Santeri Sorjonen

Avoimen karttadatan hyödyntäminen sovelluskehityksessä

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikka

Insinöörityö

10.7.2023

Tiivistelmä

Tekijä:	Santeri Sorjonen
Otsikko:	Avoimen karttadatan hyödyntäminen sovelluskehityksessä
Sivumäärä:	48 sivua
Aika:	7.10.2023

Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Tieto- ja viestintätekniikka
Ammatillinen pääaine:	Ohjelmistotuotanto
Ohjaajat:	Lehtori Jorma Räty

Opinnäytetyön tavoitteena on luoda yksi osa isompaa ohjelmistokokonaisuutta, joka on mobiilipeli, jossa käyttäjä tutkii eri kuntia ja kaupunkeja. Opinnäytetyön kuvaama osa on algoritmi, joka jakaa kunnan ruudukkoon, jonka ruutujen tiheys perustuu ruudun sisältämään karttadataan.

Työssä perehdytään karttadatan käsittelyn kannalta olennaisiin käsitteisiin sekä teknologoihin. Työ kuvaa erinäisiä ongelmia, mitä karttaprojektion myötä voi aiheutua, kun käsitellään karttadataa sekä kuvaa eri yleisiä standardeja karttadatan käsittelyssä.

Sovellus on kehitetty Python-ohjelmointikielellä, ja kehityksen tukena on käytetty useita kirjastoja, kuten Geopandas sekä Shapely. Opinnäytetyön projekti käyttää OpenStreetMap-palvelun tarjoamaa karttadataa algoritmin syötedatana, joten opinnäytetyö myös perehtyy Overpass-rajapinnan käyttöön.

Työn tulos on valmis ohjelma, joka palauttaa kahdella syötearvolla karttadataa, jota voidaan käyttää ohjelmistokokonaisuuden toiminnassa.

Avainsanat: karttadata, openstreetmap, overpass, python, geopandas

Abstract

Author:	Santeri Sorjonen
Title:	Utilization of Open Map Data in Application Development
Number of Pages:	48 pages
Date:	7 October 2023
Degree:	Bachelor of Engineering
Degree Programme:	Information and Communications Technology
Professional Major:	Software Development
Supervisors:	Jorma Räty, Principal Lecturer

The purpose of the study was to create a part of a greater game project for mobile devices, where the user explores different cities and municipalities. The part described in this thesis is an algorithm, that divides the municipality into a grid, in which the density of the grid is related of map data.

The thesis explores concepts and technologies relevant to handling map data. It describes different problems which a developer may encounter when using map data, and describes common standards for using map data in software development.

The software is developed using Python programming language, and libraries such as Geopandas and Shapely have been used to further support the development. The project uses open map data service by OpenStreetMap for the algorithm, so the thesis also explores the usage of Overpass application interface.

The finished part of the software is a ready-to-use application interface, which takes two input values and returns complete map data, ready to be used in the complete software.

Keywords: Map data, OpenStreetMap, Overpass, Python, Geopandas

Sisällys

Lyhenteet

GIS: *Geographic information system,* paikkatietojärjestelmä. Usein viittaa ohjelmistoihin, joita käytetään karttadatan tallentamiseen sekä käsittelyyn.

CRS: Coordinate reference system*,* koordinaattijärjestelmä. Koordinaattijärjestelmä määrittää yksiköt kartalle.

WSG84: *World Geodetic System 1984,* 84 viittaa järjestelmän versioon sekä julkaisuvuoteen. Yleisin käytetty maailmanlaajuinen koordinaattijärjestelmä.

EPSG: *EPSG Geodetic Parameter Dataset* on tietokanta, joka määrittää tunnukset eri geoinformatiikassa käytetyille järjestelmille, kuten koordinaattijärjestelmille.

1 Johdanto

Viimeisen vuosikymmenen aikana mobiililaitteet sekä niiden käyttö on lisääntynyt moninkertaisesti, jonka myötä myös kartta- sekä GPS-sovelluksien saatavuus sekä käyttö on lisääntynyt. [1.] Tämä on avannut mahdollisuuksia uudenlaisille käyttäjäsovelluksille, jotka käyttävät geopaikannusta osana sovelluksen toimintaa. Kartta- sekä geolokaatio-ominaisuuksia löytyy lähes kaikenlaisista sovelluksista, kuten muun muassa urheilu-, vuokraus-, ruokatilaus- sekä viihdekäyttöön tarkoitetuista sovelluksista. [2.] Tässä opinnäytetyössä käydään läpi, miten karttadataa voidaan käyttää viihdekäyttöön tarkoitetun sovelluksen kehityksen tukena.

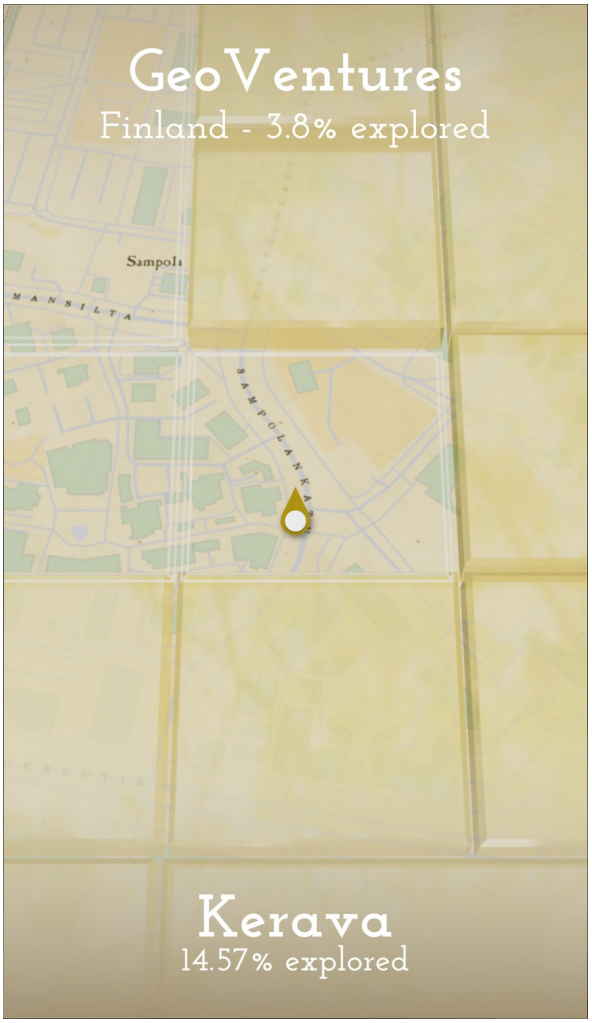
Esimerkkinä pelisovelluksesta, joka käyttää karttadataa sekä geopaikannusta, voi käyttää vuoden 2016 heinäkuussa julkaistua Pokémon GO:ta, joka nousi nopeasti suursuosioon App Store- sekä Google Play -sovelluskaupoissa. Sovellus rikkoi ennätyksen eniten ladatuimmalle sovellukselle ensimmäisen viikon aikana julkaisusta App Store -alustalla ja ylitti yli 50 miljoonan asennuksen Google Play -alustalla ensimmäisen kuukauden aikana. [3.]

Opinnäytetyön tavoitteena on luoda osa tämänkaltaisen mobiilipelin taustajärjestelmää, jossa käyttäjän päätavoitteena on liikkua oikeassa elämässä, samalla tutkien eri alueita. Opinnäytetyön kuvaama taustajärjestelmän osa jakaa kaupungin tai kunnan alueen ruudukkoon, jonka ruutujen tiheys perustuu pääosin karttadatan tiheyteen.

Raportti koostuu projektin, teknologian sekä käsitteiden ja teknisen toteutuksen kuvauksesta. Raportin lopussa kuvataan projektin tuloksia sekä mahdollisia jatkokehitysideoita.

2 Lähtökohdat

Projektin suurempi kokonaisuus on toteuttaa pelisovellus mobiililaitteille. Siinä maailmankarttaa sekä mobiililaitteen GPS-ominaisuuksia käytetään osana peliä. Pelaaja itse liikkuu oikeassa, fyysisessä maailmassa, ja sovellus tarkastelee käyttäjän sijaintia mobiililaitteen GPS-ominaisuuksilla. Käyttäjän on tarkoitus liikkua maailmankartalla, joka on jaoteltu ruudukkoon ja avata ruudukon ruutuja käymällä ruuduissa. Ruudukko on sitä tiheämpi, mitä enemmän alueella on merkittäviä kohteita, kuten teitä sekä rakennuksia. Pelaajan on tarkoituksena käydä mahdollisimman monessa ruudussa, ja käydyt ruudut tallennetaan käyttäjän tilille, jolloin käyttäjä voi seurata omaa etenemistä pelissä.



Kuva 1: Kuvituskuva sovelluksen toiminnasta

Opinnäytetyön päätavoitteena on toteuttaa osa ohjelmistosta, joka luo karttadatan pohjalta ruudukon, jota käytetään sovelluksessa. Tavoitteena on luoda rajapinta, jonka syötearvoina toimii valtion sekä kaupungin nimi, jonka pohjalta sovellus hakee vaadittavan karttadatan, suodattaa sen, ja palauttaa ruudukon pyydetyssä tiedostomuodossa. Algoritmin antama ulostulo voidaan tallentaa tietokantaan, ja täten käyttää sovelluksen kehityksessä.

Ruudukon luonnissa täytyy ottaa erilaisia seikkoja huomioon, jotta sovelluksen käyttäminen on mielekästä. Sovelluksen ideana on, että kaikki ruudut on kävel-
tävissä ja että ruudukko ei sisällä ruutuja, jotka on täysin maanpinnan alapuolel-

la tai vedessä, joten karttadataa jäsentäessä täytyy laskea, kuinka suuri osa ruudusta on vettä. On myös mielekästä, että ruutuja ei luoda alueille, joilla liikkuminen on rajoitettua. Tällaisia alueita voivat olla esimerkiksi lentoalueet sekä sotaharjoitusalueet.

Työn tavoitteena on saada mielekkäitä tuloksia mobiilisovelluksen käyttöä, sekä projektikokonaisuuden jatkokehitystä varten. Sovelluksen käytön kannalta on mielekästä, että ruudukko täyttää sekä myötäilee kunnan alueita, on tiheämpi tiheämmin rakennetuilta alueilta, ja ruudukko ei sisällä ruutuja, joihin pääsy on kiellettyä tai mahdotonta kävellen. Jatkokehityksen kannalta on hyvä, että algoritmi on suhteellisen tehokas, laajennettavissa sekä ohjelmisto-osan tuottama tulos on GeoJSON-tiedostomuodossa. Projektikokonaisuuteen ei ole rakennettu mitään tähän osioon liittyvää, joten opinnäytetyössä kehitetään ohjelman osa kokonaan alusta loppuun.

3 Teknologian kuvaus

Tässä osiossa kuvataan toteutuksen kannalta olennaiset käsitteet.

3.1 Karttadata ja käytetyt standardit

Karttadatan tallentamiseen on monia standardeja sekä muotoja. Projektin kannalta ei ole niinkään tärkeää, missä muodossa data tuodaan järjestelmään tai viedään järjestelmästä, kunhan data noudattaa jotakin ennalta määritettyä standardia, jota sovelluksen käyttämät ohjelmistokirjastot tukevat. Projektissa käytetyt standardit ovat GeoJSON sekä Shapefile.

Algoritmin käyttämä karttadata haetaan Overpass-rajapinnasta. Overpass-rajapinnasta voi hakea karttatietoja lähettämällä siihen kyselyitä. Rajapinta hakee tiedot OpenStreetMap-järjestelmän tietokannasta kyselyn perusteella, ja palauttaa tiedot pyydetyssä formaatissa.

Shapefile on vuonna 1998 esitetty tiedostomuoto geometrisen datan tallentamiseen. Shapefile-tiedostomuodon on kehittänyt yhdysvaltalainen Esri, joka on kehittänyt ArcGis-paikkatietojärjestelmän, jolla on suurin markkinaosuus paikkatietojärjestelmistä. Suurin osa paikkatietojärjestelmistä tukee Shapefile-tiedostomuotoa. Shapefile-tiedostomuotoon voi tallentaa pisteitä, viivoja ja polygoneja sekä määritteitä.

Shapefile-tiedostomuoto koostuu kolmesta tiedostosta: päätiedostosta (.shp), joka sisältää geometrisen datan, indeksitiedostosta (.shx), jonka avulla materiaalin käsittely on tehokkaampaa, sekä dBASE-tietokannasta (.dbf), joka sisältää materiaalin määritteet, kuten muodoille määritetyt nimet sekä kuvaukset. [4.] Shapefile-tiedostomuoto määrittää myös muita tiedostoja, jotka laajentavat materiaalia, mutta eivät ole välttämättömiä. Projektioinformaation sekä koordinaattijärjestelmät voidaan määrittää .prj-tiedostoon. .sbn- sekä .sbx-tiedostoilla voidaan määrittää spatiaalinen indeksi geometriasta, joka nopeuttaa materiaaliin tehtyjä spatiaalisia kyselyitä. Shapefile-formaatti sisältää myös useita muita tiedostoja, jolla karttadataa voidaan laajentaa, ja sen käsittelyä tehostaa, mutta nämä ovat opinnäytetyön kannalta tarpeettomia.

GeoJSON on vuonna 2016 esitetty formaatti geografisen datan tallentamiseen, joka käyttää JSON:n syntaksia, ja formaatille on luotu standardi RFC 7946. [5.] GeoJSON tukee pisteitä (Point), viivoja (LineString), polygoneja (Polygon) sekä listoja, jotka sisältävät edellä mainittuja elementtejä (MultiPoint, MultiLineString, MultiPolygon). GeoJSON-formaatissa tallennettu tietoaineisto tulee ladata kerralla muistiin ja tämän jälkeen käsitellä eri muodossa, sillä GeoJSON on tarkoitettu lähinnä tiedon tallentamiseen ja lukemiseen, jonka myötä operaatiot tässä formaatissa ovat hitaita muihin verrattuna.

GeoJSON-formaatin hyviä puolia on selkeä syntaksi, jota on helppo lukea sekä ihmisen että tietokoneen, jonka myötä myös jäsentäjän kehittäminen on yksinkertaista. GeoJSON-formaatissa tallennettu tietoaineisto koostuu vain yhdestä tiedostosta, jolloin käsittely on helpompaa. [6.]

3.2 OpenStreetMap ja Overpass-rajapinta

OpenStreetMap on yhteisöllinen projekti, joka tarjoaa maailmankartan sekä karttaan sisältyvät datan avoimesti ilman lisensöintiä. [7.] OpenStreetMap-projekti ei ole kaupallinen, joten sen ylläpitäjät lisäävät sekä päivittävät kartan dataa vapaaehtoisesti. Projektiin verrattava tuote sekä suora kilpailija on Googlen tarjoama Google Maps -palvelu.

Overpass-rajapinta on saksalaisen Roland Olbrichtin luoma sekä ylläpitämä tietokantamoottori, joka on kehitetty hakemaan OpenStreetMap-palvelun dataa. [8.] OpenStreetMap Foundation ei tarjoa Overpass-rajapintaa, mutta ulkopuoliset tahot tarjoavat avoimesti käytettäviä Overpass-instansseja. Overpass-rajapinta ei tee kyselyitä suoraan OpenStreetMap-palvelun käyttämään tietokantaan, mutta tietokannasta on luotu kopio, josta informaatio kysytään. Täten varmistetaan, että Overpass-rajapinnan käyttö ei hidasta OpenStreetMap-palvelun pääinstanssia.

Overpass-rajapintakyselyitä voidaan luoda joko Overpass QL -muodossa tai Overpass XML -muodossa. Ensimmäisenä kehitetty Overpass XML -syntaksi käyttää XML-syntaksia kyselyiden luomiseen. Overpass QL -syntaksi kehitettiin Overpass XML -syntaksin rinnalle, sillä rajapintaan kehitettiin toiminto luoda kyselyitä yksittäisellä HTTP GET -pyynnöllä. Overpass QL -kyselykieli on Overpass-kehittäjien luoma C-ohjelmointikielen tyylinen syntaksi, jota kuvataan proseduraaliseksi sekä imperatiiviseksi ohjelmointikieleksi. [9.] Overpass QL -syntaksi on suppeampaa verrattuna Overpass XML -kyselykieleen, ja sitä voidaan pitää myös epäselkeämpänä, sillä XML-syntaksin elementit ovat itseään kuvaavia.

Overpass-kyselyiden tärkein osa on kyselyt itse, jolla voidaan hakea tietokannasta eri sisältötyyppejä. Tietokannasta kyseltävät tyypit ovat [10.]

- node, joka on kaksiulotteinen piste, jolle on määritetty pituuspiiri sekä leveyspiiri

- way, joka kuvaa karttadataa käyränä
- area, joka kuvaa karttadataa alueena jolla on pinta-ala, eli polygonilla
- relation, joka yhdistää muita tyyppejä ja määrittää myös luokittelut muille tyypeille sekä kokonaisuuksille.

Kyselyihin voidaan lisätä suodattimia, jolla suodatetaan karttadataa siihen lisättyjen tunnistimien perusteella. Suodattimet lisätään hakasuluilla kyseltävän tyypin jälkeen, johon voidaan syöttää avainsanapareja. [11.]

Overpass QL -ohjelmointikieli myös määrittää erillisen suodattimen kyselyille, joka merkitään kaarisuluilla kyselyn perään. [12.] Ohjelmointikielen dokumentaatio määrittää monia eri suodattamia eri käyttötarkoituksiin, mutta tässä projektissa käytetään suodatinta vain alueen määrittämiseen. Kaarisulun sisälle voi määrittää area-tyyppisen sisällön, jolloin suodatin suodattaa kaiken muun karttadatan, joka on alueen sisällä. Tämä on kuvattuna Esimerkkikoodi 1 -kyselyssä.

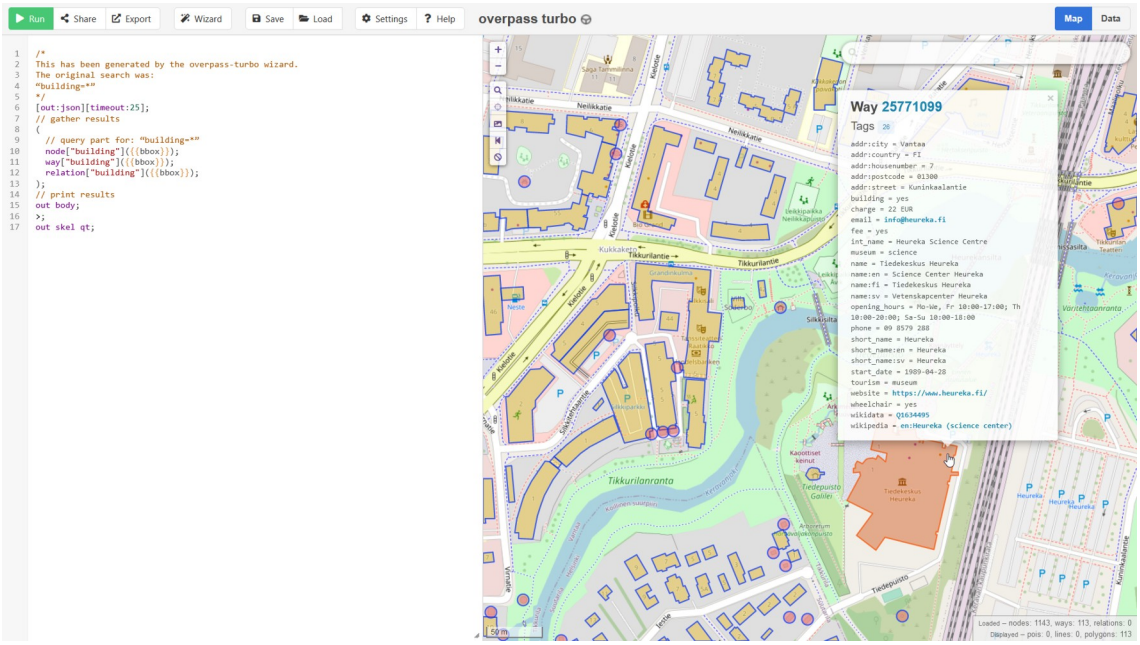
```
[out:json];
area["name"="Vantaa"];
(
  way["highway"]["name"="Kielotie"](area);
);
out geom;
```

Esimerkkikoodi 1: Esimerkkikysely Overpass QL -kielellä, joka hakee Kielotien geometrian Vantaalta.

Ohjelmointikieli tukee myös useita muita operaatiota, joita on toteutettu muihin kyselykieliin, kuten joukko-operaatioita, alikyselyitä sekä aggregaattifunktioita. Koska tässä projektissa tehdyt kyselyt on melko yksinkertaisia, suurinta osaa näistä funktiota ei käytetä tässä projektissa.

Overpass-rajapinta voi palauttaa datan joko XML-, CSV- tai JSON-formaatissa. [13.] Oletuksena rajapinta palauttaa datan XML-muodossa. Jos kyselyyn määritetään, että data palautetaan CSV -formaatissa, voidaan määrittää, mitä karttadataa dokumenttiin halutaan. On hyvä ottaa huomioon, että Overpass-rajapinnan palauttama JSON-data ei ole GeoJSON-formaatissa.

Overpass-kyselyitä voi luoda, kokeilla sekä tarkastella käyttämällä Overpass-



Kuva 2: Näyttövedos Overpass Turbo -palvelusta

OpenStreetMap-palvelun tietokannan tieto on kattava, pääosin ajan tasalla ja siihen on yhteisöllisesti lisätty vuosikymmenien ajan dataa, joten Overpass-rajapinnan tarjoama karttadata on sopiva projektiin. [16.] Overpass-kyselykieli on myös kattava, ja sitä käyttäen on mahdollista hakea dataa myös komplekseihin tilanteisiin. OpenStreetMap -wikisivusto kuvaa kaikki mahdolliset datatyypit, jota rajapinnasta voidaan hakea ja suodattaa.

Projektissa lähes kaikki karttadata haetaan Overpass-rajapinnasta, tai on haettu Overpass-rajapinnasta ja käsitelty ennalta ennen sen käyttöä. Rajapinnasta haetaan tietoa kaupunkikohtaisesti, kuten kaupungin kaikki tiet sekä rakennukset.

3.3 99boundaries.com

99boundaries.com on avoin palvelu, joka tarjoaa eri valtioiden rajat polygoneina. [17.] Palvelu tarjoaa kahdenlaisia muotoja: rajat jotka sisältävät meret, ja rajat, joista on meret otettu pois. Näistä jälkimmäinen on kiinnostavampi projektissa, sillä projektissa lasketaan, kuinka iso osa ruudusta on maata.

Polygonit on laskettu OpenStreetMap-karttadatan pohjalta. OpenStreetMap ei suoraan tarjoa polygoneja maalle tai merelle, mutta tarjoaa rannikkoviivat, joiden pohjalta voidaan täyttää polygoni. Openstreetmap.de tarjoaa tietoaineistot, joihin tämä prosessi on tehty, mutta tietoaineisto sisältää koko maapallon. 99boundaries on jakanut tämän tietoaineiston osiin valtioittain, ja tarjoaa kyseiset osat avoimeen käyttöön. [18.]

3.4 Python

Python on alun perin vuonna 1990 hollantilaisen Guido van Rossum kehittämä ohjelmointikieli, joka on sittemmin noussut yhdeksi käytetyimmistä ohjelmointikielistä [19]. Pythonin tekijänoikeudet omistaa Python Software Foundation, joka myös ylläpitää kielen kehitystä. Python-ohjelmointikielien hyviä ominaisuuksia ovat helppo syntaksi sekä luettavuus. Suuren suosion ansiosta, kehityksen sekä käytön tueksi on tarjolla paljon avoimia ohjelmakirjastoja eri käyttötarkoituksiin. Python on myös suosittu eri tiedepiireissä ja sitä käytetäänkin usein matemaattisissa sekä tieteellisissä ympäristöissä. [20.]

Ohjelmointikieli on tulkattava koodikieli, eli Python-ohjelman ajaminen on hitaampaa kuin perinteisen ohjelmointikielen, jonka lähdekoodi käännetään ohjelmaksi ennen suoritusta. [21.]

Projektiin oli alun perin tarkoitus käyttää Node.js-kehitysympäristöä, joka käyttää JavaScript-ohjelmointikieltä, mutta Node.js-kehitysympäristölle ei löytynyt tarpeeksi kattavaa kirjastoa karttadatan käsittelemiseen. Python-kehitysympäristölle kehitetty Geopandas-kirjasto osoittautui sekä kattavammaksi että huomattavasti tehokkaammaksi useissa eri operaatiossa verrattuna Node.js-kehitysympäristön vastaavaan kirjastoon.

Ohjelmakokonaisuuden muut osat kirjoitetaan toisella ohjelmointikielellä kuin Python-ohjelmointikielellä, mutta projektista on tarkoitus tehdä rajapinta, jonka myötä tällä ei ole merkitystä. Tähän osaa ohjelmaa syötetään parametriarvot, ja

ohjelma palauttaa GeoJSON-dataa, jonka myötä ohjelmointikielellä ei ole väliä suorituksen kannalta.

3.5 Pandas/Geopandas

Pandas on Python-ohjelmointikielellä kehitetty avoimen lähdekoodin kirjasto data-analyysia varten. [22.] Pandas-kirjasto sisältää DataFrame-tietorakenteen, joka toteuttaa kaksiulotteisen taulukkodatarakenteen, johon sisältyy rivejä sekä sarakkeita, jossa kaikilla riveillä sekä sarakkeilla on tunnukset. Pandas-kirjasto toteuttaa myös Series-tietorakenteen, joka on yksiulotteinen lista arvoista, jolla DataFrame-tietorakenteen sarakkeet täytetään arvoilla. Kirjaston toiminnallisuus perustuu näihin kahteen tietorakenteeseen sekä niiden ympärille rakennettuihin funktioihin, jolla tietoa käsitellään.

GeoDataFrame

index	geometry
0	POLYGON ((21.56308 62.98483, 21.55260 62.98222…
1	POLYGON ((21.56308 62.98483, 21.55260 62.98222…
2	LINESTRING (21.60050 63.09852, 21.60051 63.098…
3	POINT (21.61333 63.09531)

Kuva 3: GeoDataFramen rakenne visualisoituna

Geopandas on Pandas-työkalun päälle kehitetty ohjelmistokirjasto, joka lisää funktionaalisuuden analysoida sekä käsitellä geometrisiä tyyppejä. [23.] Geopandas toteuttaa GeoDataFrame- sekä GeoSeries-tietorakenteet, joiden periaate on sama kuin Pandas-kirjaston tietorakenteiden. GeoDataFrame- tietorakenne voi sisältää kolumneja, jotka sisältävät geometristä dataa. Geometrinen data tuodaan rakenteeseen GeoSeries-datarakenteessa. Kuten Pandas-kirjasto, Geopandas-kirjasto toteuttaa näiden tietorakenteiden ympärille funktiota, jolla dataa analysoidaan sekä käsitellään.

Geopandas tukee myös koordinaattijärjestelmiä, jonka myötä karttadataa voidaan projektoida sekä uudelleen projektoida käyttämään eri koordinaattijärjestelmiä. [24.] Tämä on tarpeellista, sillä Overpass-rajapinnasta haettu data on tallennettu käyttäen WSG84-koordinaattijärjestelmää, ja ilman tukea koordinaattijärjestelmälle mahdolliset geometriset operaatiot voivat tuottaa vääriä tuloksia.

Projektissa Geopandas-kirjasto on suuressa käytössä, sillä sen tuomat funktiot sekä geometriset operaatiot tukevat projektin kehitystä. Karttadata tuodaan GeoDataFrame-datarakenteisiin, jossa GeoDataFrame-luokan toteuttamia metodeja käytetään datan analyysiin, käsittelyyn sekä toisten GeoDataFrame-instanssejen vertailuun. GeoDataFrame voidaan tallentaa suoraan GeoJSON-formaatissa.

3.6 Shapely

Shapely on Geopandas-kirjaston käyttämä Python-paketti geometristen objektien käsittelyyn sekä analyysiin. Shapely hyödyntää yleisesti käytettyä GEOS - C/C++ laskennalliseen geometriaan tarkoitettua kirjastoa funktionaalisuuden toteuttamiseen, ja toteuttaa Python-käyttöliittymän GEOS-kirjaston käyttöä varten Python-kehitysympäristössä. [25.]

Shapely toteuttaa kolme geometristä tyyppiä, pisteen, käyrän sekä monikulmion. Pistettä edustaa Point-luokka, käyrää edustaa LineString- sekä LinearRing-luokat, ja monikulmiota edustaa Polygon-luokka. [25.]

Sillä Shapely on osa Geopandas-kirjastoa, on hyvä ymmärtää myös Shapely -kirjaston toiminta. Projektissa Shapely-kirjaston geometrisiä tyyppejä käytetään karttadatan analysoimiseen sekä Overpass-rajapinnasta haettu data jäsennetään Shapely-kirjaston toteuttamiin geometrisiin luokkiin.

3.7 Karttaprojektio, CRS (Coordinate Reference System) ja WGS84

Sillä Maapallo ei ole litteä objekti, Sen projektoimisessa sekä piirtämisessä kaksiulotteiselle alustalle täytyy tehdä kompromisseja. [26.] Tunnetuin karttaprojektiomenetelmä on Mercatorin projektio, joka on lieriöpohjainen projektio. Koska Mercatorin projektio on oikeakulmainen, leveys- sekä pituuspiiri ovat aina samansuuntaisia, jolloin suora viiva kartalla vastaa myös suoraa viivaa oikeassa elämässä. [27.] Täten projektio soveltuu hyvin navigoimiseen.

Mercatorin projektion ongelmana on, että kartan mittasuhteet vääristyvät sitä lähempänä etelä- ja pohjoisnapaa. [27.] Kartalla, johon on piirretty leveys- sekä pituuspiirien viivat ,voidaan havaita, että Suomen kohdalle muodostuneet ruudut ovat huomattavasti eri kokoisempia kuin päiväntasaajalla olevat ruudut.

OpenStreetMap sekä muut karttasovellukset käyttävät vuonna 2005 Googlen julkaisemaa WebMercator-projektiota maailmankartan piirtämiseen, joten vääristymän aiheuttamat ongelmat täytyy ottaa huomioon tässä projektissa. [28.]

Yleisin käytetty geografinen koordinaattijärjestelmä karttaprojektioissa on World Geodetic System 1984. [29.] Se määrittää leveyspiirin, joka vastaa kaksiulotteisella projektiolla vaaka-akselia sekä pituuspiirin, joka vastaa pysty-akselia.

WGS84 käyttää yksikkönä asteita, joten se ei sovellu etäisyyksien mittaamiseen. Karttadata voidaan projektoida uudestaan toiseen koordinaattijärjestelmään, joka käyttää metrejä yksikköinä, jolloin etäisyysmittauksia on mahdollista tehdä. Tätä varten joissain tapauksissa projektissa projektoidaan karttadataa uudestaan Maanmittauslaitoksen määrittämään Kkartastokoordinaattijärjestelmään, jonka avulla mittauksia voidaan suorittaa metrin tarkkuudella Suomen alueella. [30.]

Kaikille Geopandas-dataframeille tulee määrittää koordinaattijärjestelmä, sillä ohjelmaan ladattava karttadata on tallennettu käyttäen koordinaattijärjestelmää. Geopandas myös antaa varoitusviestejä ja estää tekemästä joitakin operaatiota,

jos koordinaattijärjestelmää ei ole määritetty. Koska OpenStreetMap käyttää WGS84-koordinaattijärjestelmää sovelluksessa, myös data, joka saadaan Overpass-rajapinnasta, on tallennettuna tällä järjestelmällä. Täten Geopandas-dataframeille määrätään sama WGS84-koordinaattijärjestelmä.

EPSG-rekisteri on avoin tietokanta, joka tarjoaa määritelmät monille geodesiassa tärkeille konsepteille, muun muassa eri datumeille ja koordinaattijärjestelmille. [31.] Dataframeille määritetään koordinaattijärjestelmä EPSG-tunnuksen perusteella. WSG84-geograafisen koordinaattijärjestelmän EPSG-tunnus on ”EPSG:4326”. Kartastokoordinaattijärjestelmän EPSG-tunnus on ”EPSG:3387”.

4 Toteutuksen kuvaus

Projektin tarkoituksena on toteuttaa algoritmi, joka suodattaa kaupungeittain haettua karttadataa ja tuottaa tästä sovelluksen käytön kannalta käytännöllisen ruudukon.

Algoritmi ajetaan Python-ohjelmistoympäristössä, joten ensimmäiseksi täytyy asentaa Python-kehitysympäristö sekä projektin kannalta olennaiset kirjastot. Tässä vaiheessa projektia algoritmin tallentamaa dataa voidaan helposti tarkastella geojson.io-sivustolta, johon syöttämällä GeoJSON-dataa saadaan tulos saman tien piirrettyä.

Suurin osa algoritmin käyttämästä datasta haetaan Overpass-rajapinnasta, mutta valtion maarajat haetaan 99boundaries.net-palvelusta. Overpass-rajapinnasta haetut datat haetaan dynaamisesti syöttämällä algoritmiin kaupungin nimi sekä tunnus, jonka Overpass tunnistaa.

Algoritmi on pääosin yksi rekursiivinen funktio, joka karttadatan pohjalta päättää, jaetaanko ruutu aina neljään osaan. Jos algoritmin ehdot toteutuvat, sama funktio ajetaan ruutuihin, jolloin ruudukosta tulee tietyin osin tiheämpi, mutta tietyiltä osin jää väljäksi. Rekursiivisella funktiolla tulee olla tapaus, jossa rekursio loppuu, jotta algoritmin suoritus ei ole ikuinen.

4.1 Backend-järjestelmän asennus

Projekti on toteutettu 3.10.10-versiolla Pythonista, joka on asennettuna Ubuntu-käyttöjärjestelmälle Linux Subsystem for Windows -ympäristössä. Python-ohjelma myös suoritetaan Ubuntu-käyttöjärjestelmällä.

Geopandas-kirjasto asennetaan conda-paketinhallintaohjelmistolla, sillä conda suoraviivaistaa riippuvuuksien asentamisen, ja täten Geopandas-kirjaston vaatimat kirjastot asentuu oikein. Conda-paketinhallintaohjelmistolla luodaan kehitysympäristö, johon kirjastot asennettiin.

Projektille luodaan uusi hakemisto. Tässä tapauksessa hakemiston nimi on ”runko”, ja sen sisälle luodaan pääohjelman tiedosto, ”main.py”. Samaan hakemistoon tallennetaan ennalta ladattu karttadata, eli valtion maarajat. Tähän kansioon tallennetaan myös sovelluksen tallentamat ruudukot GeoJSON-tiedostoina.

4.2 Karttadatan hakeminen

Ennen algoritmin suoritusta, ohjelma hakee rajapinnoista vaaditut datat tiedon suodattamista varten. Ohjelma hakee seuraavat karttadatat kunnittain:

- kunnan muodot polygoneina
- kunnan rakennukset pisteinä
- kunnan tiet, polut käyrinä
- kunnan vesistöt (pois lukien merivesistöt)
- kielletyt alueet jotka ovat kunnan sisällä.

OpenStreetMap ei tarjoa Overpass-rajapintaa, joten projektissa joudutaan käyttämään kolmannen osapuolen tarjoamaa Overpass-rajapintaa. [32.] OpenStreetMap-wikijärjestelmästä löytyy listaus avoimesti käytettävistä rajapinnoista, ja tässä projektissa kaikki kyselyt tehdään listasta löytyvään, overpass-api.de-tarjoamaan rajapintainstanssiin.

Jo ennen ohjelman ajamista valtion maarajat on ladattu 99boundaries.com-palvelusta ja tallennettu samaan hakemistoon, missä pääohjelma on, josta maarajat ladataan ohjelman muistiin. Valtion rajat on GeoJSON-formaatissa, joten tiedosto on mahdollista avata ilman käsittelyä GeoPandas-kirjaston funktiolla “read_file”. Funktio luo tiedostosta GeoDataFrame-olion, jota voidaan verrata muihin GeoDataFrame-olioihin. Tämä osa ohjelmasta on manuaalinen, esimerkiksi jos algoritmi halutaan suorittaa Ruotsin Tukholman perusteella, Ruotsin maarajat pitää erikseen ladata 99boundaries.com-palvelusta.

Karttadata haetaan kunnittain, joten kuhunkin kyselyyn tulee määrittää OpenStreetMap-tietokantaan määritetty kunnan tunniste. Kysely kunnan tunnisteen hakemiseksi tehdään ennen muita kyselyitä, jotta saatua informaatiota voidaan käyttää muiden kyselyiden luomisessa.

Ohjelma lukee kaksi käyttäjän syöttämää komentoriviargumenttia, valtion nimen sekä kunnan nimen, jonka perusteella kunnan tunniste kysytään Overpass-rajapinnasta. Koska maailmassa voi olla useita samannimisiä kuntia sekä kaupunkeja kuin haettava kunta, tuloksia suodatetaan valtion nimellä. Valtion nimi syötetään OpenStreetMap-tietokannan määrittämällä ”int_name”-tunnistimella (international name), sillä ”name”-tunnistin on epäsäännöllinen, ja suomenkielinen ”name:fi”-tunnistin ei skaalaudu välttämättä hyvin projektin kasvaessa.

Valtio haetaan syöttämällä ohjelmaan ”area”-tyypin kysely, johon määritetään suodatin ”int_name”-tunnisteella. Tämän alle luodaan aliohjelma, jossa kunnan tiedot haetaan. Relaatio-objekti, josta kunnan tunnus haetaan sisältää tunnisteen joka määrittää, että relaatio sisältää kunnan hallinnolliset rajat. Hallinnolli-

sia rajoja haetaan ”relation”-kyselyllä, ja määrittämällä kyselyyn tunnistesuodatin ”boundary”-arvolla ”administrative”.

Tulokset suodatetaan hakemalla vain relaatiot, joiden nimi vastaa kunnan nimeä. Suodatin lisätään määrittämällä ”name”-tunnisteen arvoksi kunnan nimi. Jotta kunnan rajat haetaan aikaisemmin määrätystä valtiosta, kyselyihin lisätään suodatin, jonka arvoksi annetaan ”area”.

```
[out:json];
area[int_name="Finland"];
(
  relation["boundary"="administrative"]["name"="Pori"](area);
);
out body;
```

Esimerkkikoodi 2: Kysely joka hakee Pori-solmun relaation

Sillä kyselystä ei tarvita muuta informaatiota kuin kunnan tunniste, ohjelma ei hae muuta informaatiota kuin relaation. Relaatiosta saadaan kunnan indeksitunnus kokonaislukuna, johon täytyy vielä lisätä 3600000000 ennen kuin tunnusta voidaan käyttää suodatukseen.

Kunnan rajat haetaan Overpass-rajapinnasta kunnan tunnuksen mukaan. Ohjelmaan voidaan määrittää “boundary”-kysely, joka palauttaa rajapinnasta kunnan hallinnollisen rajan. Kyselyyn määritetään suodatin, joka hakee relaation kunnan nimen perusteella. Kyselyyn määritetään, että tulos palautetaan geometrisinä datana. Tulos jäsennetään Shapely-kirjastoilla polygoneiksi, ja polygonit viedään uuteen GeoDataFrame-olioon. Jäsentäjän tulee ottaa huomioon, että kunta voi koostua yhdestä tai useammasta polygonista, joten kaikki polygonit tulee lisätä GeoDataFrame-olioon. GeoDataFrame-oliolle täytyy määrittää käytettävä koordinaattijärjestelmä. GeoDataframe-oliolle määritetään käyttämään WSG84-koordinaattijärjestelmää EPSG:4326 -tunnuksella.

```
[out:json];
area(id:3604446212)->.searchArea;
(
  relation["type"="boundary"]["name"="Pori"](area.searchArea);
);
out geom;
```

Esimerkkikoodi 3: Kysely joka hakee Porin kunnan rajat geometrian

Sillä myös sisävesistöjä kuvataan projektissa polygoneilla, prosessi niiden hakemiseen sekä jäsentämiseen on lähes sama kuin kunnan rajojen. Kyselyyn määritetään ”way[’natural’=’water’]”, joka hakee tietokannasta vesistöt viivoina. Kyselyyn voidaan myös määrittää ”relation[’natural’=’water’]”, jolloin vesistöistä saadaan myös lisätietoa kuten siihen liitetyt tunnisteet.

OpenStreetMap -dokumentaatio määrittää avainsanan ”water” sisältävän minkä tahansa vesimuodostuman riippumatta, onko se luonnollinen vai keinotekoinen. Avainsana ei kuitenkaan sisällä merivesiä, joten merivedet käsitellään eri tavalla. Sisävesistöt lisätään uuteen GeoDataFrame-olioon, ja olio määritetään käyttämään WSG84-koordinaattijärjestelmää EPSG:4326-tunnuksella.

```
[out:json][timeout:25];
area(id:3604446212)->.searchArea;
(
  way["natural"="water"](area.searchArea);
  relation["natural"="water"](area.searchArea);
);
out geom;
```

Esimerkkikoodi 4: Kysely joka hakee kaikki sisävesistöt ja siihen liittyvät relaatiot Porista

OpenStreetMap-palvelun sisältämät rakennukset voidaan hakea ”way[’building’]”-kyselyllä, joka hakee rakennukset viivoina. Vaikka rakennukset kuvataan polygoneina, viivoista voidaan muodostaa polygoneja. Myös rakennuksista voidaan hakea lisätietoa määrittämällä ”relation[’building’]”.

OpenStreetMap-dokumentaatio määrittää, että mikä tahansa ihmisen rakentama rakennus, jossa on katto sekä pääosin pysyy paikoillaan voidaan luokitella tämän avainsanan alle. [33.] Täten esimerkiksi paikoillaan olevat asuntoautot voivat lukeutua tämän avainsanan alle. Koska avainsanan määritelmä ei mainitse seiniä, esimerkiksi autokatokset voidaan myös laskea rakennuksiksi.

```
[out:csv(
      ::id, ::lat, ::lon)
    ];
area(id:3604446212)->.searchArea;
(
  way["building"](area.searchArea);
  relation["building"](area.searchArea);
);
out center;
```

Esimerkkikoodi 5: Kysely joka hakee kaikki rakennukset ja niihin liittyvät relaatiot Porista, ja palauttaa ne CSV-tiedostomuodossa

Rakennukset voidaan käsitellä polygoneina, mutta tämä on tarpeetonta ja tekisi myös suorituksesta raskaampaa, joten rakennukset palautetaan keskipisteinä ja käsitellään myös pisteinä. Rakennukset pyydetään CSV-muodossa, sillä kyselyssä voidaan määrittää, mitä tietoja rakennuksesta tulokseen tulee. Koska rakennuksesta ei tarvita kuin koordinaatit, jää jäsennettävä määrä tietoa huomattavasti pienemmäksi. Jokainen CSV-rivi käydään läpi, ja niiden perusteella luodaan Shapely-pisteitä. Pisteet lisätään GeoDataFrame-olioon, josta pisteitä voidaan verrata muihin GeoDataFrame-olioihin. GeoDataFrame-olio määritetään käyttämään WSG84-koordinaattijärjestelmää EPSG:4326-tunnuksella.

Ohjelman muistiin haetaan myös kaikki kunnan rajojen sisäpuolella olevat kulkuväylät. Tiet käsitellään viivoina, joten ne voidaan hakea tietokannasta lisäämällä kysely ”way[’highway’]”. Kyselyllä haetaan kaikki ajotiet kuten kadut, kujat sekä moottoritiet. Kysely palauttaa myös kaikki kävelytiet, pyörätiet sekä pienet polut. Rajapinta palauttaa kokoelman pistelistoja, jotka jäsennetään Shapely-kirjaston LineString-luokkiin. LineString-oliot lisätään GeoDataFrame -olioon josta käyriä voidaan verrata muihin GeoDataFrame-olioihin. Myös tämä GeoDataFrame-olio määritetään käyttämään WSG84-koordinaattijärjestelmää EPSG:4326-tunnuksella.

```
[out:json][timeout:25];
area(id:3604446212)->.searchArea;
(
  way["highway"](area.searchArea);
  relation["highway"](area.searchArea);
);
out geom;
```

Esimerkkikoodi 6: Kysely, joka hakee kaikki tiet ja niihin liittyvät relaatiot porista

OpenStreetMap-tietokanta ei määritä tunnistetta alueille, jossa vapaa liikkuminen on rajoitettua, joten alueiden hakua varten joudutaan luomaan useita kyselyitä. Projektissa haetaan kahdenlaisia alueita, joista ensimmäinen ovat armeijan alueet. OpenStreetMap määrittää tunnisteen ”landuse”, johon voidaan määrittää arvoksi ”military”. Tämä tunniste määritetään kaikille aluille, jotka on armeijan käytössä. [34.] Toiset alueet, jotka haetaan, ovat lentokentät. OpenStreetMap määrittää tunnisteen ”aeroway”, joka sisältää lentämistä varten rakennetun infrastruktuurin. ”aeroway”-tunnistelle voidaan määrittää ”aerodrome”-arvo, joka hakee lentokentät. [35.] Kaksi tai useampaa kyselyä voidaan yhdistää ohjelmassa ilman operaattoreita asettamalla uuden kyselyn uudelle riville. Molempien tyyppien alueet haetaan area-kyselyllä, sillä ne käsitellään alueina. Koko kyselyn tulos palauttaa datan polygoneina, ja polygonit jäsennetään uuteen GeoDataFrame-olioon samalla funktiolla kuin sisävesistöt. GeoDataFrame-oliolle määritetään WSG84-koordinaattijärjestelmä EPSG:4326-tunnuksella.

```
[out:json];
area(id:3604446212)->.searchArea;
(
  way["landuse"="military"](area.searchArea);
  relation["landuse"="military"](area.searchArea);
  way["aeroway"="aerodrome"](area.searchArea);
  relation["aeroway"="aerodrome"](area.searchArea);
);
out geom;
```

Esimerkkikoodi 7: Kysely joka hakee Porissa olevat lentokentät sekä sotakäyttöön tarkoitetut alueet

4.3 Neliön luominen ja jakaminen pienempiin ruutuihin

Kartalle piirretään neliö, joka täyttää kunnan rajat. Tavallisessa kaksiulotteisessa ruudukossa tämä onnistuu laskemalla suorakulmio kunnan äärimmäisten pisteiden perusteella. Suorakulmion pitkän sivun erotus pienemmästä sivusta summataan kumpaankin lyhyempään sivuun, jolloin suorakulmiosta tulee neliö.

Sillä koordinaattiruudukkoon on implimentoitu karttavääristymää korjaava koordinaattijärjestelmä. Yksi mittayksikkö vaaka-akselilla ei ole yhtä pitkä kuin yksi mittayksikkö pysty-akselilla. Vaaka- sekä pysty-akselin kokosuhde myös vaihtelee sijainnin mukaan. Täten neliö voi olla hyvin vääristynyt, ja se voi näyttää suorakulmiolta.

Projektin sekä sovelluksen käytön kannalta on tärkeämpää, että luotu ruudukko on neliö, tai lähes neliötä muistuttava suorakulmio, sillä se on visuaalisesti miellyttävämpi kuin epäsäännöllisen kokoiset suorakulmiot. Täten sivujen laskemiseen täytyy käyttää toista menetelmää.

Geopandas-kirjasto ei tarjoa suoraa funktionaalisutta neliön piirtämiseen muodon ympärille, mutta Shapely-kirjastolla voidaan piirtää ympyrä, johon määritetään ympyrän keskipiste sekä säde. Kunnan rajojen perusteella piirretään ympyrä, jonka keskipiste on kunnan rajojen keskellä, ja säde on etäisyys kauempana olevan sivun keskikohtaan. Jotta etäisyys voidaan laskea, pisteiden GeoDataFrame täytyy projektoida uudelleen käyttämään geografista koordinaattijärjestelmää. GeoDataFrame projektoidaan uudelleen käyttämään metri-

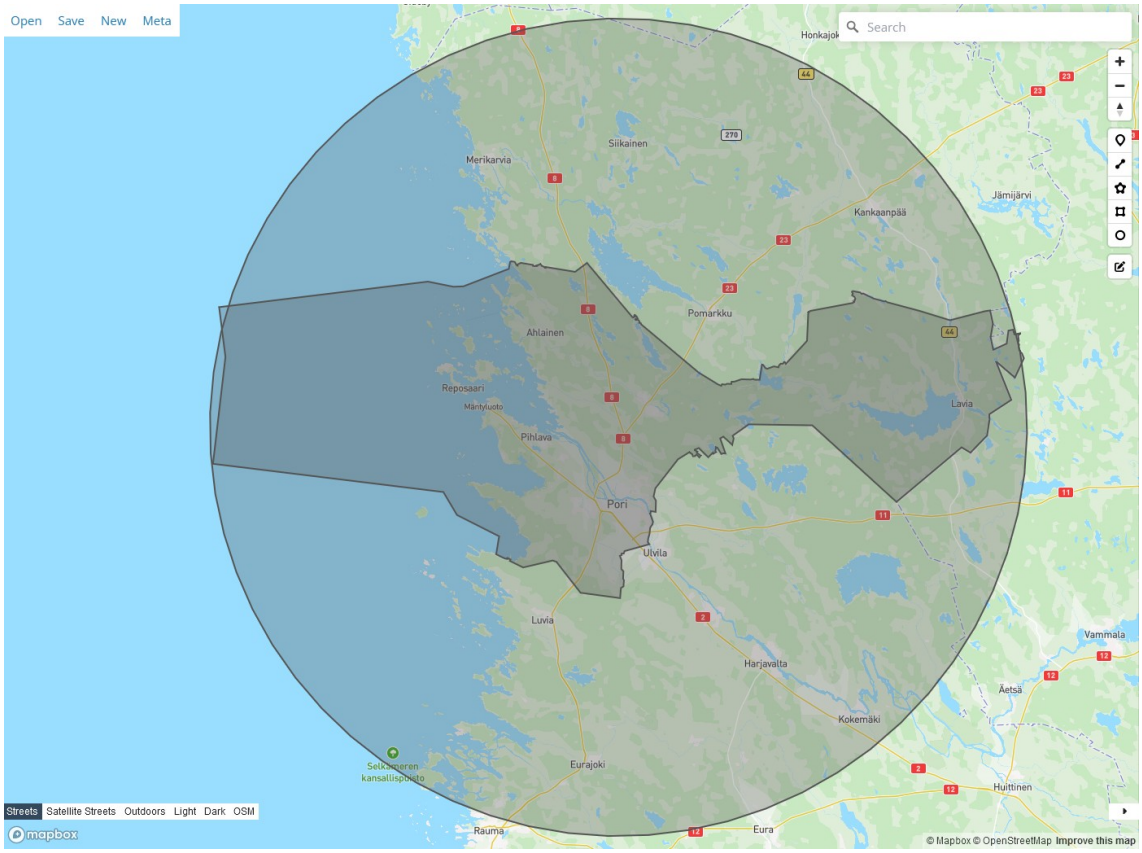
pohjaista, Suomen alueelle määritettyä Kartastokoordinaattijärjestelmää tunnuksella EPSG:3387. Uudelleen projektoidusta GeoDataFramesta voidaan laskea pisteiden etäisyys ja GeoDataFrame projektoidaan takaisin käyttämään WGS84-koordinaattijärjestelmää.

Sillä Shapely-kirjasto ei tue ympyriä, sen luoma ympyrä on ympyrän piirin varrelle tasavälein tallennettuja pisteitä, jotka yhdistämällä muodostuu ympyrää muistuttava käyrä. Muodostunut objekti ei ole kuitenkaan visuaalisesti vääristynyt, joten sen perusteella voidaan laskea suorakulmio, jonka reunat ovat visuaalisesti lähes samanpituiset. Objektista lasketaan kaksi pistettä:

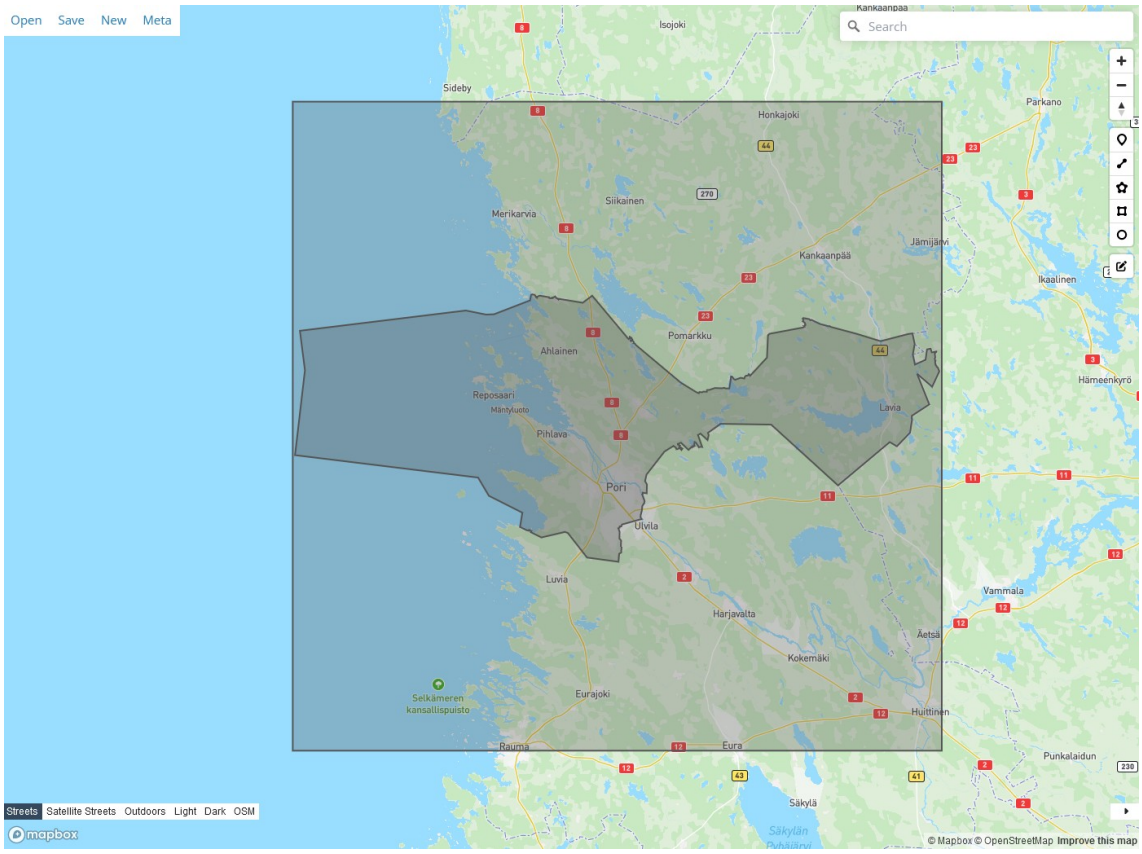
- Piste jonka x-koordinaatti on ympyrän vasemmaisimman pisteen sijainti vaaka-asteikolla, ja y-koordinaatti on ympyrän ylimmäisimmän pisteen sijainti pystyasteikolla.

- Piste jonka x-koordinaatti on ympyrän oikeimaisimmanpisteen sijainti vaaka-asteikolla, ja y-koordinaatti on ympyrän alimmaisimman pisteen sijainti pystyasteikolla.

Näistä pisteistä saadaan laskettua suorakulmio, joka sisältää kunnan rajat. Laskettu suorakulmio syötetään rekursiiviseen funktioon, joka jakaa suorakulmion aina neljään pienempään osaan.



Kuva 4: Kunnan rajojen ympärille muodostettu ympyrä



Kuva 5: Ympyrän perusteella laskettu suorakulmio, joka syötetään algoritmiin

Jotta suorakulmio voidaan jakaa neljään osaan, suorakulmiosta sekä sen sivuista lasketaan keskipisteet. Kaikki pisteet syötetään uuteen kaksiulotteiseen listaan, jonka ensimmäinen indeksi vastaa pisteen sijaintia vaakatasolla, ja toinen indeksi vastaa pisteen sijaintia pystytasolla. Kaksiulotteisen listan yksittäisen solun arvo on lista, joka sisältää pisteen pituus- sekä pystypiiriarvot. Listan arvojen perusteella voidaan luoda neljä samansuuntaista sekä samansuuruista suorakulmiota, jotka muodostavat ruudukon.

Funktiossa nämä neljä pienempää suorakulmiota ovat erillisiä Shapely-kirjaston Polygon-luokkia, ja yksittäinen suorakulmio syötetään uudestaan samaan funktioon niin monta kertaa, kuin ruudukkoa halutaan tihentää.

4.4 Datan suodatus

Rekursiiviselle funktiolle tulee kirjoittaa ehdot, milloin rekursio jatkuu tai milloin rekursion ketju keskeytetään. Funktiolla on kolme eri tapausta sen jatkumisen suhteen:

- Käsitelty suorakulmion kaikkiin neljää osaa syötetään funktioon, jolloin rekursio jatkuu.

- Suorakulmiota ei jaeta neljään osaan, ja se palautetaan sellaisenaan jolloin rekursio loppuu.

- Suorakulmiota ei palauteta, ja sen sijaan palautetaan tyhjä lista, jolloin rekursio loppuu.

Funktiossa on useita ehtolausekkeita, jotka määrittävät kuinka rekursio jatkuu:

- Tarkastetaan kuinka suuri osa suorakulmiosta on päällekkäin kunnan rajoista muodostetun polygonin kanssa.

- Tarkastetaan, kuinka suuri osa suorakulmiosta on maalla verrattuna meriin sekä järvivesiin.

- Tarkastetaan, miten paljon suorakulmion sisällä on rakennuksia sekä teitä.

- Tarkastetaan, että suorakulmion pinta-ala on suurempi kuin ennalta määritetty alaraja neliökilometreissä

Ehtolausekkeiden suhdearvot sekä alarajat on määritetty etukäteen koodissa, joka perustuvat havaintoihin ohjelman kokeiluvaiheessa. Tavoitteena on, että alueilla, joissa on tiheämmin rakennuksia sekä teitä, ruudukko on tiheämpi.

Eri polygonien päällekkäisyys voidaan laskea suorakulmion pinta-alan sekä käsiteltävän karttadatan pinta-alalla. Karttadata täytyy suodattaa siten, että se sisältää vain sen osan, joka on suorakulmion päällä. Kahden muodon intersektio voidaan toteuttaa GeoPandas-kirjaston funktiolla ”clip()”. GeoPandas-kirjasto toteuttaa metodin pinta-alan laskemiseen, joten karttadatan pinta-ala voidaan jakaa suorakulmion pinta-alalla, jolloin saadaan suhdeluku päällekkäisyydelle. Jotta pinta-ala voidaan laskea, GeoDataFrame, joka sisältää karttadatan täytyy projektoida uudestaan käyttämään geometristä koordinaattijärjestelmää. GeoDataFrame-olioista luodaan kopio, joka projektoidaan uudestaan käyttämään EPSG:3387-tunnuksella olevaa kartastokoordinaattijärjestelmää, ja alueiden pinta-alat tallennetaan erillisiin muuttujiin, joiden avulla suhdeluku voidaan laskea.

Käsitellään funktion yksi rekursio. Funktion parametriksi on annettu suorakulmio polygoni, joka vastaa ruudukon yhtä ruutua.

Kunnan polygoneista, maapolygoneista sekä sisävesistöpolygoneista lasketaan läpileikkaukset käyttäen suorakulmiota läpileikkauksen maskina. Läpileikkausten sekä suorakulmion pinta-alojen suhde lasketaan, jossa suhdeluku 1 vastaa täyttä päällekkäisyyttä, ja suhdeluku 0 tulosta, jossa ei ole päällekkäisyyttä. Seuraavat ehdot on määritetty tilanteelle:

- Maan ja suorakulmion suhdearvo tulee olla suurempi kuin 0,2.

- Sisävesistöjen sekä suorakulmion suhdearvo tulee olla pienempi kuin 0,9.

- Kunnan rajojen ja suorakulmion suhdearvo tulee olla suurempi kuin 0,02.

Kunnan rajat tulee käsitellä tällä tavoin, sillä ruudukko luodaan suorakulmiosta, joka on luotu kunnan rajojen ympärille. Kunnat saattavat olla erittäin pitkiä pituuspiirissä eikä niinkään leveyspiirissä, tai toisinpäin. Tämän myötä suorakul-

mioon saattaa jäädä suuri tila aluetta, joka ei kuulu kunnan rajojen sisäpuolelle. Kuntien muodot saattavat myös sisältää suuria, syviä koveria kohtia, jotka kuuluvat ruudukon sisälle, mutta eivät ole osa kuntaa. Esimerkkinä tällaisesta kunnasta on Joensuu. Samanlaisia tapauksia ovat kunnat kuntien sisällä, kuten Savonlinnassa oleva Enonkoski sekä Espoossa oleva Kauniainen. On myös kuntia, jotka koostuvat useammasta alueesta, kuten Vaasa.

Jos kaikki nämä ehdot ei toteudu, suorakulmion sijasta funktio palauttaa tyhjän listan.

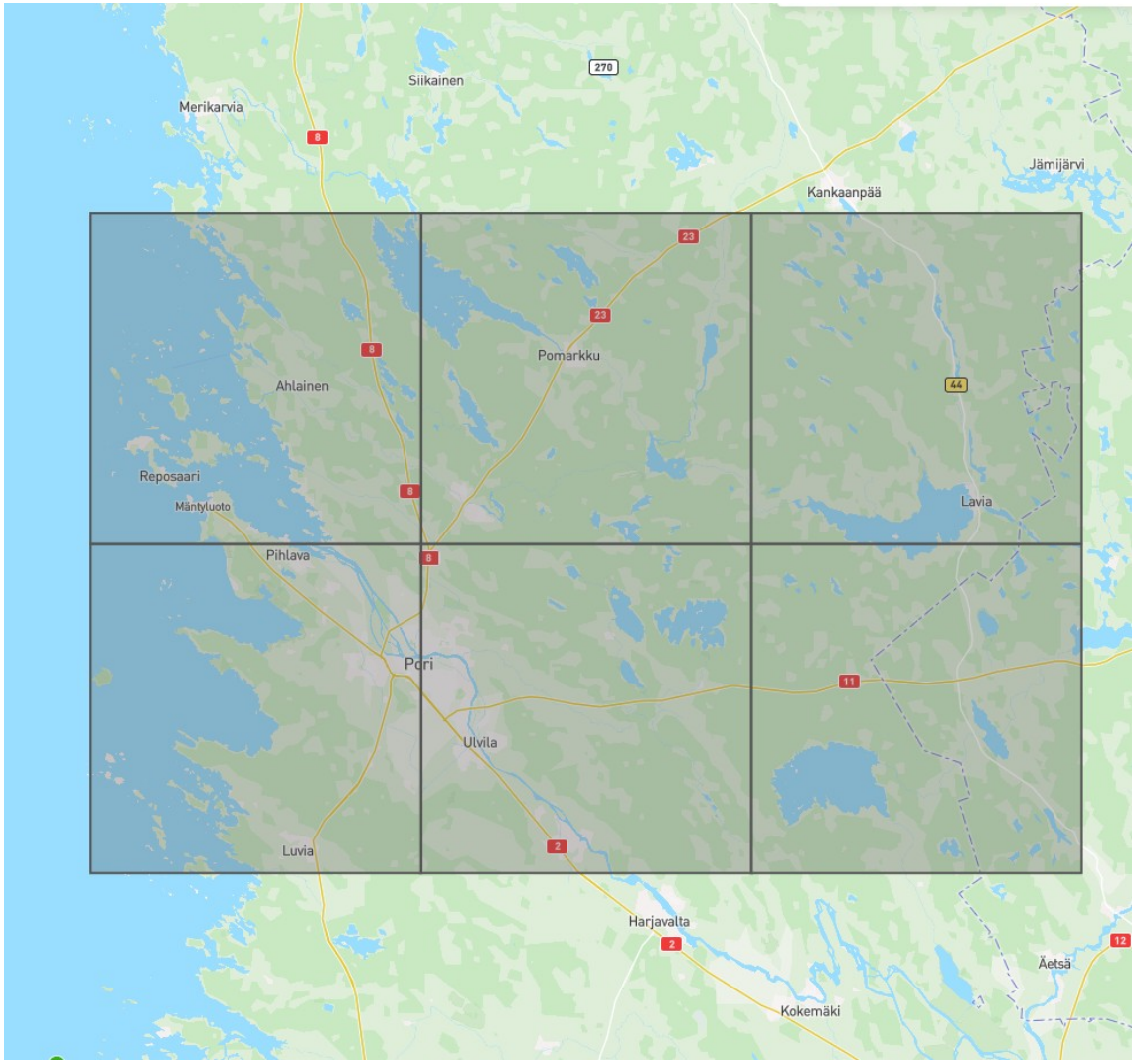
Käsitellään tilanne, jossa suorakulmio palautetaan sellaisenaan, ja rekursio päättyy. Edellä mainittujen ehtolausekkeiden jälkeen käydään kaksi muuta ehtoa läpi:

- Suorakulmion pinta-ala on pienempi kuin 0,1 km².

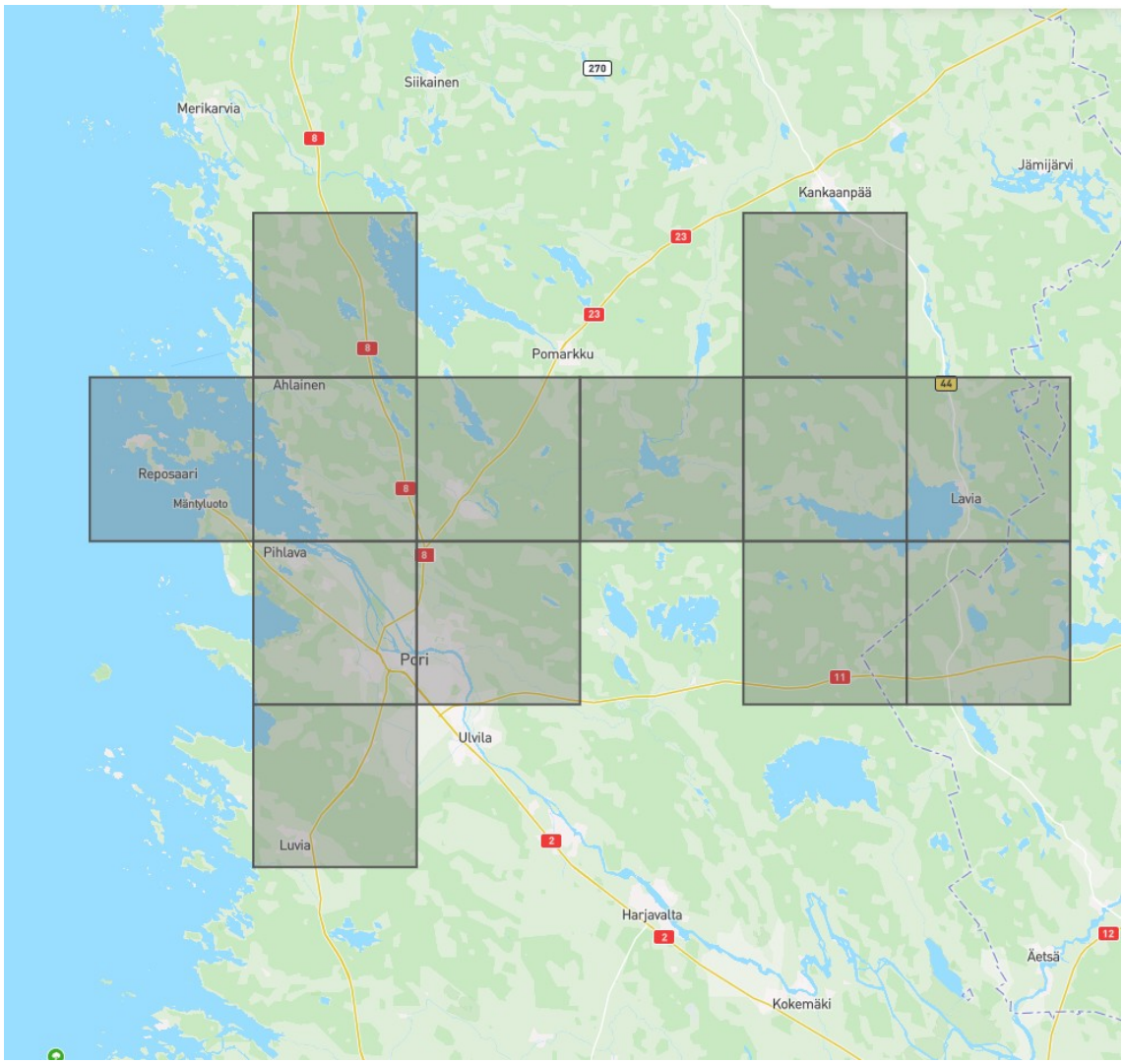
- Rakennusten sekä teiden summa on pienempi kuin 125.

Jos edes toinen näistä ehdoista toteutuu, suorakulmio palautetaan sellaisenaan listassa.

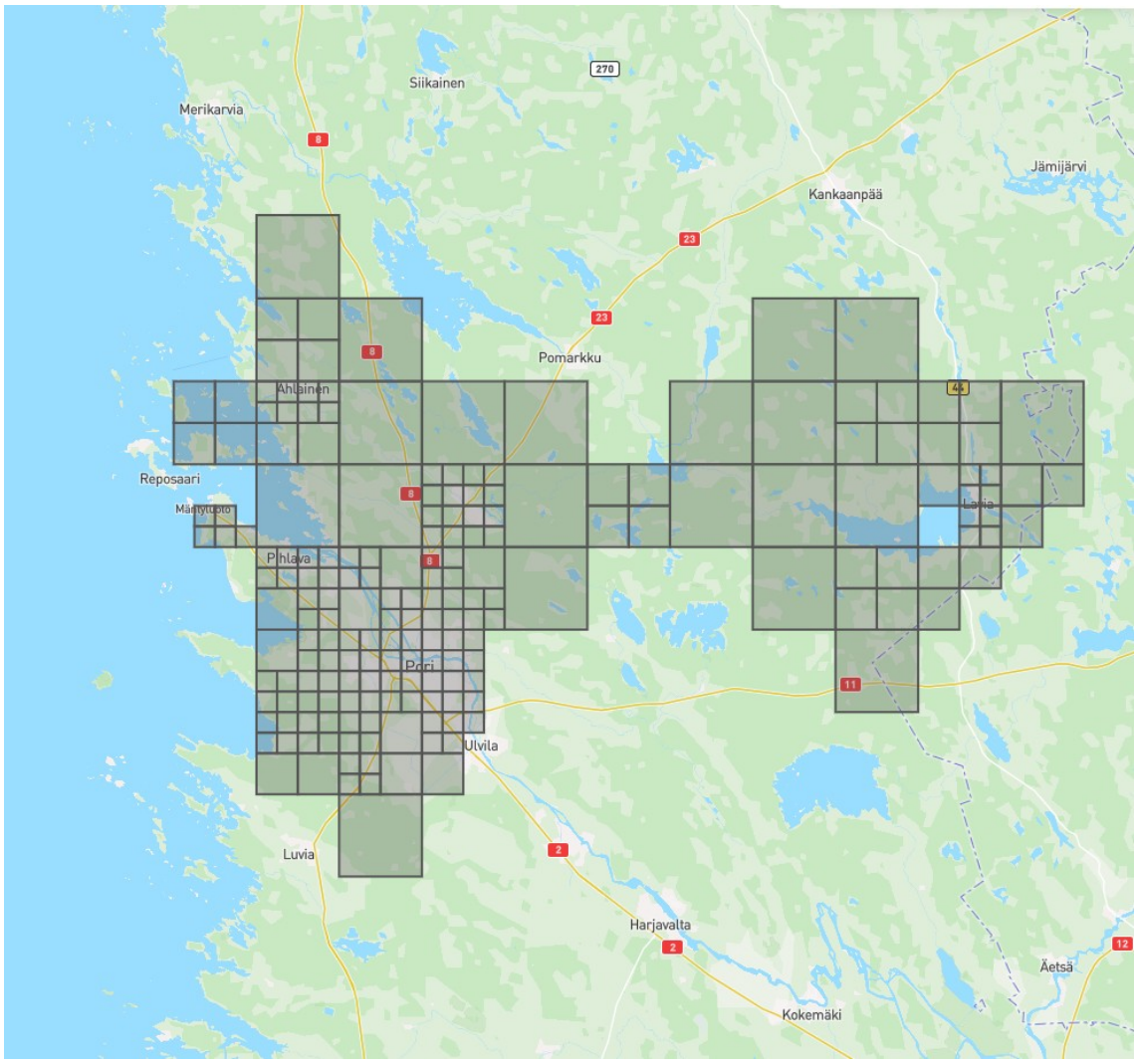
Jos kumpikaan aikaisemmista mainituista tapauksista ei tapahtunut, funktio jakaa suorakulmion neljään osaan ja palauttaa ne listassa. Funktio aina yhdistää rekursion palautusarvot, jolloin koko rekursioketjun päätyttyä saadaan lista, jossa on suorakulmio-objekteja, jotka vastaavat haluttua ruudukkoa.



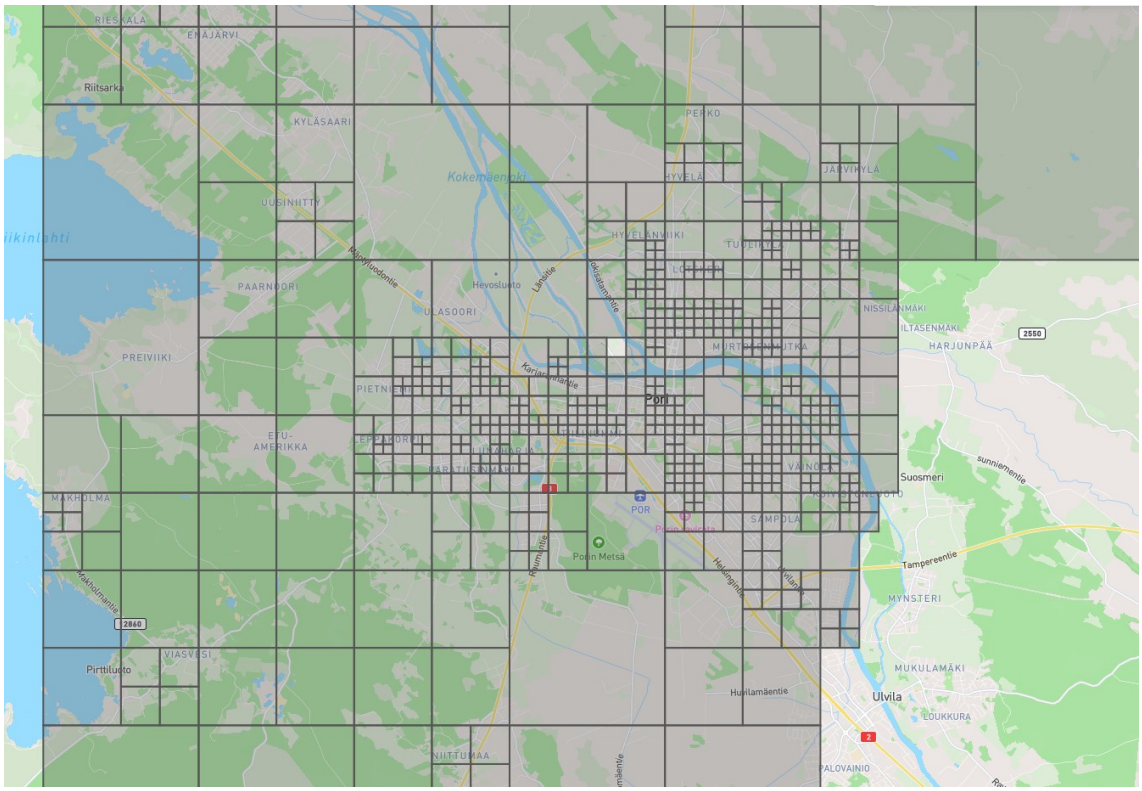
Kuva 6: Ruudukko kahden iteraation jälkeen



Kuva 7: Ruudukko kolmen iteraation jälkeen



Kuva 8: Ruudukko viiden iteraation jälkeen. Ruudut ovat erikokoisia.



Kuva 9: Lopullinen ruudukko. Porin keskusta-alueella ruudukko on tiheämpi

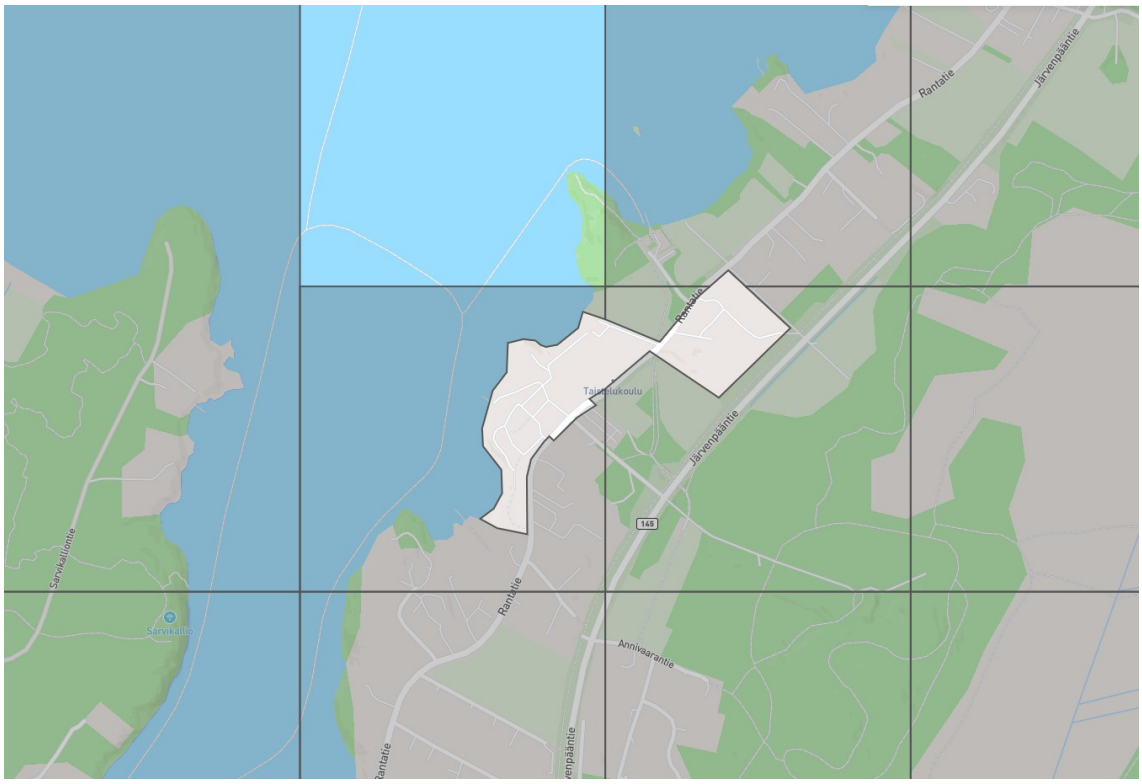
4.5 Kiellettyjen alueiden poistaminen

Sillä Luotu ruudukko on tarkoitettu siihen, että sovelluksen käyttäjä käy ruuduissa fyysisesti paikan päällä, on mielekästä, että alueet, joilla liikkuminen on lainvastaista tai muuten kiellettyä, on poistettu ruudukosta. Tämän voisi toteuttaa samalla tavalla kuin ruutujen poistamisen vesistöistä, mutta käytettävyyden kannalta alueiden tarkka poistaminen on mielekkäämpää.

Kun ruudukko on luotu aikaisemmassa osassa ohjelmaa, pääohjelma tarkastaa, onko aikaisemmin luodussa kiellettyjen alueiden GeoDataFrame-oliossa geometristä dataa. Jos GeoDataFrame-olion ”geometry”-lista on tyhjä, ohjelma ohittaa tämän vaiheen ja jatkaa karttadatan tallentamiseen.

Jos kiellettyjen alueiden GeoDataFrame-olion geometry-lista sisältää dataa, alueet poistetaan ruudukosta. Ohjelma käy läpi kaikki laatikot ja syöttää laatikon sekä polygonit GeoPandas-kirjaston spatiaaliseen funktioon ”overlay()”, joka toteuttaa spatiaalisen operaation päällekkäisten muotojen laskemiseen. Funktiolla on kaksi pakollista parametriä, joiden syötearvoksi funktio odottaa GeoDataFrame-tyypin oliota. Ensimmäinen parametri on data, jota käsitellään. Jotta laatikkoa voidaan käyttää funktiossa, täytyy laatikko-objekti lisätä uuteen GeoDataFrame -olioon, johon on määritetty sama koordinaattijärjestelmä, kun kiellettyjen alueiden oliolla, eli WGS84. Toisen parametrin syötearvo on data, jolla ensimmäisen parametrin dataa käsitellään, eli tähän syötetään kiellettyjen alueiden GeoDataFrame-olio.

Funktioon määritetään vaihtoehtoiseksi parametriksi, mitä metodia muodon laskemiseen käytetään, johon tässä tapauksessa määritetään eroavuus. Eroavuus metodi poistaa kiellettyjen alueiden polygonejen muodot laatikosta, ja palauttaa uuden GeoDataFrame-olion, joka sisältää eroavuuden.



Kuva 10: Tuusulan taistelukoulu on poistettu ruudukosta

Sillä Kaikkien suodatettujen laatikoiden tulee olla samassa GeoDataFrame-oliossa, spatiaalisen funktion palautusarvo yhdistetään muiden suodatettujen laatikko GeoDataFrame-olioiden kanssa. Uusi GeoDataFrame-olio, johon on yhdistetty kaikki suodatetut laatikot, viedään ohjelman seuraavan osaan.

4.6 Datan tallennus

Kun rekursiivisen funktion suoritus loppuu, se palauttaa listan Shapely-polygoneja. Luodaan uusi GeoDataFrame-olio, jonka ’geometry’-parametriksi voidaan määrittää funktion palauttama lista.

GeoDataframe-luokka sisältää metodin, jolla objektin tiedot voidaan tallentaa pyydetyssä tiedostomuodossa. Tässä vaiheessa projektikokonaisuutta tieto tallennetaan GeoJSON-muodossa, jotta tulosten tarkastelu on mielekkäämpää,

mutta ohjelman muuttaminen siten, että tieto tallennetaan esimerkiksi tietokantaan, on mahdollista.

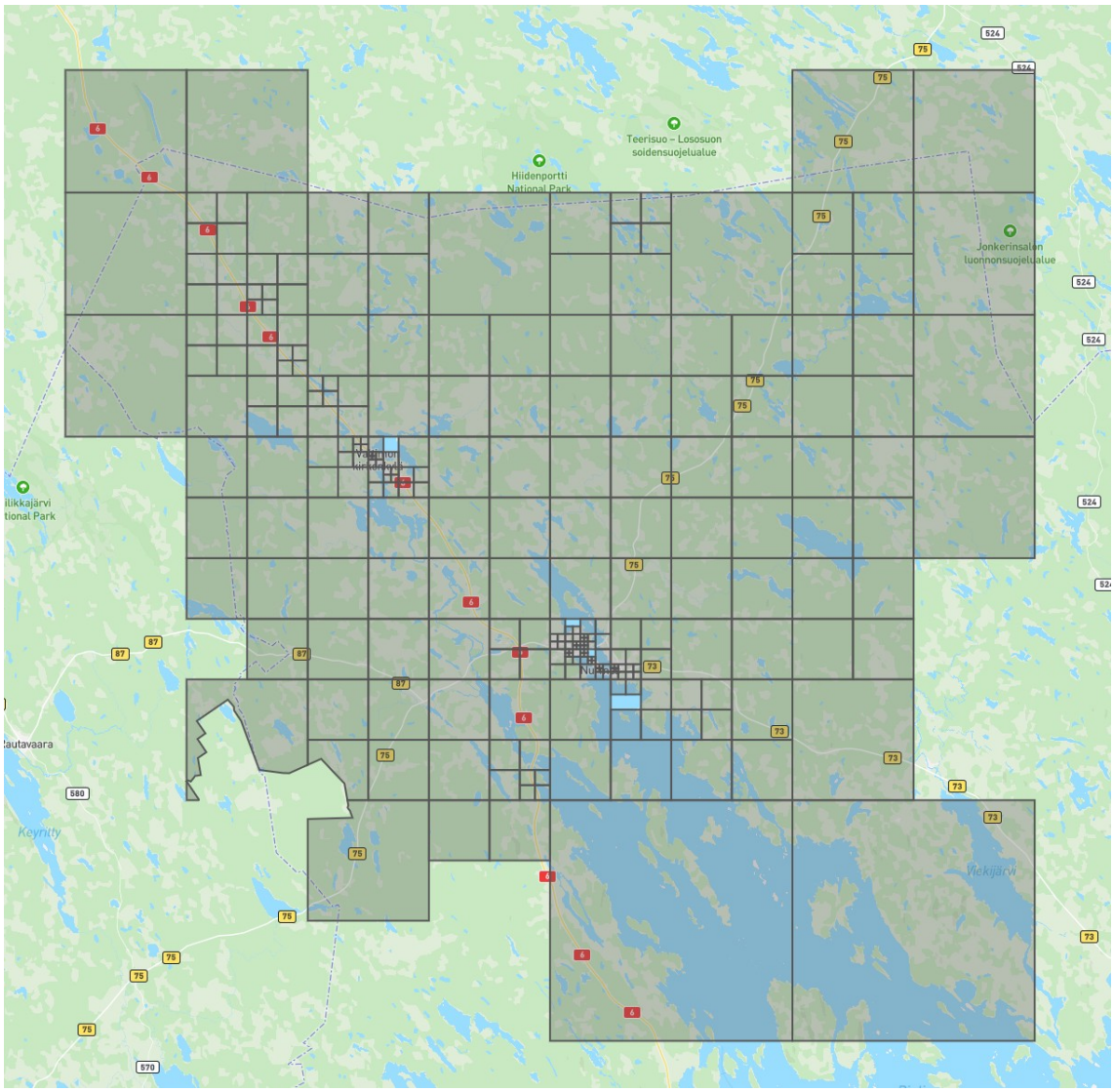
Algoritmin kehityksessä sekä kokeiluvaiheessa tulosten nopea visuaalinen tarkastaminen on tärkeää, sillä ehtolausekkeissa olevien suhdearvojen hienosäätäminen vaatii monia peräkkäisiä ohjelman suorituksia. On myös mielekästä, että tulokset ovat karttaa vasten, ja kartalle voi piirtää myös muita muotoja. Projektin kehityksessä tulosten tarkasteluun käytettiin geojson.io -verkkopalvelua, joka antaa graafisen liittymän tulosten tarkasteluun. Verkkosivun oikeassa laidassa on tekstikenttä, johon voi syöttää mitä tahansa validia GeoJSON-dataa, ja määritetty karttadata piirtyy kartalle saman tien.

5 Tulokset

Aikaisemmassa osiossa tarkasteltiin, miten ruudukko rakentuu Porin ympärille, mutta ohjelman kokeilun myötä on hyvä kokeilla rakentaa ruudukko myös muiden kuntien pohjalta. Ohjelmaan syötetään kolme eri kuntaa: Nurmes, Helsinki sekä Savonlinna, joiden tuloksia tarkastellaan. Ohjelman kannalta kuntien piirteet ovat erilaisia, joten on mielenkiintoista nähdä, miten hyvin ohjelma onnistuu rakentamaan mielekkään ruudukon.

5.1 Esimerkki: Nurmes

Koska Nurmeksen keskusta on pääosin järven ympäröimä, on mielenkiintoista tarkastella, miten ruudukko muodostuu vesistön ympärille.



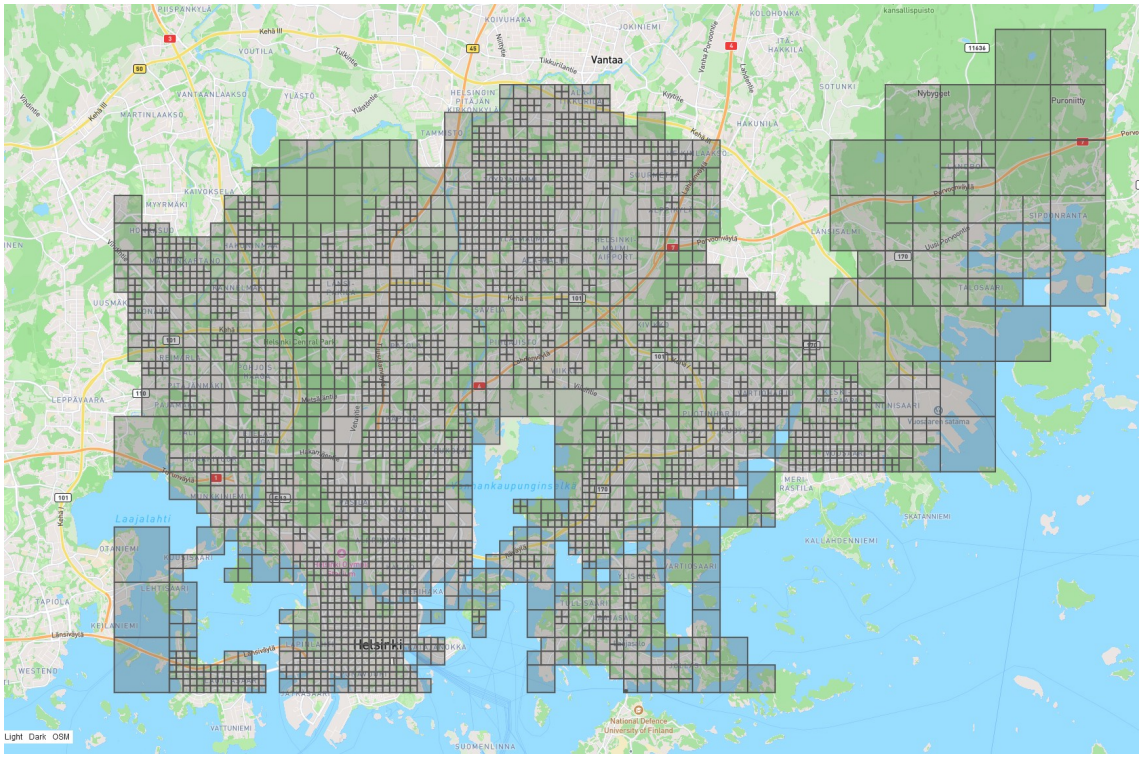
Kuva 11: Nurmeksen kunnan pohjalta rakennettu ruudukko

Osa ruuduista, jotka olisivat pääosin järven päällä, ei ole tallennettu tulokseen. Voidaan myös havaita, että Sotipuron ampuma-alue on poistettu ruudukon vasemmasta alareunasta, sillä se kuuluu armeijan käyttöön. Kuten ohjelmalta voi odottaakkin, kyläkeskuksissa ruudukko on tiheämpi verrattuna ruudukon reu-

noilla oleviin alueisiin, jotka koostuvat pääosin metsistä. Ohjelman ajossa meni 50 sekuntia.

5.2 Esimerkki: Helsinki

Helsinki on hyvä kaupunki ohjelman testaamiseen, sillä se on tiheään rakennettu, ja täten sisältää paljon karttadataa. Helsinki on myös satamakaupunki, joten voidaan myös tarkastella, miten algoritmi ottaa meren rajan huomioon.



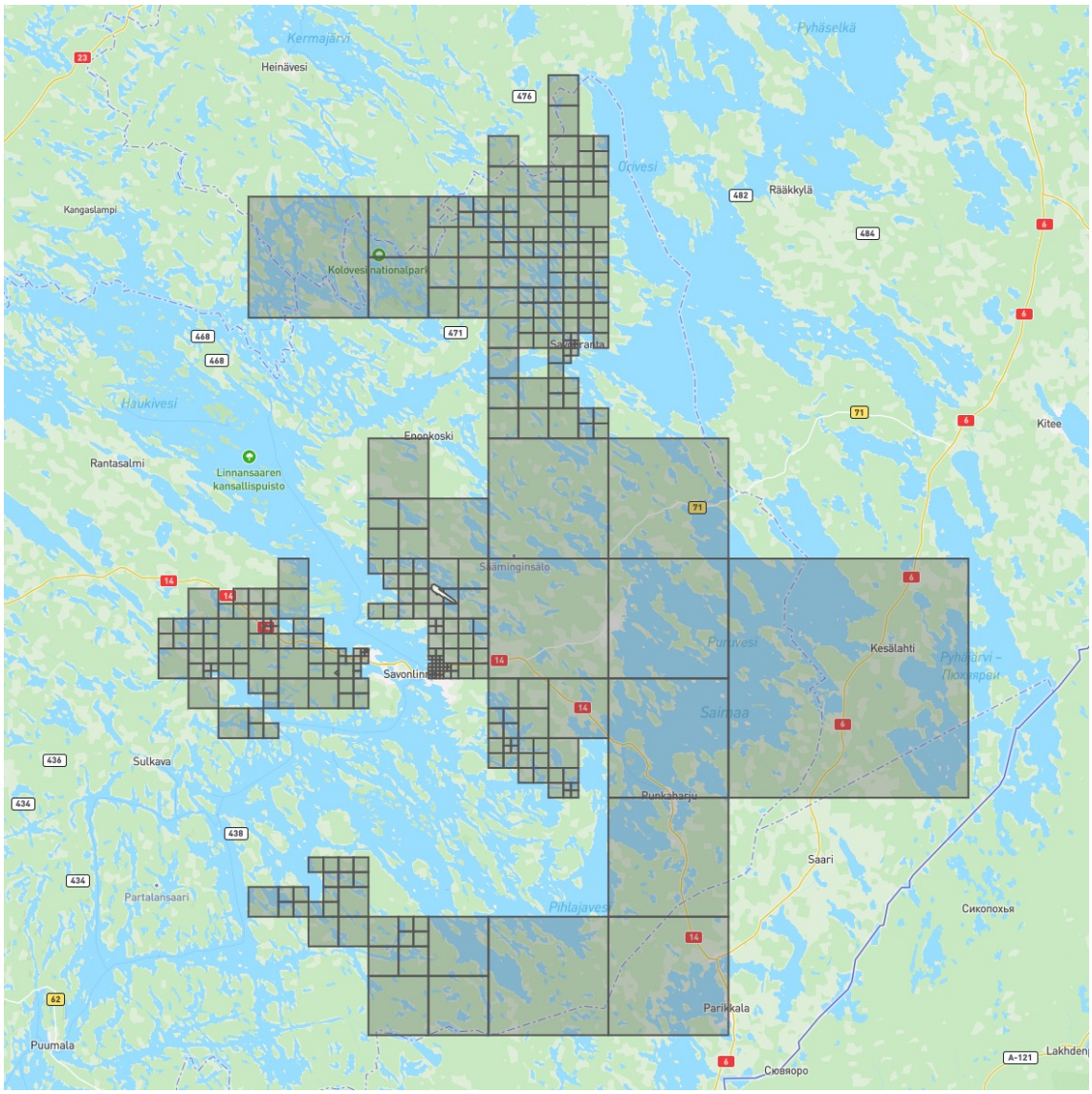
Kuva 12: Helsingin kaupungin pohjalta rakennettu ruudukko

Ruudukko on useissa keskusta-alueissa niin tiheää, kuin algoritmi antaa luoda, ja voidaan havaita, että asuinalueilla sekä puistoalueilla ruudukko on hieman väljempää. Helsingin keskusta-alueella olevista niemistä on ruudut poistettu. Pääesikunnan rakennus Kaartinkaupungissa sekä Santahaminan varuskunnan alueet on poistettu, sillä se luetaan Puolustusvoimien alueeksi. Helsinki-Malmin lentoasemaa ei ole poistettu.

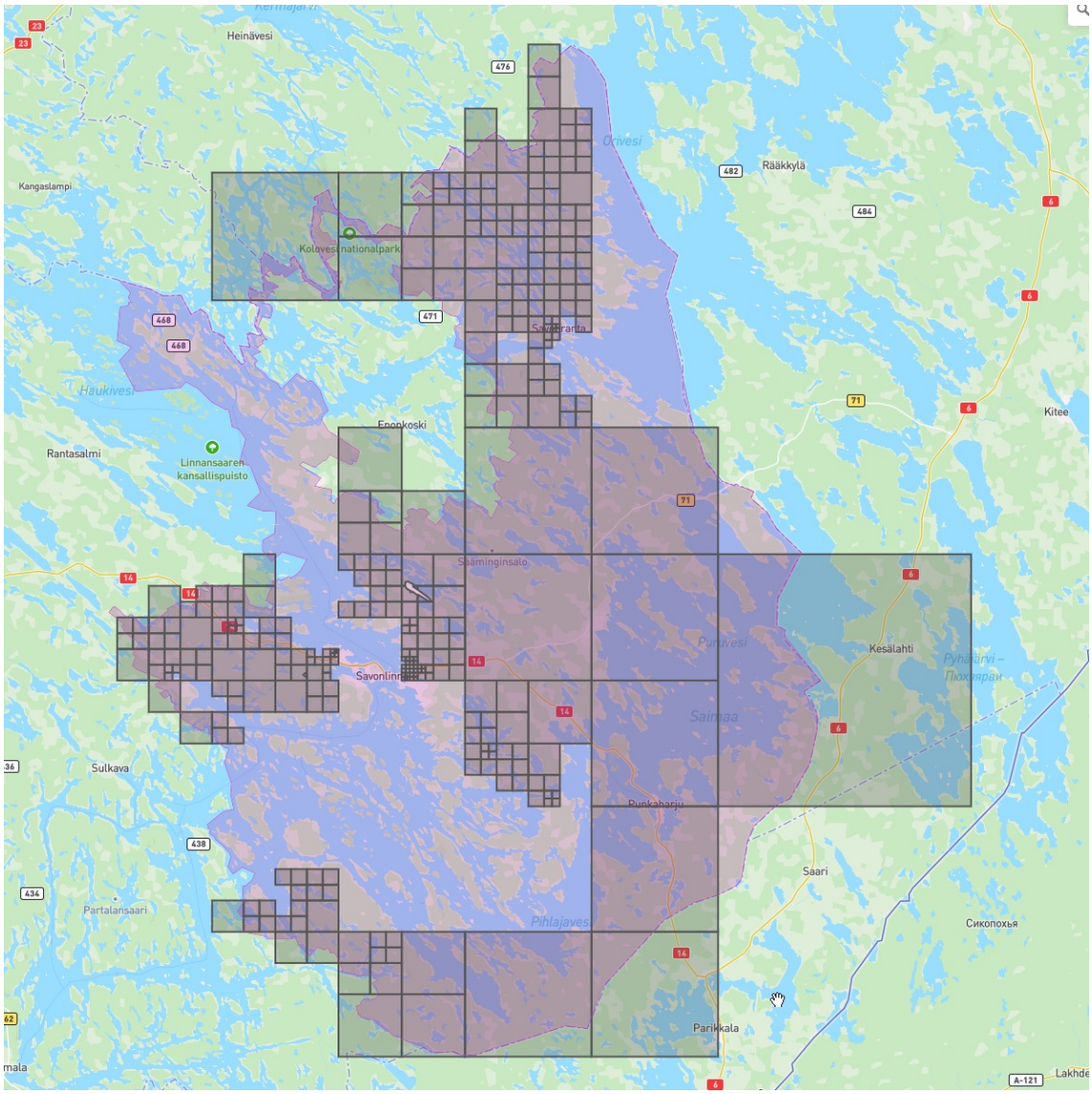
Osassa rantaviivaa lähellä olevista alueista ruudut on poistuneet virheellisesti, kuten Länsisatamassa sekä Vuosaarta ympäröivissä alueissa. Tämän ongelman aiheuttaa ehtolause, joka tarkastaa, kuinka suuri osa ruudusta on merivettä. Sillä Iso ruutu sisältää pääosin merta ja pienen osan maanpintaa, algoritmi virheellisesti poistaa myös tämän pienen osan maanpintaa. Ohjelman ajossa meni 10 minuuttia ja 12 sekuntia.

5.3 Esimerkki: Savonlinna

Savonlinna on Saimaan ympäröivä, vesistöinen kaupunki. Savonlinnan muoto on myös hieman yleisestä poikkeava, jossa kaupungin pohjoisosassa on toinen kunta, Enonkoski. Täten on mielenkiintoista, miten algoritmi rakentaa ruudukon kunnan rajojen ympärille.



Kuva 13: Savonlinnan kaupungin pohjalta rakennettu ruudukko



Kuva 14: Savonlinnan kaupungin pohjalta rakennettu ruudukko (punaisella kaupungin rajat)

Tulosten kannalta Savonlinnan pohjalta rakennettu ruudukko on tuloksista kehnoin. Suurta osaa keskusta-alueesta ei ole sisällytetty ruudukkoon, sillä se on järven ympäröimä, ja aikaisempi ruutu on poistunut. Suuri osa kunnan alueesta ei ole sisällytetty ruudukkoon, sillä alue on runsasvesistöistä. Enonkosken aluetta ei ole sisällytetty ruudukkoon, mutta osa rajalla olevista ruuduista on hieman Enonkosken alueen sisällä. Ohjelman ajossa meni minuutti ja 52 sekuntia.

Tuloksista voidaan havaita, että ruudukot tihenevät infrastruktuurin perusteella oikein, ja ruudukot myötäilevät kuntien rajoja. Alueet, jotka on määritetty poistettavaksi, poistuvat myös oikein ruudukoista. Ruudukot myötäilevät myös rantaviivoja melko hyvin. Ohjelman suoritusaika on myös miellyttävä.

Ohjelman luomissa ruudukoissa on vielä parannettavaa. Ruudukoiden kriittisin ongelma on ruutujen puuttuminen alueilla, joissa niiden olisi hyvä olla. Tämä johtuu siitä, kun ruudukkoa tihennetään, iso osa ruudusta saattaa olla vesistöä, jonka myötä ruutu poistetaan. Tämän myötä myös se pieni osa, joka on maanpintaa, poistuu ruudukosta. Pieni osa saattaa kuitenkin olla merkittävä alue, kuten Helsingin Länsisataman alue.

5.4 Jatkokehitys

Vaikka projektissa on toteutettu toimiva algoritmi, joka toteuttaa suunnittelussa määritetyt määritteet, sen ominaisuuksia tullaan lisäämään sekä parantamaan sen mukaan, kun projektin kokonaisuus kasvaa. Jatkokehitys ei haittaa muuta kehitystä, sillä projektissa luotu ohjelma on rajapinta eikä sen ulostulon formaatti muutu.

Tulokset-osiossa kuvattu ongelma vesistöjen rajojen kanssa tulee korjata, mikä vaatii refaktorointia algoritmiin. Ruudun voi jakaa pienempään osaan, ja tämän jälkeen tarkastaa maan ja vesistön pinta-alojen välisen suhteen. Tämä kuitenkin lisää ohjelman suoritusaikaa, eikä välttämättä ole optimaalisin vaihtoehto.

Sillä Kunnan ruudukko luodaan kunnan koon perusteella. Tällä hetkellä eri kuntien ruudukot ovat erikokoisia, jonka myötä myös ruutujen koko vaihtelee. Tämän voi korjata luomalla asteikon, minkä pohjalta ruudukon alkuperäinen suorakulmio lasketaan. Koska yksittäisen ruudun sivujen pituus aina puolitetaan, asteikko olisi eksponentiaalinen, jossa aikaisempi arvo aina kerrotaan kahdella. Jos alkuperäisen ruudun yhden sivun pituus jää kahden arvon väliin asteikolla, sivujen pituus nostetaan korkeamman arvoksi. Kun tämä tehdään kaikille ruudukoille samalla asteikolla, kaikki suorakulmiot ovat samalla mittasuhteella.

Tällä hetkellä projektissa käsitellään kaikki tiet sekä rakennukset samanarvoisina. Koska Overpass-rajapinnasta saadaan paljon tietoa näistä sisällöistä, erityyppisille teille sekä rakennuksille on mahdollista antaa eri painoarvot. Esimerkiksi isolla kerrostalolla tai merkittävällä museorakennuksella voisi olla enemmän painoarvoa kuin roskakatoksella. Myös isommilla autoteillä voisi olla suurempi painoarvo kuin pienillä kävelypoluilla.

Kiellettyjen alueiden poistaminen ei ole tällä hetkellä erityisen kattavaa, sillä esimerkiksi useat varastoalueet, satama-alueet yms. voi olla myös alueita, joissa vapaa liikkuminen on rajoitettua. OpenStreetMap-tietokannassa on paljon eri alueluokitteluja ja on mahdollista, että tietokannasta löytyy luokitteluja, jotka soveltuvat hyvin tähän tarkoitukseen.

Jos sovelluskokonaisuutta on tulevaisuudessa tarkoitus laajentaa muihin valtioihin Suomen lisäksi, projektia täytyy kokeilla uudestaan riippuen kunnan tiheydestä ja datan saatavuudesta. Tämän lisäksi voi olla mahdollista, että projektiin pitää tehdä muutoksia. Koska metrimittauksia tehdään käyttäen Suomeen tarkoitetulla kartastokoordinaattijärjestelmällä, täytyy tämä vaihtaa sopivampaan koordinaattijärjestelmään riippuen alueesta, jota käsitellään.

Joissain valtioissa voi olla useita samannimisiä kuntia tai kaupunkeja, jota ei oteta huomioon projektin toteutuksessa. OpenStreetMap-tietokantaa ei päivitetä yhtä usein kaikissa valtioissa kuin Suomessa, jonka myötä Overpass-rajapinnasta saatava karttadata ei ole välttämättä ajan tasalla. [36.]

6 Yhteenveto

Opinnäytetyössä on perehdytty karttadatan käyttöön sovelluskehityksessä, ja avoimiin työkaluihin sekä palveluihin, jotka edistävät karttadatan hakemista sekä käsittelyä. Opinnäytetyössä perehdyttiin myös yleisiin ongelmiin, käsitteisiin sekä konsepteihin, joita karttojen kanssa työskentelyssä saattaa kohdata.

Projektissa on luotu käytettävä rajapinta, joka tuottaa määrittelyn sekä käytön kannalta miellyttävän tuloksen. Sovellus hakee lähes kaiken karttadatan, suodattaa datan ja tallentaa tuloksen automaattisesti, ja käyttäjän tarvitsee vain syöttää käynnistyskomento komentoikkunaan.

Algoritmi ei kuitenkaan ole vielä valmis. Algoritmi on tuottanut käytettävän ruudukon kaikilla kokeilukerroilla, mutta useissa on huomattavissa jonkin verran puutteita sekä virheitä. Sovellus vaatii myös refaktorointia, jos kokonaisprojektin laajuutta tullaan laajentamaan maailmanlaajuiseksi.

OpenStreetMap sekä Overpass-rajapinta osoittautuivat projektin aikana erittäin hyödyllisiksi sekä kattaviksi työkaluiksi. OpenStreetMap sisältää erittäin paljon avoimesti käytettävää karttadataa, ja Overpass-rajapinta, vaikkakin sen käyttämä kyselykieli on monimutkainen, on erittäin tehokas ja mahdollistaa datan suodattamisen erittäin yksityiskohtaisesti. Projektissa käytetty Geopandas-kirjasto on tehokas työkalu karttadatan käsittelyyn. Vaikka se on tarkoitettu enemmän datan analysointiin kuin manipulointiin, se täytti kaikki projektin asettamat tavoitteet.

Lähteet

1 How Many People Own Smartphones (2023-2028). 2023. Verkkoaineisto. Josh Howarth. https://explodingtopics.com/blog/smartphone-stats. Luettu 24.8.2023

2 Which Apps Actually Need Location Permissions?. 2020. Verkkoaineisto. Dashlane. https://www.dashlane.com/blog/which-apps-actually-need-location-permissions. Luettu 24.8.2023.

3 Pokémon GO Hit 50 Million Downloads in Record Time, Now at More Than 75 Million Worldwide. 2016. Verkkoaineisto. Randy Nelson. https://sensortower.com/blog/pokemon-go-50-million-downloads. Luettu 24.8.2023.

4 ESRI Shapefile. 2020. Verkkoaineisto. Library Of Congress. https://www.loc.gov/preservation/digital/formats/fdd/fdd000280.shtml. Luettu 14.9.2023.

5 RFC 7946 - The GeoJSON Format. 2016. Verkkoaineisto. H. Butler, M. Daly, A. Doyle, Sean Gillies, T. Schaub, Stefan Hagen. https://datatracker.ietf.org/doc/html/rfc7946. Luettu 14.9.2023.

6 Shapefile vs. GeoJSON vs. GeoPackage. 2019. Verkkoaineisto. Juho Häme. https://feed.terramonitor.com/shapefile-vs-geopackage-vs-geojson/. Luettu 14.9.2023.

7 About OpenStreetMap. 2023. Verkkoaineisto. OpenStreetMap community. https://wiki.openstreetmap.org/wiki/About_OpenStreetMap. Luettu 14.9.2023.

8 Overpass API. 2023. Verkkoaineisto. OpenStreetMap community. https://wiki.openstreetmap.org/wiki/Overpass_API. Luettu 14.9.2023.

9 Overpass API/Overpass QL. 2023. Verkkoaineisto. OpenStreetMap community. https://wiki.openstreetmap.org/wiki/Overpass_API/Overpass_QL. Luettu 14.9.2023.

10 The Data Model of OpenStreetMap. 2023. Verkkoaineisto. Roland Olbricht. https://dev.overpass-api.de/overpass-doc/en/preface/osm_data_model.html. Luettu 14.9.2023.

11 The Query Statement. 2023. Verkkoaineisto. OpenStreetMap community. https://wiki.openstreetmap.org/wiki/Overpass_API/Overpass_QL#The_Query_Statement. Luettu 14.9.2023.

12 The Query Filter. 2023. Verkkoaineisto. OpenStreetMap community. https://wiki.openstreetmap.org/wiki/Overpass_API/Overpass_QL#The_Query_Filter. Luettu 14.9.2023.

13 Output format (out:). 2023. Verkkoaineisto. OpenStreetMap community. https://wiki.openstreetmap.org/wiki/Overpass_API/Overpass_QL#Output_format_(out:). Luettu 14.9.2023.

14 Overpass turbo. 2023. Verkkoainesto. OpenStreetMap community. https://wiki.openstreetmap.org/wiki/Overpass_turbo. Luettu 14.9.2023.

15 Overpass turbo/GeoJSON. 2023. Verkkoaineisto. OpenStreetMap community. https://wiki.openstreetmap.org/wiki/Overpass_turbo/GeoJSON. Luettu 14.9.2023.

16 Stats. 2023. Verkkoaineisto. OpenStreetMap community. https://wiki.openstreetmap.org/wiki/Stats. Luettu 14.9.2023.

17 About 99boundaries. 2023. Verkkoaineisto. Timothy Ellersiek. https://99boundaries.com/about. Luettu 14.9.2023.

18 99boundaries. 2023. Verkkoaineisto. Timoythy Ellersiek. https://github.com/TimMcCauley/nintynine-boundaries/tree/4ea7a61418fd92c97a59be5d8b0fd82618fc19c9. Luettu 14.9.2023.

19 Stack Overflow Developer Survey 2023. 2023. Verkkoaineisto. Stack Overflow. https://survey.stackoverflow.co/2023/. Luettu 14.9.2023.

20 About Python. 2023. Verkkoainesto. Python Institute. https://pythoninstitute.org/about-python. Luettu 14.9.2023.

21 How does Python Work?. 2020. Verkkoaineisto. Dhruvil Karani. https://towardsdatascience.com/how-does-python-work-6f21fd197888. Luettu 14.9.2023.

22 Package overview — pandas 2.1.0. 2023. Verkkoaineisto. Pandas. https://pandas.pydata.org/docs/getting_started/overview.html. Luettu 14.9.2023.

23 Introduction to GeoPandas. 2023. Verkkoaineisto. Geopandas. https://geopandas.org/en/stable/getting_started/introduction.html. Luettu 14.9.2023.

24 Projections. 2023. Verkkoaineisto. Geopandas. https://geopandas.org/en/stable/docs/user_guide/projections.html. Luettu 14.9.2023.

25 The Shapely User Manual — Shapely 2.0.1 documentation. 2023. Verkkoaineisto. Sean Gillies. https://shapely.readthedocs.io/en/stable/manual.html. Luettu 14.9.2023.

26 Types of Map Projections. 2023. Verkkoaineisto. Elizabeth Borneman. https://www.geographyrealm.com/types-map-projections/. Luettu 14.9.2023.

27 Mercator projection. 2023. Verkkoaineisto. The Editors of Encyclopaedia Britannica. https://www.britannica.com/science/Mercator-projection. Luettu 1.9.2023.

28 Mercator. 2023. Verkkoaineisto. OpenStreetMap community. https://wiki.openstreetmap.org/wiki/Mercator. Luettu 1.9.2023.

29 Geographic Coordinate Systems. 2022. Verkkoaineisto. Caitlin Dempsey. https://www.gislounge.com/geographic-coordinate-system/. Luettu 1.9.2023.

30 KKJ / Finland zone 5 – EPSG:3387. 2020. MapTiler Team. https://epsg.io/3387. Luettu 1.9.2023.

31 EPSG Geodetic Parameter Dataset. 2023. Verkkoaineisto. Geomatic Solutions. https://epsg.org/home.html. Luettu 1.9.2023.

32 Public Overpass API instances. 2023. Verkkoaineisto. OpenStreetMap community. https://wiki.openstreetmap.org/wiki/Overpass_API#Public_Overpass_API_instances. Luettu 14.9.2023.

33 Key:building. 2023. Verkkoaineisto. OpenStreetMap community. https://wiki.openstreetmap.org/wiki/Key:building. Luettu 14.9.2023.

34 Tag:landuse=military. 2023. Verkkoaineisto. OpenStreetMap community. https://wiki.openstreetmap.org/wiki/Tag:landuse%3Dmilitary. Luettu 14.9.2023.

35 Tag:aeroway=aerodrome. 2023. Verkkoaineisto. OpenStreetMap community. https://wiki.openstreetmap.org/wiki/Tag:aeroway%3Daerodrome. Luettu 14.9.2023.

36 OSMstats - Statistics of the free wiki world map. 2023. Verkkoaineisto. Pascal Neis. https://osmstats.neis-one.org/?item=countries. Luettu 14.9.2023.