prikazan je na slikama 3.2a i 3.2b.

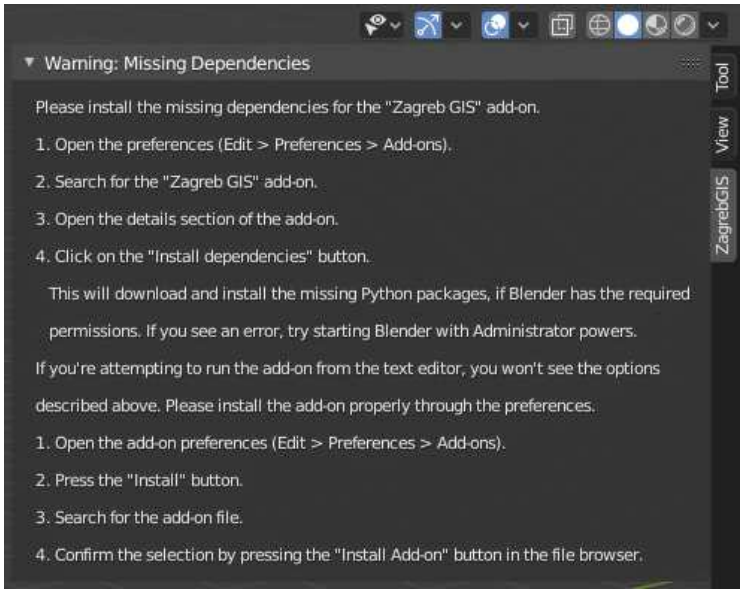
[8] `https://docs.blender.org/manual/en/latest/advanced/scripting/addon_tutorial.html#install-the-add-on`



Slika 3.1: Postavke instaliranog dodatka u alatu *Blender*



(a) Instalirane biblioteke



(b) Biblioteke nedostaju

Slika 3.2: Grafičko sučelje dodatka ZagrebGIS

4. Teren

4.1. Visinska mapa

Visinska mapa (engl. *heightmap*) posebna je vrsta slike koja se u 3D grafici koristi za pohranjivanje podataka o visini točaka u trodimenzionalnom prostoru [3].

Najjednostavnije visinske mape slike su koje sadrže samo jedan kanal boje koji predstavlja udaljenost pojedine točke od referentne razine, odnosno visinu točke. Pri tome se crna boja koristi za oznaku najniže, dok bijela boja predstavlja najvišu poziciju. U jednoj mapi nije nužno koristiti cijeli raspon boja. Primjer visinske mape koja pokriva područje Zagreba veličine približno $20\,\mathrm{km} \times 20\,\mathrm{km}$ nalazi se na slici 4.1.

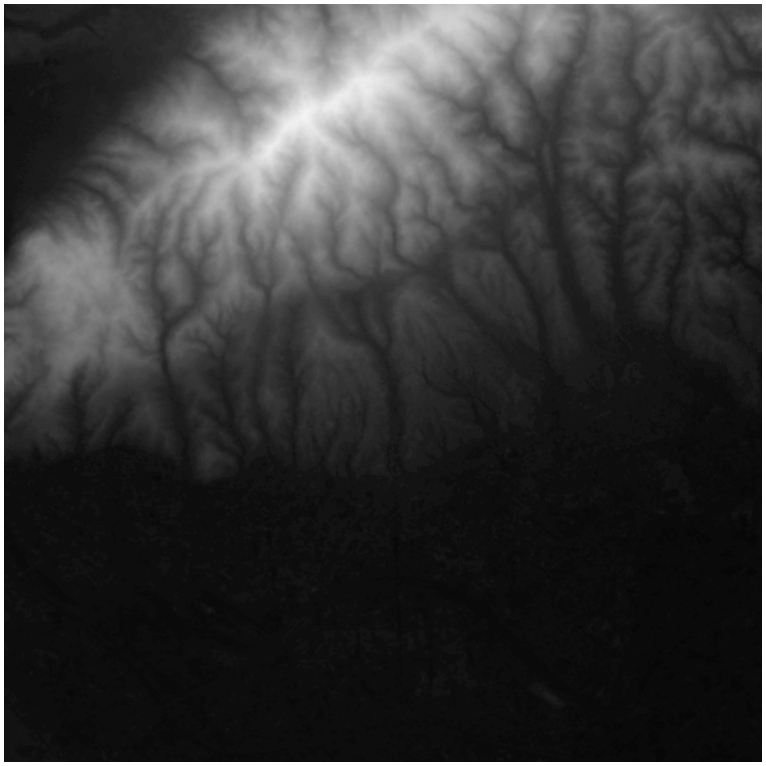
Točnost podataka ovisi o broju bitova po kanalu slike. Ako se koristi samo jedan 8-bitni kanal, moguće je prikazati svega 256 različitih visina zbog čega se na terenu mogu vidjeti oštre granice poput stepenica. S druge strane, u slučaju jednog 16-bitnog kanala, broj razina povećava se na čak $65\,536$. Na primjer, za raspon visine od $1024\,\mathrm{m}$, koji ugrubo odgovara području Zagreba i Medvednice, sa 16 bitova po kanalu moguće je prikazati 64 visinske razine po jednom metru, odnosno jednu razinu svakih $1.56\,\mathrm{cm}$.

4.2. Mapiranje Zemljine površine

SRTM (*Shuttle Radar Topography Mission*) internacionalni je istraživački projekt koji je rezultirao digitalnim modelom elevacije Zemljine površine od $60°$ sjeverne do $56°$ južne geografske širine izrađenim na temelju radarskih snimaka iz *Space Shuttlea Endeavour*. Rezolucija mapa jest $30\,\mathrm{m}$ nad područjem Sjedinjenih Američkih Država te $90\,\mathrm{m}$ nad ostalim dijelovima [11].

Japanski senzor ASTER (*Advanced Spaceborne Thermal Emission and Reflection Radiometer*) korišten je u najnovijem NASA-inom globalnom istraživanju i mapiranju visinskih podataka o površini Zemlje.[9] Satelitskim snimanjem pokriveno je područje

[9] `https://asterweb.jpl.nasa.gov/gdem.asp`



Slika 4.1: Visinska mapa Zagreba i dijela Medvednice preuzeta s *terrain.party*

od $83°$ sjeverne do $83°$ južne geografske širine, odnosno $99\,\%$ površine kopna, s rezolucijom od $30\,\mathrm{m}$ nad cijelim područjem [9]. Glavni je nedostatak to što je instrument sklon pogreškama u slučaju visoke koncentracije oblaka ili naglih promjena u visini čime nastaju rupe u podatcima koje se često moraju ručno popravljati.

4.3. Visinski podatci za područje Zagreba

Za potrebe ovog rada visinske mape preuzimaju se s besplatne web-aplikacije *terrain.party*[10] koja je originalno razvijena za potrebe simulacijske igre *Cities: Skylines*.[11]

Visinske mape koje aplikacija generira crno-bijele su slike rezolucije 1081×1081 piksela u formatu PNG (*Portable Network Graphics*) s jednim 16-bitnim kanalom. Dok se za područje Sjedinjenih Američkih Država i Danske mogu preuzeti posebne visinske mape rezolucije $10\,\mathrm{m}$ odnosno čak $1.6\,\mathrm{m}$, područje Zagreba pokriveno je globalnim podatcima ASTER (rezolucije $30\,\mathrm{m}$) i SRTM3 v4.1 (rezolucije $90\,\mathrm{m}$). Osim ASTER i SRTM3 mapa generira se i spojena mapa koja većinom koristi ASTER podatke, a eventualne nepotpune informacije nadopunjuje iz ostalih izvora. Upravo je ta mapa

[10] `https://terrain.party/`

[11] `https://www.citiesskylines.com/`

korištena za automatsku izradu terena u ovom radu.

4.4. Aproksimacija zakrivljene površine

Radi jednostavnosti, prilikom izrade modela zakrivljena Zemljina površina iz opravdanih se razloga aproksimira ravnom plohom u obliku pravokutnika.

Naime, na $5\,\mathrm{km}$ zračne udaljenosti zakrivljenost Zemlje iznosi točno $2\,\mathrm{m}$, dok je na $1\,\mathrm{km}$ zakrivljenost svega $8\,\mathrm{cm}$. Doda li se tome reljefnost terena jasno je da je greška koja nastaje zbog aproksimacije ravnom plohom efektivno neprimjetna.

S druge strane, a ponovno zbog zakrivljenosti Zemlje, područje koje se zadaje preko geografskih koordinata donjeg lijevog i gornjeg desnog kuta oblikom je sličnije jednakokračnom trapezu nego pravokutniku. Greška koja nastaje zbog aproksimacije pravokutnikom ovisi o geografskoj širini. Ako se na području Zagreba promatra područje dužine i širine približno $4\,\mathrm{km}$, razlika duljine osnovica trapeza iznosi otprilike $3.5\,\mathrm{m}$, odnosno manje od $0.1\,\%$ što se može smatrati zanemarivim.

4.5. Generiranje 3D modela terena

Blender omogućuje jednostavno stvaranje terena iz visinske mape uporabom modifikatora za premještanje vrhova (engl. *displace modifier*).

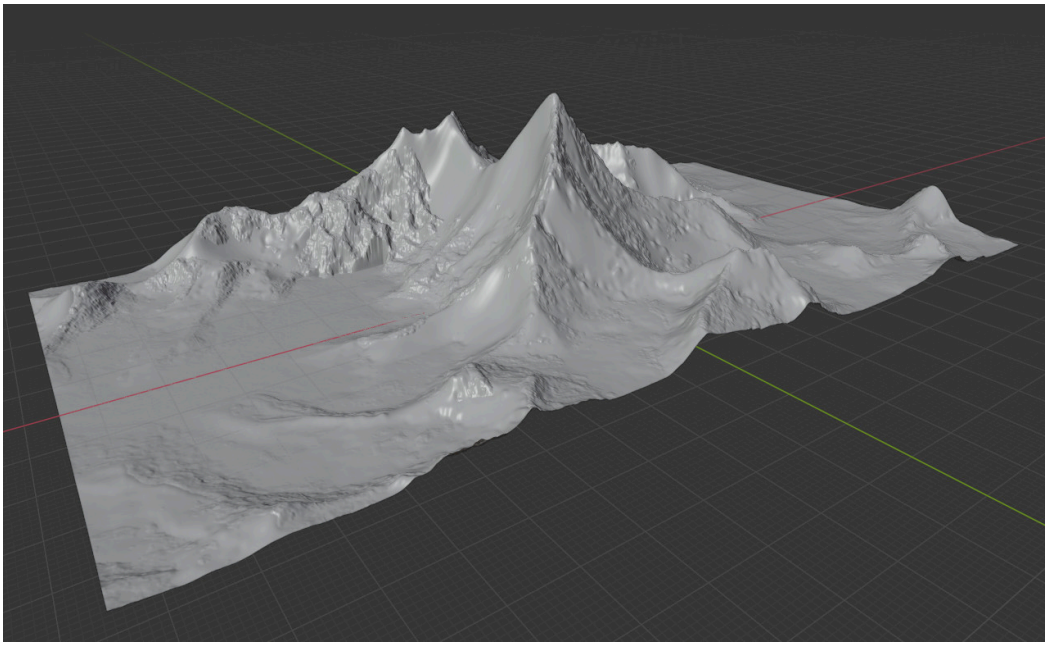
Kao osnova za izradu modela terena koristi se objekt tipa rešetke (engl. *grid*) jednakih dimenzija kao teren koji se modelira. Gustoća rešetke povećava se razdjeljivanjem (engl. *subdivision*) tako da je konačni objekt pravilna mreža vrhova međusobno udaljenih četiri metra. Prilikom razvoja uočeno je da razmak od upravo četiri metra pruža balans između zadržavanja kvalitete terena i želje za smanjenjem broja vrhova i trokuta u sceni. Kada bi se primjerice koristio razmak od dva metra, ukupan broj vrhova i trokuta potrebnih za modeliranje terena porastao bi četiri puta zbog kvadratne ovisnosti, što ima značajan negativan utjecaj na performanse, dok je vizualni efekt zaglađenja nezamjetno poboljšan.

U slučaju da je teren pravokutnog umjesto kvadratnog oblika, programski se mijenjaju dimenzije slike koja predstavlja visinsku mapu. Ona se rasteže tako da omjer širine i visine slike odgovara omjeru dimenzija terena. Za određivanje vrijednosti novonastalih slikovnih elemenata koristi se bikubična interpolacija.

Za ispravno modeliranje terena, osim visinske mape, potrebna je i informacija o apsolutnoj razlici visine najniže i najviše točke terena. Uz visinsku mapu, *terrain.party*

vraća i taj brojevni podatak te se on koristi za podešavanje jačine (engl. *strength*) modifikatora za premještanje kako bi maksimalna razlika vrijednosti z-koordinata vrhova bila jednaka maksimalnoj razlici elevacije stvarnog terena. Pri tome se na visinskim mapama manjih područja može javiti šum uslijed krivog očitanja stvarne razlike u elevaciji čime generirani teren postaje značajno grbaviji nego što u stvarnosti jest. Zbog toga je u grafičkom sučelju dodatka omogućeno podešavanje faktora skaliranja visine terena kojim korisnik po volji povećava ili smanjuje maksimalnu razliku elevacije.

Na slici 4.2 prikazan je model planinskog vrha Mount Everest izrađen koristeći visinsku mapu koju generira *terrain.party*, s faktorom skaliranja postavljenim na 1 i u omjeru $1 : 1$. Maksimalna razlika elevacije u prikazanom modelu približno iznosi $3500\,\mathrm{m}$. Model se sastoji od otprilike $12\,500\,000$ vrhova.



Slika 4.2: Model planinskog vrha Mount Everest u omjeru $1 : 1$

4.6. Određivanje visine terena u proizvoljnoj točki

Za potrebe ispravnog postavljanja objekata na teren nužno je poznavanje visine modela terena u bilo kojoj točki. Problem je moguće formalizirati kao traženje vrijednosti funkcije $f : \mathbb{R}^2 \to \mathbb{R}$ koja iz para koordinata (x, y) određuje vrijednost z koordinate.

Blender za to ne nudi gotovo rješenje. Međutim, moguće je dohvatiti globalne koordinate svih vrhova u modelu. S obzirom na to da je približno rješenje dovoljno dobro, problem se može modificirati tako da se traže vrhovi mreže (engl. *mesh*) koji su,

gledajući jedino projekciju na xOy ravninu, *blizu* tražene točke. Vrijednosti njihovih z koordinata tada se mogu koristiti za predviđanje vrijednosti z koordinate u točki (x, y). U ovom slučaju *bliskim* vrhovima smatraju se vrhovi iz skupa V' definiranog formulom 4.1, pri čemu V označava skup svih vrhova modela.

$$V' = \{\, v \in V \mid |v.x - x| < d \wedge |v.y - y| < d \,\} \tag{4.1}$$

Testiranjem različitih vrijednosti parametra d odlučeno je postaviti ga na $70\,\%$ iznosa udaljenosti između susjednih vrhova, odnosno vrijedi $d = 2.8\,\mathrm{m}$.

Vremenski efikasan pronalazak elemenata skupa V' postiže se modificiranim dvodimenzionalnim binarnim pretraživanjem. Obavlja se pretprocesiranje podataka u kojem se svi vrhovi grupiraju i sortiraju prema vrijednostima x koordinate. Vrhovi u svakoj tako nastaloj grupi potom se sortiraju prema vrijednosti y koordinate. Time je omogućeno binarno pretraživanje prvo po x koordinati, a zatim unutar grupe po y koordinati. Od standardnog binarnog pretraživanja ovo pretraživanje razlikuje se po tome što u prvom koraku može pronaći više grupa, odnosno u drugom koraku više vrhova. S prethodno definiranom vrijednošću parametra d, algoritam može vratiti 0, 1, 2 ili 4 različita vrha.

Za mrežu od $n \times m$ vrhova, dvodimenzionalno binarno pretraživanje, nakon pretprocesiranja, za svaki upit ima složenost $O(\log n + \log m) = O(\log(nm))$, dok trivijalno slijedno pretraživanje cijelog skupa V ima složenost $O(nm)$.

Alternativno rješenje jest indeksiranje vrhova u modelu terena uporabom strukture podataka *R-stablo* koje se često primjenjuje u geografskim informacijskim sustavima (GIS) [10]. Primjeri *Python* biblioteka trećih strana koje implementiraju R-stablo su *Rtree*[12] i *GeoPandas*[13].

[12] `https://rtree.readthedocs.io/`
[13] `https://geopandas.org/`

5. Građevine

5.1. Izvor podataka

OpenStreetMap[14] (dalje u tekstu: OSM) pokazao se najboljim izvorom podataka o građevinama za potrebe ovog rada.

Iako za područje grada Zagreba postoje značajno detaljniji podatci od onih koje pruža OSM, poput podataka nastalih višegodišnjim aerofotogrametrijskim i LiDAR snimanjima grada,[15] dohvaćanje tih podataka nije jednostavno jer se korisnicima ne nudi nikakvo sučelje (engl. *Application programming interface – API*).

S druge strane, uz OSM postoji *Overpass* API i njemu pripadni jezik za pisanje strukturiranih upita (engl. *Overpass Query Language*) koji korisnicima omogućuje gotovo trivijalno pretraživanje i dohvaćanje podataka dostupnih u OSM bazi [4][5]. Za *Overpass* API razvijeno je i grafičko web sučelje pod imenom *Overpass Turbo* [6].[16]

5.2. Struktura *OpenStreetMap* podataka

Podatci koje *Overpass* API vraća sastoje se od popisa čvorova (engl. *node*), puteva (engl. *way*) i relacija (engl. *relation*) u JSON (*JavaScript Object Notation*)[17] formatu. Svaki element ima jedinstveni cjelobrojni identifikator.

Čvorovi imaju podatke o svojoj geografskoj širini i dužini. Putevi su definirani listom čvorova. Zatvorene petlje posebna su vrsta puteva kojima su prvi i zadnji čvor jednaki te se u pravilu koriste za opisivanje poligona. Relacije se sastoje od više puteva koji su najčešće ujedno i zatvorene petlje.

Svi elementi mogu imati dodatne metapodatke (oznake).[18] Za potrebe dohvaćanja i stvaranja zgrada interesantni su metapodatci koji razlikuju građevine od ostalih OSM

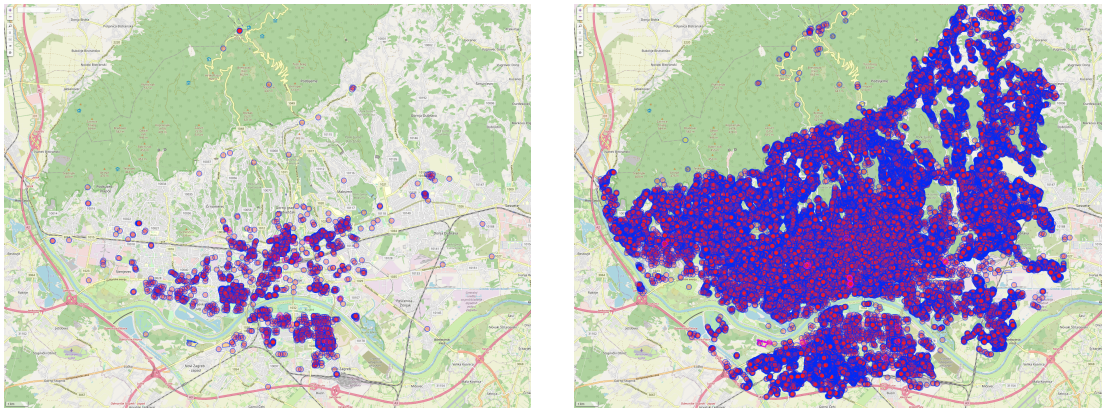
[14] https://www.openstreetmap.org/
[15] https://zagreb.gdi.net/zg3d/
[16] https://overpass-turbo.eu/
[17] https://www.json.org/
[18] https://wiki.openstreetmap.org/wiki/Tags

podataka, kao i metapodatci o visini pojedine građevine.

U ovom radu zgradama se smatraju svi putevi i relacije koji imaju postavljenu bilo kakvu vrijednost oznaka `building`[19] ili `building:part`[20] bez njihove daljnje analize [7], te se upravo takav upit šalje prema OSM bazi koristeći *Overpass* API. Razlog tome jest što na globalnoj razini više od $81\,\%$ građevina ima jednostavnu oznaku `building=yes`,[21] dok više od $86\,\%$ zgrada ima oznaku `building:part=yes`,[22] bez dodatnog specificiranja tipova zgrada.

Podatci o visini pojedine zgrade dostupni su kroz nekoliko različitih oznaka. Standardne su oznake `height`[23] i `building:height` kojima se obilježava ukupna visina u metrima. Posredni način određivanja visine moguć je koristeći podatak o broju etaža zgrade, zapisan u oznakama `levels`, `building:levels`[24] i `roof:levels`.

Bilo kakvi podatci o visini zgrada u Zagrebu zapravo su vrlo rijetki. Od približno $83\,000$ zgrada koje postoje u OSM bazi, njih svega $3.74\,\%$ ima podatke o točnoj visini ili broju etaža. Taj odnos vizualiziran je slikom 5.1. Za sve ostale zgrade visina se odabire kao slučajan broj iz uniformne distribucije na intervalu $[8, 25]$.



(a) Sve građevine u Zagrebu za koje postoje podatci o visini

(b) Sve građevine u Zagrebu, bez postavljenih uvjeta

Slika 5.1: Vizualizacija OSM podataka o građevinama u Zagrebu (web sučelje *Overpass Turbo*)

[19] `https://wiki.openstreetmap.org/wiki/Key:building`
[20] `https://wiki.openstreetmap.org/wiki/Key:building:part`
[21] `https://taginfo.openstreetmap.org/keys/building#values`
[22] `https://taginfo.openstreetmap.org/keys/building:part#values`
[23] `https://wiki.openstreetmap.org/wiki/Key:height`
[24] `https://wiki.openstreetmap.org/wiki/Key:building:levels`

5.3. Stvaranje 3D modela jednostavnih zgrada

Jedine informacije na temelju kojih se stvara model zgrade su oblik baze (engl. *footprint*) i visina zgrade. Modeli jednostavnih zgrada, čija se baza sastoji samo od jedne zatvorene petlje, imaju ravne krovove i oblika su prizme.

Za stvaranje mreže (engl. *mesh*) objekta *Blender* treba dva podatka: listu koordinata vrhova (engl. *vertex*) i listu stranica (engl. *faces*). Pri tome je svaka stranica predstavljena listom indeksa vrhova.

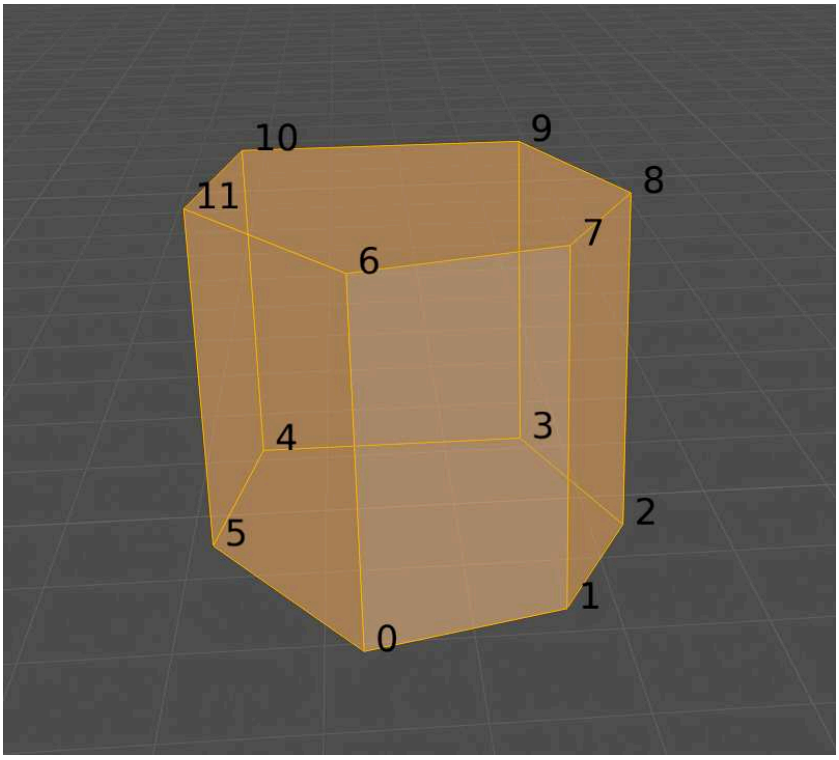
Algoritam stvaranja 3D modela zgrade započinje stvaranjem kopije svih n vrhova baze te njihovom translacijom u smjeru pozitivne z-osi za iznos visine zgrade čime se dobiva konačna lista od $2n$ vrhova. Na slici 5.2 prikazan je model s bazom od $n = 6$ vrhova.

Jednostavnim algoritmom, čiji se programski kôd nalazi u isječku 5.1, stvaraju se stranice objekta. Prvo se u petlji stvara n bočnih stranica, od kojih svaka ima po dva vrha iz donje baze i dva vrha iz gornje baze. Naknadno se dodaju same baze: prvo donja baza s indeksima vrhova u padajućem poretku, a nakon toga gornja baza s indeksima u rastućem poretku. Time je stvorena lista svih $n + 2$ stranica potrebnih za stvaranje prizme.

Algoritam 5.1: Isječak programskog kôda za stvaranje stranica 3D modela zgrade

```
1 def _generate_faces_indices(n: int) -> List[List[int]]:
2     faces = [[i, (i + 1) % n, n + (i + 1) % n, n + i]
3              for i in range(n)]
4     faces.append(list(range(n - 1, -1, -1)))
5     faces.append(list(range(n, 2 * n)))
6     return faces
```

Ovisno o tome jesu li vrhovi baze zadani u smjeru kazaljke na satu (engl. *clockwise – CW*) ili suprotno od smjera kazaljke na satu (engl. *counterclockwise – CCW*), normale svih stranica bit će usmjerene ili prema unutrašnjosti modela ili prema van. Radi poštivanja konvencija nad svakim se generiranim objektom poziva naknadno izračunavanje normala (engl. *recalculate normals*) ugrađeno u *Blender* tako da su u konačnici sve normale ujednačene i pokazuju iz objekta prema van bez obzira na originalan poredak točaka u bazi.



Slika 5.2: Primjer objekta stvorenog iz točaka baze algoritmom 5.1. Vrhovi su obilježeni svojim indeksima.

5.4. Stvaranje 3D modela složenih zgrada

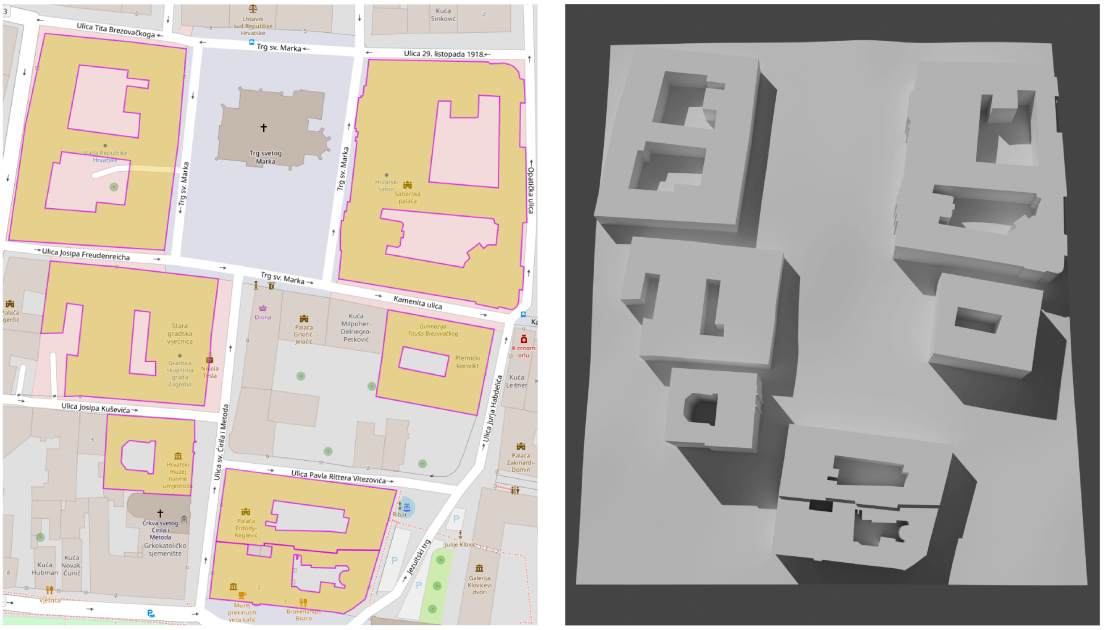
Složenim se zgradama obris sastoji od više poligona (engl. *relation*).

Zgrade sastavljene od više susjednih dijelova koji se međusobno ne preklapaju trivijalno se svode na modeliranje jednostavnih zgrada koje je opisano u prethodnom poglavlju – za svaki dio zgrade (engl. *way*) u relaciji stvara se po jedna samostalna jednostavna zgrada.

Zgrade čiji je obris u obliku poligona koji u unutrašnjosti ima jednu ili više praznina značajno su teže za modeliranje. Njihov model nije prizma u užem smislu riječi. Stvaranje takvih objekata obavljeno je konstruktivnom geometrijom čvrstih tijela (engl. *Constructive Solid Geometry – CSG*) [8]. Prvo se stvara jednostavna zgrada koja je opisana vanjskim poligonom. Za svaki unutrašnji poligon koji opisuje jednu prazninu također se stvara po jedna jednostavna zgrada, ali takve su zgrade privremene. Koristeći *Blenderov* ugrađeni modifikator za CSG (engl. *boolean modifier*) sve privremene zgrade oduzimaju se od početne te se potom svi privremeni objekti uklanjaju iz scene. Time se dobiva konačni model zgrade kao jedna mreža (engl. *mesh*).

Primjer na slici 5.3 prikazuje složene zgrade u dijelu Gornjeg Grada oko Trga svetoga Marka. U gornjem lijevom kutu vidljivi su Banski dvori, dok se u gornjem des-

nom kutu nalazi Saborska palača. U generiranom modelu prikazanom na slici 5.3b naknadno su ručno uklonjene sve jednostavne zgrade.



(a) Slika iz web sučelja *Overpass Turbo* **(b)** Generirani model dijela grada

Slika 5.3: Složene zgrade na Gornjem Gradu (Trg svetoga Marka)

5.5. Uređivanje javno dostupnih podataka

Podatke koji se pohranjuju u OSM bazi prikupljaju dobrovoljci širom svijeta. Bilo tko može kreirati korisnički račun te doprinijeti dodavanjem novih ili ispravljanjem postojećih podataka. U svibnju 2020. godine OSM je imao više od 6.5 milijuna registriranih korisnika.[25]

Na početku izrade ovog rada o zgradama Fakulteta elektrotehnike i računarstva u OSM bazi je, osim obrisa zgrada, postojalo vrlo malo dodatnih informacija. Za potrebe ovog rada podatci su dopunjeni:

- visina zgrade C postavljena je na 50 m (oznaka `height`) [12],
- broj etaža zgrade C postavljen je na 13 (oznaka `building:levels`),
- broj etaža zgrade B postavljen je na 2,
- broj etaža zgrade A postavljen je na 4,

[25] `https://web.archive.org/web/20200524000234/https://www.openstreetmap.org/stats/data_stats.html`

– broj etaža zgrade D postavljen je na 4,

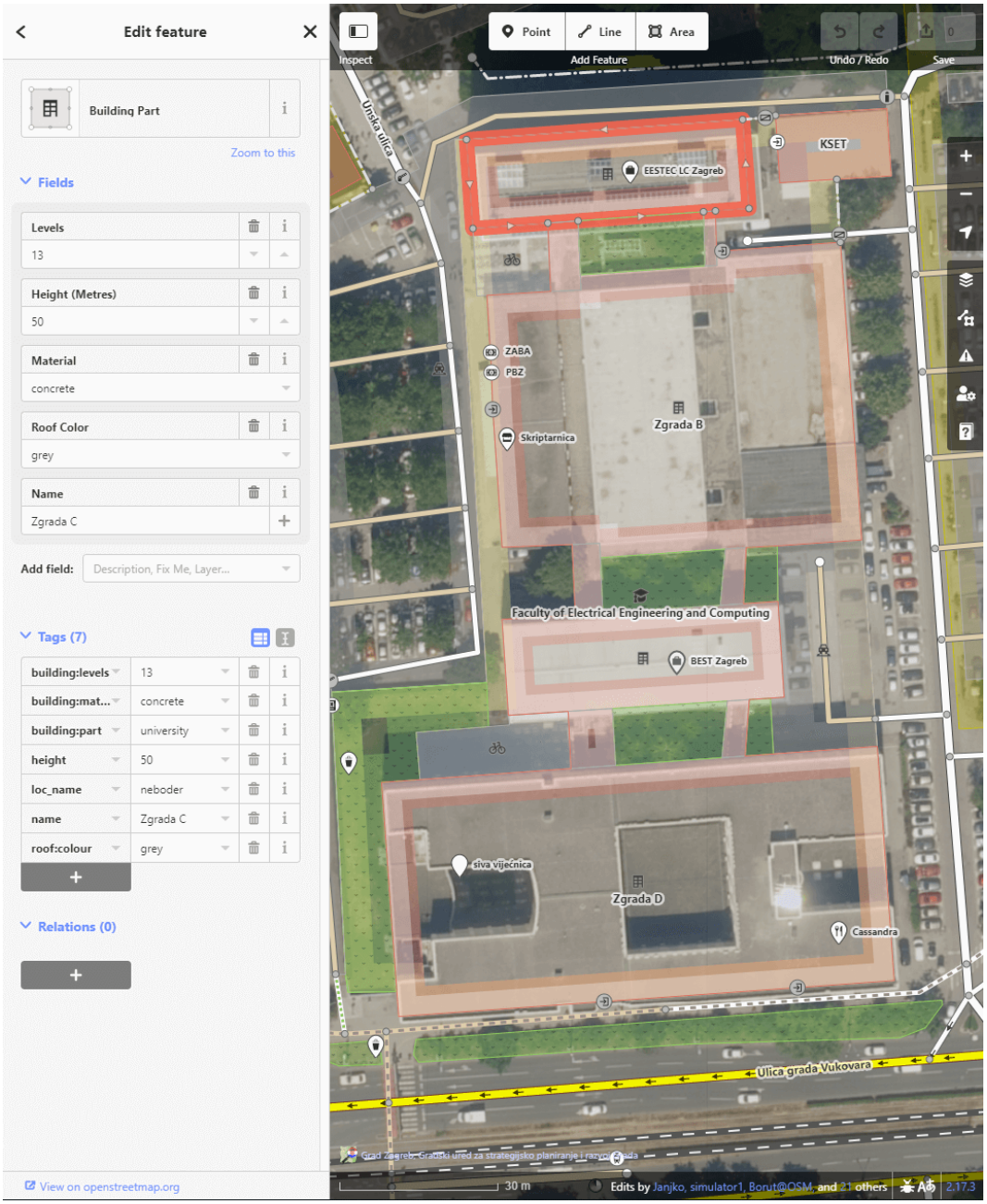
– visina svih šest hodnika koji spajaju zgrade postavljena je na $4\,\mathrm{m}$,

– broj etaža svih hodnika postavljen je na 1,

– označeno je da su svi hodnici izrađeni od stakla (oznaka `building:material`),

– svim zgradama i hodnicima oznaka `building:part` postavljena je na `university` (prethodne vrijednosti bile su `yes` ili `school`).

Osim kroz specijalizirane programe, ažuriranje podataka može se obaviti i u web-pregledniku po izboru. Primjer uređivanja podataka vezanih uz zgradu C prikazan je na slici 5.4.



Slika 5.4: Uređivanje OSM podataka o FER-ovoj zgradi C

6. Stabla

6.1. Izvor podataka

Najdetaljniji izvor podataka o stablima na području Zagreba jest Katastar zelenila[26] Zrinjevca[27]. Katastar sadrži lokaciju i druge informacije o svim stablima za čije je održavanje zadužen Zrinjevac te omogućuje jednostavan dohvat podataka na odabranom području u JSON formatu.

Za razliku od koordinatnog sustava EPSG:4326[28] koji se koristi u ostatku programskog rješenja i koji točke identificira preko geografske širine i dužine, Katastar sve koordinate vraća u projekcijskom koordinatnom sustavu EPSG:3857 (WGS 84 / Pseudo-Mercator)[29]. Stoga se sve dohvaćene koordinate prvo moraju transformirati u EPSG:4326 sustav za što se koristi biblioteka *pyproj* odnosno njezin razred `Transformer`.

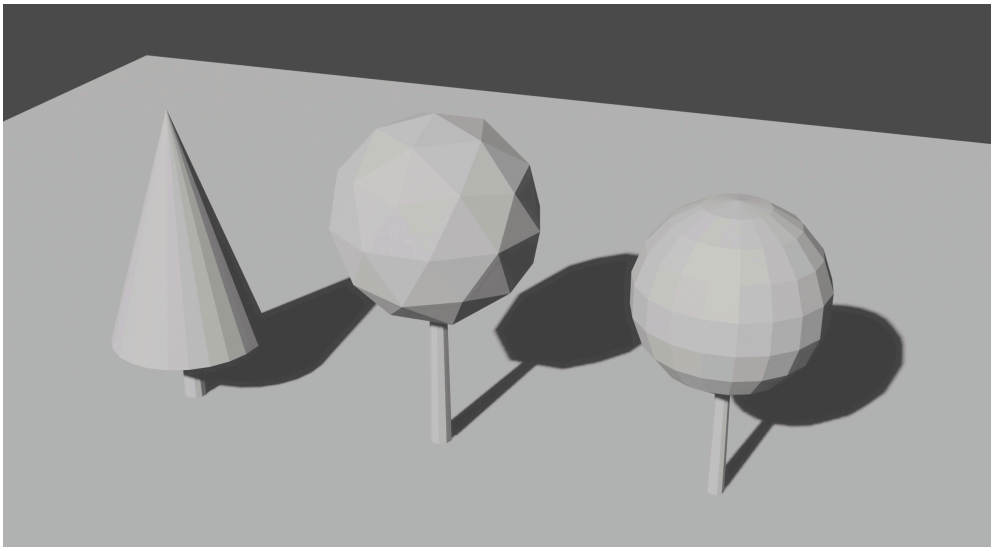
6.2. Modeli stabala

U generiranom modelu grada koriste se tri vrste stabala čiji su primjerci prikazani na slici 6.1. Vrsta stabla na pojedinoj lokaciji bira se nasumično.

Deblo svakog stabla predstavljeno je krnjim stošcem. Krošnje su predstavljene s tri različita objekta:

- stožac s varijabilnim brojem vrhova baze, polumjerom baze i visinom,
- geodetska sfera (engl. *Ico Sphere*)[30] s varijabilnim polumjerom,
- UV sfera (engl. *UV Sphere*)[31] s varijabilnim brojem prstena, segmenata i polumjerom.

[26] `https://gis.zrinjevac.hr/`
[27] Podružnica tvrtke Zagrebački Holding d.o.o.
[28] `https://epsg.io/4326`
[29] `https://epsg.io/3857`
[30] Sfera sastavljena od trokuta.
[31] Sfera sastavljena od četverokuta.



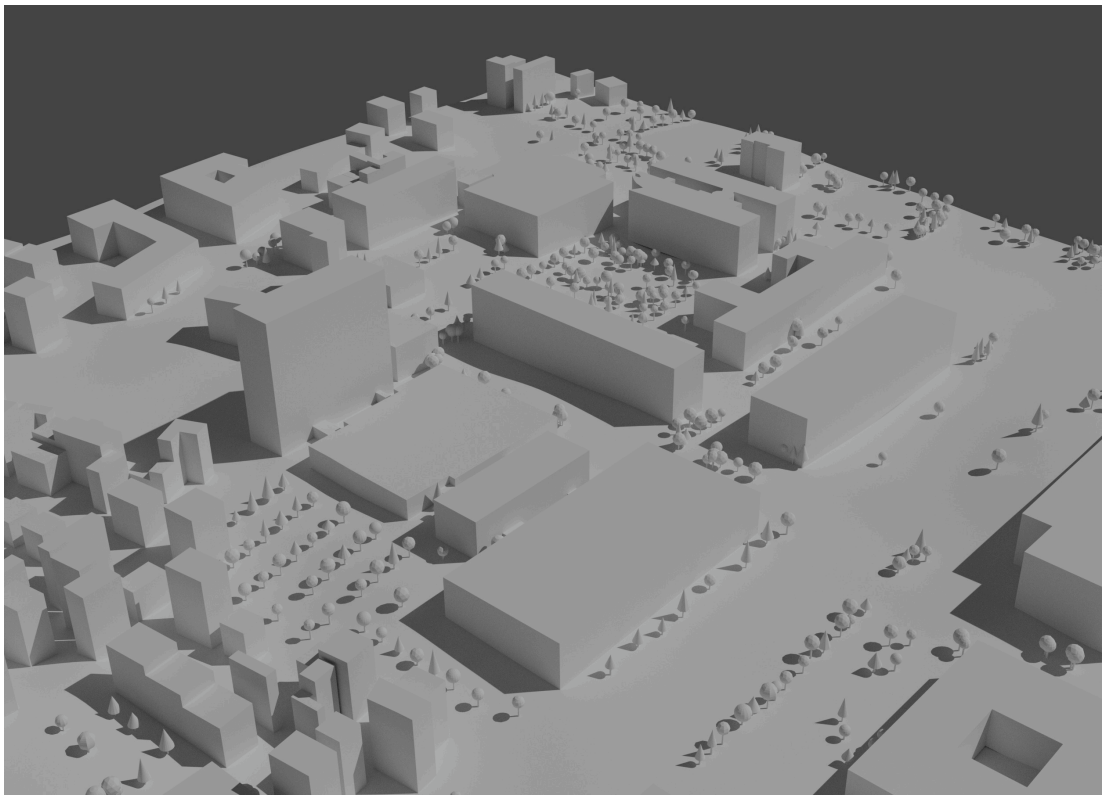
Slika 6.1: Tri vrste stabala koja se koriste u modelu grada

Svi opisani parametri slučajno se biraju iz uniformnih distribucija. Radi smanjenja broja vrhova i poligona u modelu grada koji može imati i desetak tisuća stabala većinom se koriste stošci i sfere s malim brojem vrhova.

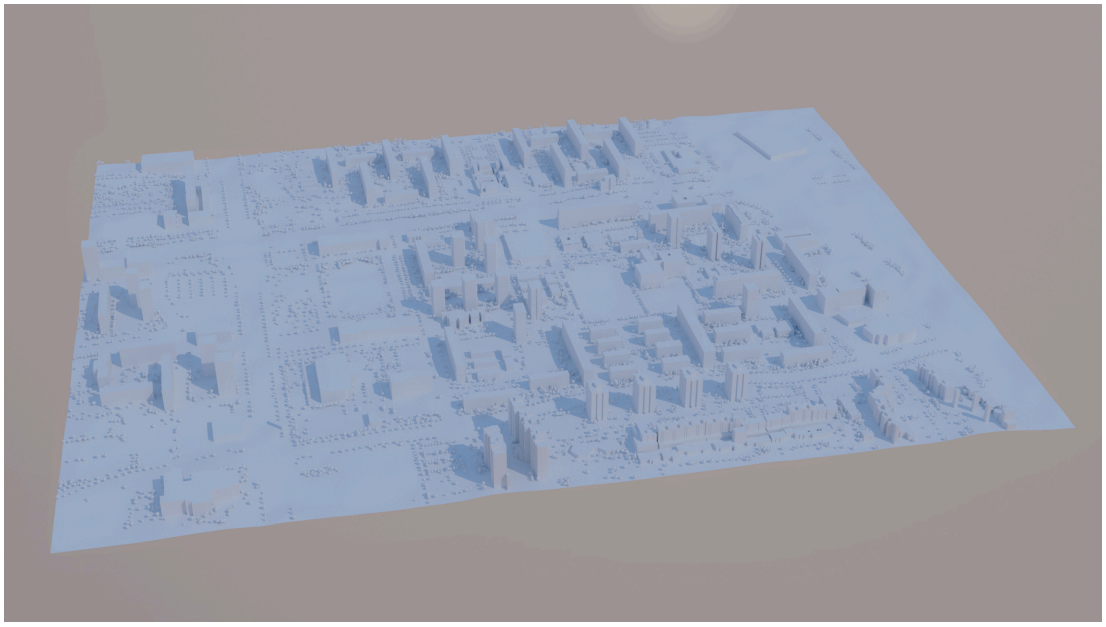
7. Rezultati

Razvijeni *Blender* dodatak kao rezultat izvođenja u scenu dodaje sve generirane objekte – teren, zgrade i drveće. Oni se dalje mogu po potrebi ručno uređivati. Stvoreni su objekti jednostavni i sastoje se samo od mreže vrhova bez tekstura ili drugih posebnih podataka.

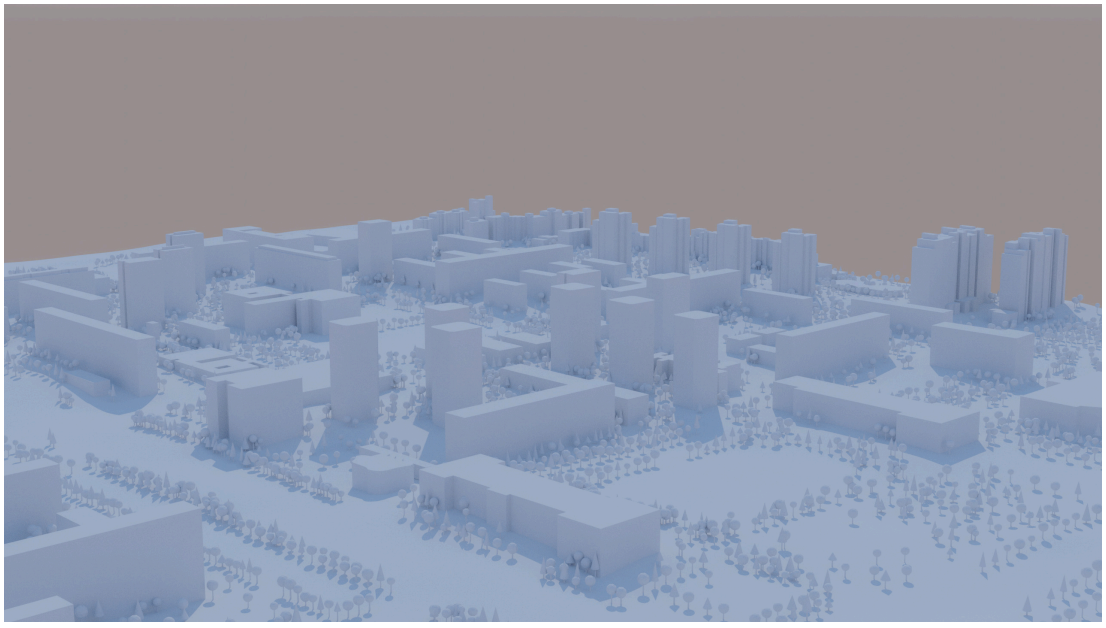
Na slici 7.1 vidljiv je izrađeni prikaz (engl. *render*) zgrada Fakulteta elektrotehnike i računarstva kao i sportske dvorane Martinovka i okolnih zgrada. Nakon uređivanja OSM podataka sve zgrade i hodnici ispravnih su dimenzija.



Slika 7.1: Model zgrada Fakulteta elektrotehnike i računarstva



(a) Cijela četvrt Utrina i dio okolice



(b) Detalj četvrti Utrina

Slika 7.2: Izrađen prikaz generiranog modela gradske četvrti Utrina

Model gradske četvrti Utrina prikazan na slici 7.2 pokriva površinu od $1.42\,\mathrm{km}^2$. Generiranje je trajalo približno 20 minuta, od čega je $90\,\%$ vremena utrošeno na stvaranje stabala. Ukupno je stvoreno približno 320 zgrada i 7100 stabala. Cijeli model ima približno $1\,455\,000$ trokuta i $757\,000$ vrhova. Na slici 7.2b uočljivo je da su gotovo sve zgrade i neboderi ispravne visine. Takav ishod mogao se i predvidjeti s obzirom na to da je četvrt odlično pokrivena podatcima o visinama zgrada (slika 5.1a).

Generirani model dijela grada može se izvesti (engl. *export*) iz *Blendera* te primjerice uvesti (engl. *import*) u *Unreal Engine*.[32] Upravo je to učinjeno i prikazano na slici 7.3. S lijeve strane može se iz trećeg lica vidjeti model igrača u skoku, dok je s desne strane u pozadini vidljiva Zagrebačka katedrala.

Slika 7.3: Scena iz grafičkog pogona *Unreal Engine*

[32] `https://docs.unrealengine.com/en-US/Engine/Content/Importing/FBX/FullScene/index.html`

8. Zaključak

Rezultat rada jest programsko rješenje koje omogućuje automatiziranu izradu osnovnog modela proizvoljnog dijela grada Zagreba. Model u omjeru $1:1$ vjerno prikazuje odabrani dio grada.

Očekivano, ostaje mnogo prostora za poboljšanje. Osim stabala, Zrinjevac u svojem Katastru zelenila nudi podatke o grmlju koje održava, kao i urbanoj opremi poput klupa, sprava na dječjim igralištima te koševa za smeće. OSM ima podatke o travnjacima, livadama i šumama koji se mogu iskoristiti za točniju reprezentaciju zelenih površina. Također, kvaliteta generiranog modela grada značajno bi se poboljšala dodavanjem tekstura, primjerice projiciranjem satelitske snimke na model terena ili teksturiranjem pročelja zgrada. Bilo bi poželjno i da krovovi svih zgrada nisu potpuno ravni.

Navedena poboljšanja doprinijela bi boljoj predočivosti i vizualnoj atraktivnosti stvorenog modela. Time bi bilo omogućeno učinkovitije urbanističko planiranje pojedinih gradskih četvrti.

Programski kôd u cijelosti je javno dostupan na *GitHub* repozitoriju:
`https://github.com/LMesaric/BSc-Thesis-FER-2020`

LITERATURA

[1] Filip Biljecki, Jantien Stoter, Hugo Ledoux, Sisi Zlatanova, i Arzu Çöltekin. Applications of 3D City Models: State of the Art Review. *ISPRS International Journal of Geo-Information*, 4(4):2842–2889, 2015. URL https://doi.org/10.3390/ijgi4042842.

[2] Robert Gützkow. Example scripts and add-ons for Blender: Install dependencies, Svibanj 2020. URL https://github.com/robertguetzkow/blender-python-examples/tree/master/add-ons/install-dependencies.

[3] Danijel Janković. *Reljefne tehnike teksturiranja u prikazu terena*. Diplomski rad, Fakultet elektrotehnike i računarstva Sveučilišta u Zagrebu, Lipanj 2010. URL http://www.zemris.fer.hr/predmeti/ra/Magisterij/10_Jankovic/diplomski-Jankovic.pdf.

[4] OpenStreetMap Wiki. Overpass API — OpenStreetMap Wiki. https://wiki.openstreetmap.org/w/index.php?title=Overpass_API&oldid=1996368, 2020. Pristupljeno: 4. lipnja 2020.

[5] OpenStreetMap Wiki. Overpass API/Overpass QL — OpenStreetMap Wiki. https://wiki.openstreetmap.org/w/index.php?title=Overpass_API/Overpass_QL&oldid=1997069, 2020. Pristupljeno: 4. lipnja 2020.

[6] OpenStreetMap Wiki. Overpass turbo — OpenStreetMap Wiki. https://wiki.openstreetmap.org/w/index.php?title=Overpass_turbo&oldid=1958362, 2020. Pristupljeno: 4. lipnja 2020.

[7] OpenStreetMap Wiki. Simple 3D buildings — OpenStreetMap Wiki. https://wiki.openstreetmap.org/w/index.php?title=

Simple_3D_buildings&oldid=1953593, 2020. Pristupljeno: 2. lipnja 2020.

[8] Igor S. Pandžić et al. *Virtualna okruženja: Interaktivna 3D grafika i njene primjene*, stranica 22. Element, 2011.

[9] Wikipedia contributors. Advanced Spaceborne Thermal Emission and Reflection Radiometer — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Advanced_Spaceborne_Thermal_Emission_and_Reflection_Radiometer&oldid=961620638, 2020. Pristupljeno: 9. lipnja 2020.

[10] Wikipedia contributors. R-tree — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=R-tree&oldid=955552664, 2020. Pristupljeno: 5. lipnja 2020.

[11] Wikipedia contributors. Shuttle Radar Topography Mission — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Shuttle_Radar_Topography_Mission&oldid=951021374, 2020. Pristupljeno: 2. lipnja 2020.

[12] Wikipedia contributors. Popis najviših nebodera u Hrvatskoj — Wikipedija, Slobodna enciklopedija. https://hr.wikipedia.org/w/index.php?title=Popis_najvi%C5%A1ih_nebodera_u_Hrvatskoj&oldid=5523381, 2020. Pristupljeno: 3. lipnja 2020.

Model dijela grada Zagreba temeljen na realnim podacima

Sažetak

U ovom radu razmatra se postupak automatskog programskog generiranja trodimenzionalnog modela proizvoljnog dijela grada Zagreba. Diskusija je popraćena konkretnom programskom implementacijom rješenja u programskom jeziku *Python* u obliku dodatka za alat *Blender*. Model se temelji na realnim i javno dostupnim podatcima o terenu, zgradama i stablima te se izrađuje u omjeru $1 : 1$. Kao izvori podataka opisuju se i koriste NASA-ine satelitske snimke senzorom ASTER, baza prostornih kartografskih informacija *OpenStreetMap* i Katastar zelenila Zagreba. Generirani model sastoji se od mreže vrhova i poligona bez teksture kojom su predstavljeni teren, zgrade s ravnim krovovima te stabla.

Ključne riječi: 3D model grada, 3D geoinformacije, interaktivna vizualizacija, Zagreb, OpenStreetMap, Blender, Python, Unreal Engine

Zagreb District Model Based on Real Data

Abstract

This thesis examines the procedure of automatic computer generation of a three-dimensional model of an arbitrary Zagreb district. The discussion is accompanied by the implementation of the solution in software using the *Python* programming language in the form of an add-on for *Blender* toolset. The model is based on accurate and publicly available data on terrain, buildings and trees, and is made in $1 : 1$ ratio. NASA's satellite images created using the ASTER sensor, spatial geoinformation database *OpenStreetMap* and the Zagreb Green cadastre are described and used as data sources. The generated city model consists of meshes without textures representing the terrain, buildings with flat roofs and trees.

Keywords: 3D city model, 3D geoinformation, interactive visualization, Zagreb, OpenStreetMap, Blender, Python, Unreal Engine