Osaamista
ja oivallusta
tulevaisuuden
tekemiseen

Tommi Lehtisaari

Vektoritiilien julkaisu Vantaan karttarajapinnassa

Metropolia Ammattikorkeakoulu
Insinööri (AMK)
Maanmittaustekniikka
Insinöörityö
11.4.2019

Tekijä Otsikko Sivumäärä Aika	Tommi Lehtisaari Vektoritiilien julkaisu Vantaan karttarajapinnassa 52 sivua + 3 liitettä 11.4.2019
Tutkinto	insinööri (AMK)
Tutkinto-ohjelma	maanmittaustekniikka
Ohjaajat	osaamisaluepäällikkö Jussi Laari kaupunkimittausinsinööri Arsi Juote

Vektoritiilet ovat tämän hetken paikkatiedon julkaisemisessa käytetyistä teknologioista yksi nopeinten kehittyvä tekniikka. Vektoritiilet perustuvat esilaskettuihin datatiileihin, jotka ovat osia alkuperäisestä lähdemateriaalista. Tällä pyritään parantamaan järjestelmän suorituskykyä. Samaa menetelmää käytetään myös rasterikuvien kanssa, jolloin keskeisenä erona on kuvan muodostuksen ajankohta. Rasteritiilet ovat palvelimelle esilaskettuja kuvatiedostoja, kun taas vektoritiilet muodostetaan kuviksi vasta asiakasohjelmassa.

Vektoritiilitekniikalla julkaistuilla karttapalveluilla on useita etuja järjestelmän skaalautuvuuden ja ylläpidon suhteen verrattuna rasteritiilitekniikalla toteutettuun karttapalveluun. Yleisen olettamuksen mukaan myös keskiverto vektoritiili on pienempi kuin keskiverto rasteritiili, jonka pitäisi teoriassa johtaa pienempään verkkoyhteyden kuormitukseen ja täten lyhyempiin latausaikoihin.

Tutkielmassa syvennytään karttarajapintoihin ja pohditaan vektoritiilien roolia niissä. Empiirisessä osiossa tutkitaan Vantaan kaupungilla käytössä olevilla järjestelmillä vektoritiilien vertautuvuutta vastaavaan rasteritiilitoteutukseen. Lopputuloksena on kaksi testitapausta, joissa mitataan vasteaikoja karttapalveluista, joista toinen hyödyntää vektoritiiliä ja toinen rasteritiiliä. Mittaukset suoritettiin kummankin tekniikan osalta asiakasohjelmassa sekä palvelimen puolella. Oletuksen mukaan karttakuvan muodostaminen asiakasohjelmassa vie vektoritiiltä hyödyntävällä tekniikalla kauemmin kuin rasteritiileltä, koska kuvaa ei ole vielä muodostettu valmiiksi. Vektoritiilet ovat kuitenkin kevyempiä siirtää tietoliikenneverkossa, jonka täytyisi korvata kuvan muodostamiseen käytetty aika saavuttaen pienempi vasteaika kokonaisuudessaan kuin rasteritiilien avulla tehdyssä toteutuksessa.

Avainsanat	Vantaa, Mapbox, vektoritiili, Geoserver

Author	Tommi Lehtisaari
Title	Hosting Vector Tiles on Vantaa Web Service
Number of Pages	52 pages + 3 appendices
Date	11 April 2019
Degree	Bachelor of Engineering
Degree Programme	Land Surveying
Instructors	Jussi Laari, Head of Department Arsi Juote, City Survey Engineer

This final year project aimed at testing the general assumption that vector tiles should be an answer to better performing and more scalable web maps. Furthermore, the project aimed at finding the benefits of using vector tiles over raster tiles.

To reach the aim, tests were performed both on the server side and on the client side. Two identical sets of raster tiles and vector tiles were pre-cached to a server and hosted for further client testing. The client-side response times were measured and compared to find the differences in performance.

The test results indicated that vector tiles did not perform better than raster tiles. However, this was not due to the amount of client-side rendering, which is larger for vector tiles than for raster tiles, but because of the size of the vector tiles on larger scales. This was a rather unexpected result, which leads to the conclusion that raster tiles are more forgiving on performance if the geometry is not generalized carefully. Vector tiles should be generalized more carefully on larger scales and it might be necessary to use different generalization rules than for raster tiles.

Keywords	Mapbox, vector tiles, Geoserver

Sisällys

Lyhenteet

1 Johdanto 1

2 Karttarajapinnat 3

2.1 Web Map Service (WMS) 3
2.2 Kyselyt karttarajapinnassa 4
2.2.1 Palvelun metatietojen kysely (GetCapabilities) 5
2.2.2 Karttakyselyt (GetMap) 5
2.2.3 Kohteen lisätietojen kysely (GetFeatureInfo) 6
2.3 Tiilitystä tukevat karttapalvelut 7
2.3.1 Mittakaavatasot tiilitystä tukevissa karttapalveluissa 7
2.3.2 Karttakyselyt tiilitetyissä kartoissa 10
2.4 Vektoritiilet yleisesti karttapalveluissa 11

3 Mapbox-vektoritiili 11

3.1 Tiedostoformaatti 12
3.2 Geometrian koodaaminen 13
3.2.1 Geometrian komentotyypit 13
3.2.2 Geometriatyypit 14
3.3 Attribuuttien koodaaminen 17

4 Geoserver-vektoritiilitoteutus 18

4.1 Tiilitystä tukevat karttapalvelut Geoserver-ohjelmistolla 18
4.2 Vektoritiilet Geoserver-ohjelmistolla 19

5 Suorituskykytestaus 21

5.1 Tutkimuksen toteuttaminen 21
5.2 Vektori- ja rasteripyramidien julkaisu 22
5.3 Testisovellusten rakentaminen suorituskyvyn vertailemiseksi 25
5.3.1 Openlayers-testisovellus 26
5.3.2 Leaflet-testisovellus 29
5.4 Suorituskykyvertailu asiakasohjelmassa 31
5.5 Palvelimen suorituskyvyn vertaileminen 33

6 Päätelmät 38

6.1 Vektori- ja rasteritiilien vertautuvuus 38
6.2 Suorituskyvyn parantaminen 39
6.3 Vektoritiileille ainutlaatuiset ominaisuudet 41
6.4 Vektoritiilet käytännössä 42

7 Vektoritiilien käytön yleistyminen 43

7.1 Useita vektoritiilitoteutuksia 43
7.2 Vektoritiilien standardointi 44
7.3 Vektoritiilien tulevaisuus 45
7.4 Vektoritiilet Matissa 45
7.5 Paikkatiedon yhteiskäyttö vektoritiileillä 46

Lähteet 48

Liitteet

Liite 1. Openlayers-testisovelluksen lähdekoodi

Liite 2. Asiakasohjelmien suorituskykymittaus

Liite 3. JMeter-mittaustulokset

Lyhenteet

KVP	*Key Value Pairs* suom. Avainarvoparit. Yksi käyttökohde on URL-kyselyissä, jossa attribuuttia esittävälle avaimelle voidaan antaa arvo muuttamatta alkuperäistä tietorakennetta
OGC	*Open Geospatial Consortium.* Kansainvälinen paikkatietoa koskevien standardeja ylläpitävä järjestö.
REST	*Representational state transfer.* Ohjelmistoarkkitehtuuri, jota käytetään yleisesti web-palveluissa.
TMS	*Tile Map Service.* Open Source Geospatial Foundationin rajapintastandardi karttatiilien hakemiseen REST-lähestymistavalla.
VCT	Yleinen lyhenne vektoritiileille.
WMS	Web Map Service. OGC:n määrittelemä standardi siitä, kuinka karttarajapinnasta voidaan pyytää kuvia rajatulta alueelta halutussa mittakaavassa.
WMTS	Web Map Tile Service. OGC:n määrittelemä standardi tiilitystä tukeville karttarajapinnoille.

1 Johdanto

Vektoritiiliä (Vector Tiles) käytetään moderneissa web-kartoissa, joissa järjestelmän tehokkuutta, ylläpitoa ja dynaamisuutta pyritään optimoimaan. Menetelmässä esilasketaan vektorimuotoinen lähdemateriaali pienempiin osiin ja siitä käytetään asiakasohjelmassa vain niitä osia, joita tarvitaan toivotun alueen karttakuvan esittämiseen. Vektoritiilejä hyödyntävissä karttapalveluissa on tarkoitus siirtää spatiaalista tieto siten, että alkuperäinen geometria säilyy vektorimuodossa siirrettäessä sitä tietoliikenneverkossa. Lopullinen karttakuva muodostetaan vasta asiakasohjelmassa. Vastaavan kaltaisia menetelmiä käytetään mahdollisesti monissa kartta- ja paikannuspalveluita tarjoavissa suljetuissa järjestelmissä, mutta vektoritiileistä on julkaistu myös avoin spesifikaatio ”vector-tile-spec”. (Mapbox – vector-tile-spec 2014.) Tämän avoimen spesifikaation on julkaissut yhdysvaltalainen paikkatietoihin erikoistunut yritys Mapbox, jonka juuret ovat avoimien järjestelmien parissa (Springmeyer 2015). Mapboxin vektoritiilispesifikaatio on julkaistu Creative Commons -lisenssoituna. Käytännössä vektoritiili on sarjallistamista, jossa voidaan koodata vektorimuotoista spatiaalista tietoa binäärimuotoon ja siirtää tehokkaasti tietoliikenneverkon välityksellä. Motiivina vektoritiilien kehityksessä ja spesifikaation julkaisemisessa on ollut Mapboxin kehittämä Javascript-kirjasto Mapbox-GL-JS, joka on tarkoitettu web-karttojen esittämiseen selaimessa hyödyntäen laitteistoriippumatonta WebGL-ohjelmointirajapintaa.

Markkinoille on ilmestynyt useita järjestelmiä, jotka hyödyntävät jollakin tavalla tiilitettyjä vektoreita. OGC on tutkinut useita näistä markkinoilla esiintyviä vektoritiilitoteutuksia insinööriraportissaan, jonka tarkoituksena on toimia esitutkimuksena mahdolliselle OGC-vektoritiilistandardille (Cavazzi 2017). Yksi tämän tutkimuksen vektoritiilitoteutuksista oli avoimen lähdekoodin palvelinohjelmisto Geoserver, jonka kehitystä tukee Open Source Geospatial Foundation. Geoserver on myös Vantaan kaupungilla aktiivisessa käytössä niin ulko-, kuin myös sisäverkossa. Geoserver mahdollistaa paikkatiedon julkaisemisen hyödyntäen OGC:n rajapintastandardeja. Vektoritiilet ovat tarkoitettu nimensä mukaan käytettäväksi tiilitystä tukevissa karttarajapinnoissa, joista OGC:n standardoima versio on WMTS-palvelu (Web Map Tile Servise). Geoserver tukee vektoritiilejä WMTS-palvelun lisäosana.

Vantaan kaupunki julkaisee useita paikkatietoaineistoja sekä WMS että WMTS-rajapinnassa Geoserver-palvelinohjelmistolla. WMTS-palvelun haasteena on tällä hetkellä se, että saavuttaakseen paras mahdollinen suorituskyky WMTS-rajapinnassa julkaistavat rasterikuvat täytyy muodostaa etukäteen palvelimelle, joka saattaa viedä aineistosta riippuen paljon aikaa ja levytilaa. Tämä tuottaa erityisesti ylläpidollisia haasteita aineistoille, jossa tiedon täytyy olla mahdollisimman ajankohtaista. Tämän insinöörityön tutkimuksen kohteena on kaupungin johtoverkon julkaisua Geoserver-palvelinohjelmistolla vektoritiileinä sekä tutkia sitä, miten niitä voitaisiin hyödyntää kaupungin sisäisissä ja ulkoisissa asiakasohjelmissa. Tavoitteena on tutkia sitä, miten vektoritiileillä julkaistu johtoverkko vertautuu vastaavaan rasteritiileillä julkaistuun johtoverkkoon. Saavutetaanko vektoritiileillä sama lopputulos kuin rasteritiileillä ja onko niillä mahdollisesti joitain ylläpidollisia tai suorituskyvyllisiä etuja rasterikuviin nähden?

Tämän insinöörityön empiirinen osio keskittyy tutkimaan erityisesti vektoritiilien suorituskykyä verrattuna rasteritiileihin. Tätä tutkitaan vertaamalla vasteaikaa, joka kestää ennen, kuin karttapalvelu palauttaa keskimäärin vektoritiilin verrattuna rasteritiileen. Geoserver tarjoaa vektoritiileille kolme formaattia: Geojson, Topojson sekä MapBox-vector-tile (MVT). Tämän projektin alussa päätettiin tutkia Mapboxin spesifikaatioon perustuvaa formaattia, sillä se on Geoserverin suositus ja pienen tiedostokokonsa ansiosta sen pitäisi vaatia vähemmän kaistaa ja johtaa nopeampiin latausaikoihin tietoliikenneverkossa. (Geoserver 2018.)

Konstruktiivisessa osiossa perehdytään karttarajapintojen toimintaa sekä Mapboxin vektoritiilispesifikaatioon. Konstruktiivisessa osiossa tarkoituksena on keskittyä Vantaan kaupungilla käytössä oleviin avoimen standardin karttarajapintoihin, jotta voidaan muodostaa käsitys siitä, miten vektoritiilien toimintaperiaate vertautuu muihin käytössä oleviin karttarajapintoihin. Empiirisessä osiossa vertaillaan vektori- ja rasteritiilien suorituskykyä muodostamalla kaksi testitapausta. Ensimmäisessä vertaillaan karttapalvelun suorituskykyä käytettäessä kumpaakin formaattia. Toisessa testitapauksessa rakennetaan testisovellus Vantaalla entuudestaan käytössä olevilla asiakasohjelmilla. Tämän jälkeen vertaillaan asiakasohjelmassa vasteaikaa karttanäkymien muodostamisessa.

2 Karttarajapinnat

Karttapalveluiden tarjoaminen tietoverkossa on muodostunut keskeiseksi tavaksi hyödyntää ja jakaa paikkatietoja (JHS 180). Web-kartat mahdollistavat paikkatiedon esittämisen laitteistoriippumattomasti ja ne tarjoavat käyttäjän näkökulmasta kartan, jossa on dynaamisesti muutettava mittakaava sekä mahdollisuus vierittää karttaa halutulla alueella. Web-karttojen yhteydessä tarjotaan usein myös muita palveluita, kuten paikannus- ja reitityspalveluita. Paikkatiedon julkaisemisessa ollaan siirrytty perinteisistä suljetuista järjestelmistä avoimiin standardeihin perustuviin julkaisumenetelmiin. Standardointi paikkatiedon julkaisemisessa tietopalvelurajapinnoissa alkoi OGC:n toimesta 1990-luvulla. Tietopalvelurajapinnan toiminta perustuu siihen, että on olemassa määritelmä siitä, miten palvelimen ja asiakasohjelman tulee vaihtaa tietoa keskenään. Käytettäessä avoimia paikkatietostandardeja on niihin valmiiksi määritelty se, missä muodossa palvelimen tulee vastaanottaa kyselyitä ja missä muodossa se antaa vastauksen.

Suomessa paikkatiedon jakamisen yhtenäistämisen tarve syntyi vuonna 2007 voimaan tullen INSPIRE-direktiivin myötä. (JHS 180) Paikkatiedon yhteiskäytön ohjaamiseksi julkisen hallinnon tietohallinnon neuvottelulautakunta (JUHTA) on laatinut suosituksensa, joka pohjautuu pitkälti OGC:n määrittelemiin standardeihin sekä ISO TC 211 -standardikomitean verkkopalvelustandardiin. Suosituksen tarkoituksena on yhtenäistää menetelmä, jolla viranomaisten hallussa olevaa paikkatietoa julkaistaan avoimena tietona. Käytännössä tällä pyritään vähentämään rinnakkaisten tietokantojen ylläpitoa sekä saavuttamaan mahdollisimman tehokkaasti hyödynnettävä yhteiskäyttöinen tietopalvelurajapinta.

2.1 Web Map Service (WMS)

WMS on OGC-standardi rajapintapalvelusta, jota noudattamalla asiakasohjelma voi tehdä karttakyselyitä palvelimelta (OGC 2006: 06-042). Karttakyselyt tehdään http-protokollalla koodatuin URL-viittauksin, jonka perusteella palvelin muodostaa georeferoituja rasterikuvia dynaamisesti suoraan tietolähteestä. Sen jälkeen palvelin lähettää kyselyn tehneelle asiakasohjelmalle muodostetun karttakuvan käyttäen hyvin tunnettuja formaatteja kuten PNG, JPEG, GIF tai joissain harvinaisissa toteutuksissa SVG. Koska kuva muodostetaan suoraan tietokannasta, se soveltuu hyvin karttapalveluihin, jossa on

oleellista paikkatiedon ajantasaisuus. Tämän lisäksi kuva voidaan muodostaa juuri halutulta mittakaavalta sekä se voidaan visualisoida dynaamisesti toivotulla tavalla tyylisääntöjen mukaan. Visualisointiin on olemassa OGC-standardi Styled Layer Description (SLD).

2.2 Kyselyt karttarajapinnassa

OGC-standardi WMS-palvelusta määrittelee kolme perustehtävää, jotka karttapalveluun tulee määrittää, jotta standardia noudattavat asiakasohjelmat voivat hakea karttakuvia palvelusta (JHS 180). Nämä kyselyt ovat: palvelun metatietojen kysely ”GetCapabilities” (esimerkkikoodi 1), karttakysely ”GetMap” sekä kohdetietojen kysely ”GetFeatureInfo”. Suomessa julkisen hallinnon suositus on se, että karttapalvelussa tulee määritellä metatietojen kysely ja karttakysely, mutta kolmas eli kohdetietojen kysely on vapaaehtoinen.

Kuten edellä on mainittu, kyselyt tehdään kansainväliseen standardiin Hyper Text Transfer Protocol HTTP-GET-kyselyillä, jotka koodataan URL-muotoon (OGC 2006: 06-042). Kyselyt tehdään avainarvopareilla, jossa avain edustaa parametria ja arvo parametrin muuttujaa. Parametrit ovat joko standardin määrittelemiä parametrejä tai ne voidaan selvittää metatietokyselyn perusteella. Standardi varaa joitakin symboleita käytettäväksi URL-kyselyissä, joista WMS käyttää symboleita ”?”, ”&”, ”=”, ”,” ja ”+”.

Taulukko 1. Varatut symbolit WMS-palvelukyselyissä (OGC 2006: 06-042)

Symboli	Varattu käyttötarkoitus
?	Ilmaisee kyselyn alkamista
&	Erottaa kyselyssä olevat parametrit toisistaan
=	Erotin parametrin avaimen ja arvon välillä
,	Erottaa listatut arvot toisistaan listatuissa parametreissä (kuten BBOX, LAYERS ja STYLES getMap-metodissa)
+	Välilyönnin lyhyt kuvaus

Palvelu vastaa kyselyyn palauttamalla tiedoston XML-muodossa (GetCapablities- ja GetFeatureInfo-kyselyyn) tai mahdollisen virheviestin. GetMap-kyselyn vastauksena palvelu lähettää kuvatiedoston joko toivotussa formaatissa tai oletusformaatissa. (JHS 180)

```
http://karttapalveu.fi/wms?SERVICE=WMS&REQUEST=GetCapablities
```

Esimerkkikoodi 1. Näin voitaisiin esimerkiksi tehdä GetCapablities-kysely palveluun

2.2.1 Palvelun metatietojen kysely (GetCapabilities)

Karttapalvelun perustiedot eli metatietojen selviävät asiakasohjelmalle GetCapabilities-kyselyn avulla (JHS 180). GetCapabilities-dokumenttia voidaan ajatella karttapalvelun katalogina, josta selviää palvelun tarjonta. Metatietojen kysely käsittää myös parametreinä version ja palvelutyypin. Versiolla tarkoitetaan standardin versiota, jolla voidaan varmistaa, että asiakasohjelma ja palvelu toteutettavat samaa standardiversiota, sillä esimerkiksi karttatyyleihin liittyvissä skeemoissa on eroja versioittain. Palvelutyyppi voi olla WMS, mutta samassa palvelussa voi olla olemassa myös muita palvelutyyppejä, jotka toimivat hieman eri toimintaperiaatteella, kuten tiilitystä tukevat karttapalvelut. Metatietojen pohjalta voidaan määritellä parametrit, joilla asiakasohjelma pystyy aloittamaan karttakyselyiden tekemisen palvelimelle.

2.2.2 Karttakyselyt (GetMap)

Karttakysely on karttapalvelun keskeisin tehtävä, sillä sen avulla asiakasohjelma hakee karttakuvat, joita se lopulta esittää käyttäjälle (JHS 180). Aina kun asiakasohjelmassa vieritetään kuvaa tai muutetaan sen mittakaavaa, tehdään karttapalveluun yksi tai useampi karttakysely (taulukko 2). Poikkeuksena tähän ovat välimuistiin tallennetut kuvat, joita jotkut asiakasohjelmat käyttävät hyödyksi. Mikäli palvelin ei kykene tuottamaan uutta karttakuvaa, tulee sen lähettää paluuviestissä virheilmoitus.

Taulukko 2. Luettelo GetMap-kyselyn parametreista (P = pakollinen, V = vapaaehtoinen). (JHS 180)

Pyyntö=parametri	P/V	Kuvaus
VERSION=1.3.0	P	WMS-standardin versio.
REQUEST=GetMap	P	Kyselytyyppi.
LAYERS=layer_list	P	Pilkulla erotettu lista halutuista tasoista.
STYLES=style_list	P	Pilkulla erotettu lista halutuista tyyleistä.
CRS=namespace:identifier	P	Koordinaatti järjestelmän nimi ja koodi, esim ESPG:3857.
BBOX=minx,miny,maxx,maxy	P	Kyselyn koordinaattirajaus.
WIDTH=output_width	P	Vastauskuvan leveys pikseleinä.
HEIGHT=output_height	P	Vastauskuvan korkeus pikseleinä.
FORMAT=output_format	P	Vastauskuvan formaatti.
TRANSPARENT=TRUE|FALSE	V	Vastauskuvan taustan läpinäkyvyys.
BGCOLOR=color_value	V	Vastauskuvan taustaväri (heksadesimaaliluku).
EXCEPTIONS=exception_format	V	Virheviestin esitysmuoto (oletus=XML).
TIME=time	V	Ajanhetki, josta visualisointi halutaan.
ELEVATION=elevation	V	Korkeusasema, josta visualisointi halutaan.
<muu dimensio>=parametri	V	Valittava parametri

Viimeisenä listassa esiintyvä ”muu dimensio” on mahdollinen ylimääräinen parametri, jonka määrittämällä WMS-palvelu voi tarjoilla kuvia toisesta kuvakulmasta esimerkiksi, jos on kyse satelliittikuvista (OGC 2006: 06-042.)

2.2.3 Kohteen lisätietojen kysely (GetFeatureInfo)

GetFeatureInfo-kysely on Suomessa karttapalveluille vapaaehtoinen määritellä. Sen käyttötarkoitus on antaa karttakohteelle lisätietoja, kun esimerkiksi kartalta osoitetaan kohdetta hiirellä, saadaan esille ponnahdusikkuna, joka sisältää kohteen ominaisuustiedot (JHS 180.)

2.3 Tiilitystä tukevat karttapalvelut

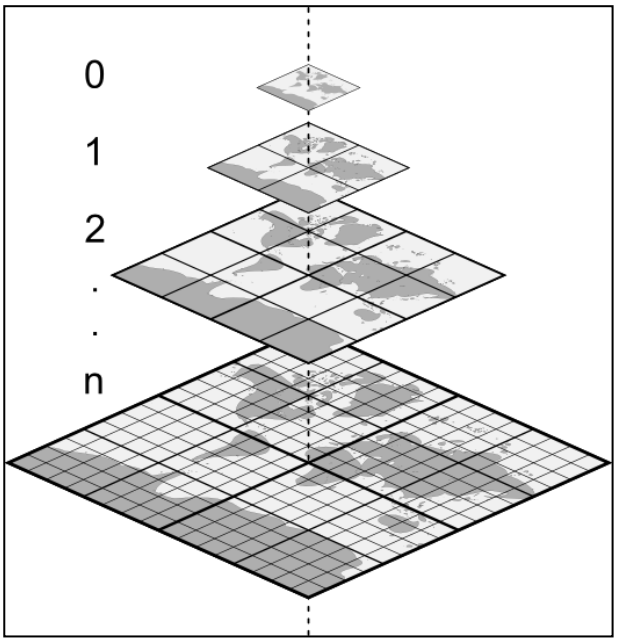
2.3.1 Mittakaavatasot tiilitystä tukevissa karttapalveluissa

Tiedonsiirron tehostamiseksi karttakuva voidaan myös ladata osissa yhden kuvan sijaan. Esimerkiksi tässä tutkielmassa käytetyt karttakirjastot Openlayers ja Leaflet tarjoavat kummatkin kaksi latausstrategiaa WMS-palvelulle, joista toinen lataa kerralla koko kuvan ja toinen lataa kuvan osissa. Karttapalvelun suorituskykyä ja skaalautuvuutta voidaan tehostaa myös sillä, että kuvat muodostetaan valmiiksi palvelimen välimuistiin. Tämä menetelmä tunnetaan olleen käytössä Googlella ja Nasalla, jonka inspiroimana Open Source Geospatial Foundation on julkaissut oman spesifikaationsa Tile Map Service TMS-karttapalvelun (OGC 2010: 07-057r7.) Toinen avoimeen standardiin perustuva karttapalvelu, joka on yleisessä käytössä, on OGC:n standardoima Web Map Tile Service (WMTS), jonka tarkoitus on täydentää WMS-palvelua.

Siinä missä WMS on erinomainen tapa julkaista dynaamisesti lähtöaineistosta karttamateriaalia, vaihtaa WMTS dynaamisuutensa tehokkuuteen (OGC 2010: 07-057r7). Tiilitystä tukevissa karttapalveluissa määritellään ennalta kuvasarja, joka muodostaa kuvapyramidin. Kuvapyramidi (kuva 1) voidaan muodostaa globaalisti tai rajata paikallisesti. Kummassakin tapauksessa asiakasohjelman tulee olla tietoinen siitä, millä alueella ja missä koordinaattijärjestelmässä karttapalvelussa oleva kuvapyramidi sijaitsee.

Toisin kuin WMS-palvelussa, jossa kartan mittakaavaa voidaan vaihtaa dynaamisesti, rajoittaa valmiiksi muodostettu kuvapyramidi mittakaavan vapaasti valitsemista. Tämän johdosta kartan lähdemateriaali muodostetaan juuri pyramidiksi, sillä pyramidin seuraava taso on aina mittakaavaltaan puolet edellisestä, jolloin pinta-ala nelinkertaistuu. Sen sijaan, että koko maailma tarkimmilla mittakaavoilla muutettaisiin yhdeksi erittäin korkean resoluution kuvaksi, paloitellaan kartta osiin eli tiileihin. Tällöin asiakasohjelmassa voidaan ladata kartta vain siltä osin, kuin se halutaan esittää. Kun ladattavien tiedostojen määrä pyritään rajoittamaan, pienenee vasteajat lopullisen karttakuvan muodostamiseksi asiakasohjelmassa. Yleisin käytetty tiilien koko on 256x256 pikseliä, joka on oletuksena esimerkiksi Geoserver WMTS-palvelussa, sekä asiakasohjelmissa kuten Leaflet ja Openlayers. Toinen yleinen suositus on se, ettei kuvapyramidin korkeus ylittäisi 20 tasoa, sillä tiilien määrä kasvaisi tällöin hallitsemattoman suureksi vaatien suuren

määrän levytilaa. Kuvapyramidin tiilien kokonaismäärä on suurimmillaan $x = \sum 4^0 + 4^1 + \cdots + 4^n$, jossa *n* on pyramidin korkeus. Tiilien lopulliseen määrään vaikuttaa kuitenkin myös aluerajaus.



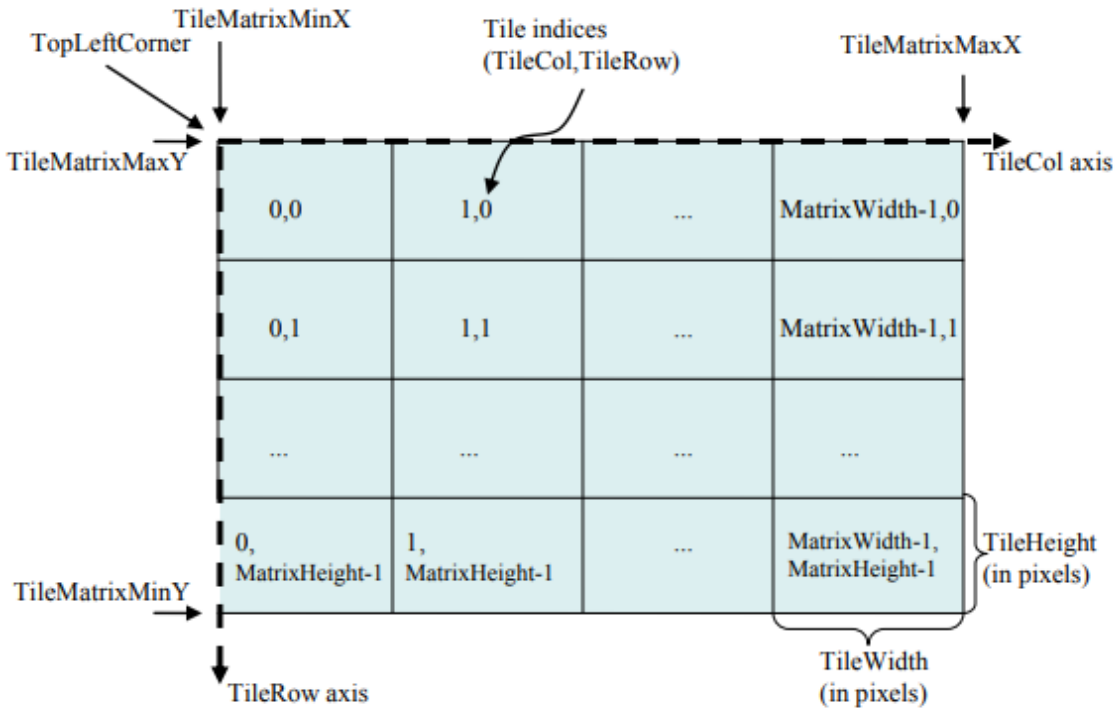
Kuva 1. Kuvapyramidi havainnollistettuna (Michael Beale:AutodeskForgecommunity-blog)

Taulukko 3. Esimerkki, miten OpenStreetMap yhteisön voimin ylläpidetty kartta muodostaa koko maailmankartan seuraavanlaiseen kuvapyramidiin (openstreetmap: zoom-levels)

Level	# Tiles	Tile width (° of longitudes)	m / pixel (on Equator)	~ Scale (on screen)	Examples of areas to represent
0	1	360	156 412	1:500 million	whole world
1	4	180	78 206	1:250 million	
2	16	90	39 103	1:150 million	subcontinental area
3	64	45	19 551	1:70 million	largest country
4	256	22.5	9 776	1:35 million	
5	1 024	11.25	4 888	1:15 million	large African country
6	4 096	5.625	2 444	1:10 million	large European country
7	16 384	2.813	1 222	1:4 million	small country, US state
8	65 536	1.406	610.984	1:2 million	
9	262 144	0.703	305.492	1:1 million	wide area, large metropolitan area
10	1 048 576	0.352	152.746	1:500 thousand	metropolitan area
11	4 194 304	0.176	76.373	1:250 thousand	city
12	16 777 216	0.088	38.187	1:150 thousand	town, or city district
13	67 108 864	0.044	19.093	1:70 thousand	village, or suburb
14	268 435 456	0.022	9.547	1:35 thousand	
15	1 073 741 824	0.011	4.773	1:15 thousand	small road
16	4 294 967 296	0.005	2.387	1:8 thousand	street
17	17 179 869 184	0.003	1.193	1:4 thousand	block, park, addresses
18	68 719 476 736	0.001	0.596	1:2 thousand	some buildings, trees
19	274 877 906 944	0.0005	0.298	1:1 thousand	local highway and crossing details

2.3.2 Karttakyselyt tiilitetyissä kartoissa

Koska karttatiilet ovat valmiiksi muodostettuja georeferoituja kuvia, tulee asiakasohjelman ymmärtää ruudukkorakenne (kuva 2), mihin kuvat ovat määritelty karttapalvelussa (OGC 2010: 07-057r7.) Tietojen pohjalta asiakasohjelma kykenee laskemaan haluamansa kuvat karttapalvelusta ja hakemaan niitä HTTP-kyselyinä. Asiakasohjelman tulee tietää vähintään ruudukosta sen koordinaattijärjestelmä sekä kuvien koko pikseleinä.



Kuva 2. OGC: WMTS-standardin määritys kuvaruudukolle (OGC 2010: 07-057r7).

WMTS-palvelussa voidaan edelleen pyytää kuvia karttapalvelusta HTTP-kyselyillä KVP-koodattuina, mikä tarkoittaa attribuuttien asettamista avain-arvo-pareina URL-kyselyissä. Kysely voidaan myös koodata HTTP-kyselyn runkoon Simple Object Access Protocol eli SOAP-koodattuina. Koska kuvat ovat karttapalvelussa tallennettuna ja niillä on olemassa sijainti, ne voidaan myös palvella RESTful-lähestymistavalla, jossa jokainen tiili voidaan hakea suoralla URL-viittauksella. (JHS 180.)

2.4 Vektoritiilet yleisesti karttapalveluissa

Riippumatta siitä onko kyse rasteri- vai vektoritiileistä, on tiilitystä tukeville karttapalveluille aina ominaista se, että tiilet lasketaan palvelimen välimuistiin etukäteen (Springmeyer 2015). Keskeinen ero rasteri- ja vektoritiiliformaateilla on se, että rasterit muodostetaan valmiiksi hyvin tunnettuihin kuvaformaatteihin jo palvelimen puolella, kun taas vektoriformaatti muodostetaan kuvaksi vasta asiakasohjelman puolella. Vektorimuotoinen aineisto antaa asiakasohjelmalle pääsyn alkuperäiseen geometriaan ja mahdollistaa kuvan muodostamisen valikoidusti. Lopullinen kuva voidaan muodostaa asiakasohjelmassa määritetyllä omalla väri- ja tyyliskeemalla, jotka voidaan myös halutessa vaihtaa lennossa hyödyntäen samaa lähdemateriaalia.

Vektoritiilien tarkoitus on parantaa järjestelmän skaalautuvuutta entisestään verrattuna rasteritiedostoihin pohjautuvaa tiilitykseen (Springmeyer 2015). Skaalautuvuus saavutetaan sillä, että vektoritiilet vaativat keskimäärin vähemmän levytilaa kuin rasteritiilet. Toinen skaalautuvuuden etu on se, että vektoritiilet muodostetaan kuviksi vasta asiakasohjelmassa, jolloin samalla tiilipyramidilla voidaan muodostaa useita kartta tyylejä. Rasteritiileissä haluttaessa useita eri tyylivaihtoehtoja täytyy tehdä kokonaan uusi kuvapyramidi, joka vaatii nopeasti paljon levytilaa. Vektoritiilien heikkous on siinä, että niiden konfigurointi asiakasohjelmassa on haastavampaa kuin rasteritiilien. (Cavazzi 2017.) Lisäksi markkinoilla esiintyy niin monta toteutusta vektoritiileistä, että vielä ei ole olemassa yhtä yhtenäistä tapaa toteuttaa vektoritiilejä, mikä johtaa siihen, että kaikkia formaatteja tukevan asiakasohjelman suunnittelu on hankalaa.

3 Mapbox-vektoritiili

Keskeinen osa vektoritiilitoteutuksissa on vaihtoformaatti, jossa Mapbox on halunnut erityisesti kiinnittää huomiota tiedostokoon minimointiin sekä vähentää redundanssia säilyttäen kuitenkin attribuutit ja geometriat tiileissä (Springmeyer 2015). Mapbox on julkaissut spesifikaation vektoritiileistään avoimella lisenssillä. Formaatin taustalla on ollut yhtiön JavaScript-kirjasto Mapbox-GL-JS, jossa yhtiöllä on ollut tarkoituksena luoda tuote, jossa karttasymbologia on täysin asiakkaan muokattavissa. Tämän kirjaston tarkoituksena on ollut se, että samaa tietolähdettä käyttäen voidaan muodostaa useita eri

karttatyylejä. Mapbox-GL-JS piirtää kartan selaimessa HTML-canvas-elementtiin nimensä mukaan hyödyntäen WebGL ohjelmointirajapintaa (Mapbox-GL-JS. 2019.) Koska vektoritiilet sisältävät geometrisiä alkioita on käytännössä mahdollista piirtää karttakuva useassa eri mittakaavassa hyödyntäen samoja vektoritiilejä. Mapbox-GL-JS ottaa tämän huomioon ja esimerkiksi Mapbox julkaisee omat taustakarttansa vain 14 tasoisena. Pienempi mittakaavaiset tasot piirretään hyödyntäen myös tason 14 vektoritiiliä. (Springmeyer 2015.) Luvun kaksi (taulukko 1) tämä vastaisi mittakaavaa 1:70 000, jolloin tasoja on kuusi vähemmän kuin perinteisessä kahdessakymmenessä tasossa. Käytännössä tämä tarkoittaa sitä, että toisin kuin rasteritiileillä, vektoritiilejen mittakaavaa voidaan muuttaa dynaamisesti.

3.1 Tiedostoformaatti

Mapbox hyödyntää vektoritiileissään Googlen avoimena lähdekoodina julkaisemia protokollapuskureita (Google Protocol Buffers) (vector-tile-spec 2014). Googlen protokollapuskurit mahdollistavat tiedon sarjallistamisen binäärimuotoon, jossa se voidaan tallentaa tiedostoon ja siirtää tietoliikenneverkossa toiselle koneelle (Protocol Buffers 2018). Protokollapuskurit ovat monikäyttöisä ja tarjoavat tehokkaan keinon automatisoida tiedon sarjallistamisen. Toiminta perustuu siihen, että Googlen julkaisemalla lähdekoodilla voidaan koodata tietoa binäärimuotoon, minkä jälkeen se voidaan siirtää esimerkiksi tietoliikenneverkon välityksellä toiselle koneelle. Kun tietoa halutaan hyödyntää, koodi voidaan purkaa käyttämällä samaa koodiavainta. Mapbox-vektoritiilin tapauksessa sen spesifikaatio määrittää skeeman, jossa geometria ja attribuutit tulevat koodata, jotta samaa spesifikaatiota hyödyntävät ohjelmistot voivat esittää tietoa. Käytännössä vektoritiilispesifikaatio on ”.proto”-tiedostopäätteinen tiedosto, joka sisältää määritelmän skeemasta, jossa vektoritiilet tulee koodata. Samalla proto-tiedostolla voidaan myös purkaa vektoritiili esimerkiksi asiakasohjelmassa.

Protokollapuskurit ovat alustariippumattomia, ja ne ovat tuettuja usealla ohjelmointikielellä (Protocol Buffers 2018). Laajan tukensa ansiosta protokollapuskureita käyttääkseen ei tarvitse erikseen kirjoittaa jäsennintä tiedon käyttämiseksi, vaan tietoon päästään suoraan käsiksi koodiavaimella eli proto-tiedostolla ja avoimen lähdekoodin purkutyökaluilla. Google on kehittänyt tätä teknologiaa palvelimella tehtäviä pyyntöjä ja vastauksia varten. Googlen mukaan nämä pyynnöt voitaisiin suorittaa myös XML-formaatissa, mutta

protokollapuskureilla on monia etuja XML-tiedostoihin nähden: ne ovat yksinkertaisempia, pienempiä tiedostokooltaan sekä nopeampia lähettää tietoliikenneverkossa. Protokollapuskureiden kompakti tiedostokoko perustuu tavujen tehokkaaseen käyttöön, jossa on määritelty tapoja, jolla voidaan koodata tietoa sen tyypistä riippuen dynaamisesti.

3.2 Geometrian koodaaminen

Geometriset alkiot ovat koodattuna vektoritiileissä tietokoneen kuvaruuduissa yleisesti käytettyyn koordinaattijärjestelmään, jossa origo sijaitsee ruudun vasemmassa ylänurkassa ja x-akseli kasvaa oikeaan reunaan päin ja y-akseli ruudun alareunaan päin (Taulukko 4). Vektoritiilet eivät ole tietoisia kartan projektiosta taikka sen reunoista, vaan se olettaa, että koodinpurkajaohjelma tietää kaiken tarvittavan tiedon ennen purkamista.

3.2.1 Geometrian komentotyypit

Vektoritiilin koordinaatistossa muodostettavat geometriset alkiot muodostetaan kursoria liikuttamalla komentotyypeillä. (vector-tile-spec 2014.) Spesifikaatiosta on pääteltävissä se, että geometrian koodaus on suunniteltu siten, että se on tehokasta muodostaa kuvaksi hyödyntämällä WebGL-rajapintaa. Spesifikaatiossa määritellyt komentotyypit muistuttavat HTML-canvas-elementin 2D context-objektin metodeja. (HTML Canvas 2D Context 2015.)

Taulukko 4. Komentotyyppien parametrit taulukoituna sekä käytön kuvaus (vector-tile-spec 2014).

Komento	Parametrit	Kuvaus
MoveTo()	X, Y	Aloittaa geometrian vektoritiilin koordinaatistosta annetulta lukuparilta, joidenka tulee olla kokonaislukuja.
LineTo()	ΔX, ΔY	Parametrien tulee olla kokonaislukuaja sekä komentoa tulee edeltää joko MoveTo()-komento tai toinen LineTo()-komento.
ClosePath()	-	Parametriton komento, joka sulkee lineaarisen kehän monikulmioksi. Tämä komento ei siirrä kursorin sijaintia sen suorittamisen jälkeen

Kaikki spesifikaation määrittelemät geometriat voidaan muodostaa näillä kolmella komentotyypillä (vector-tile-spec. 2014).

3.2.2 Geometriatyypit

Vektoritiilispesifikaatio määrittelee geometrialle neljä tyyppiä:

- UNKNOWN
- POINT
- LINESTRING
- POLYGON.

Ensimmäinen geometriatyyppi ”UNKNOWN” on jätetty tuntemattomaksi tulevaisuudessa monimutkaisempien geometrioiden tukemista varten, esimerkiksi kaarien (vector-tile-spec 2014). Tämä antaa myös ohjelmistokehittäjille mahdollisuuden spesifikaatiota noudattamalla lisätä järjestelmiinsä monimutkaisempia geometrioita. Koodia purkavat ohjelmat saattavat jättää kyseisen geometrian huomioimatta. Muut geometriat muodostetaan kolmen komentotyypin yhdistelmillä (taulukko 5).

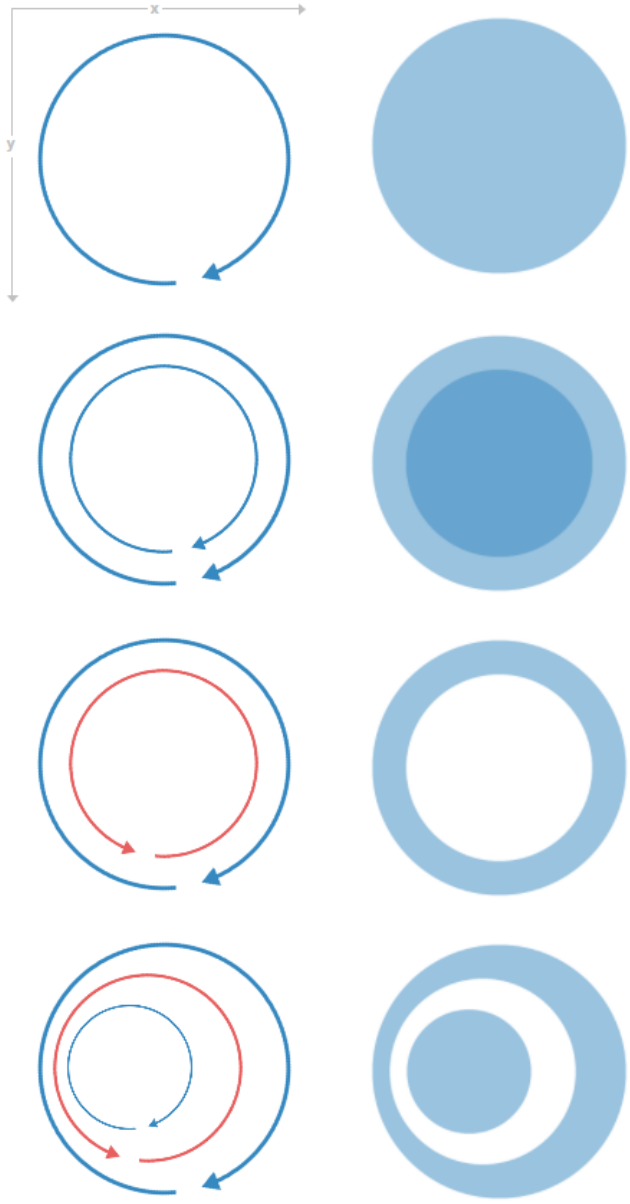
Taulukko 5. Nämä komentotyyppiyhdistelmät muodostavat kaikki geometriat. Muuttujan n on oltava nollaa suurempi LINESTRING ja suurempi kuin yksi POLLYGON-geometriassa. Kaikki geometriat alkavat MoveTo()-komennolla

Geometriatyyppi	**Komentotyypit**
POINT	MoveTo()
LINESTRING	MoveTo() + **n** · LineTo()
POLYGON	MoveTo() + **n** · LineTo() + ClosePath()

Geometria aloitetaan aina pisteellä siirtämällä kursori MoveTo()-komennolla. Seuraava MoveTo()-komento voidaan aloittaa uudestaan muodostetun geometrian sisällä monimutkaisemmissa geometrioissa ja sulkea lopulta ClosePath()-komennolla.

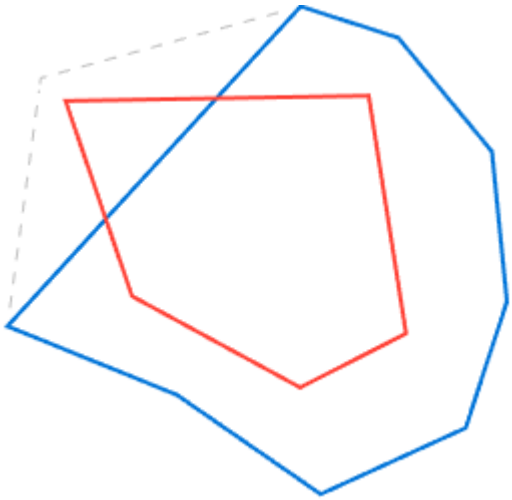
Monikulmiogeometriat käyttävät hyödyksi maanmittaustekniikassa yleisesti tunnettua Gaussin kolmiokaavaa (vector-tile-spec. 2014). Tämä kaava on hyödyllinen muodostettaessa monimutkaisia monikulmioita, jossa monikulmio on pinta, mutta pinta sisältää

reikiä. Tämän tyyppinen geometria voisi olla esimerkiksi järven pinta, joka on eheä geometria, mutta saaret muodostavat aukkoja pinnalle. Gaussin kolmiokaavassa geometrian pinta-ala saadaan laskettua, kun monikulmion uloimman reunana koordinaattien muodostama kuvio lasketaan myötäpäivään ja aukon muodostava sisäreunan kuvio lasketaan vastapäivään. Vektoritiileissä muodostettu pinta-ala peitetään pinnalla ja aukot jätetään läpinäkyviksi.



Kuva 3. Havainnekuva siitä kuinka kiertosuunta vaikuttaa monikulmion geometriaan (vector-tile-spec 2014)

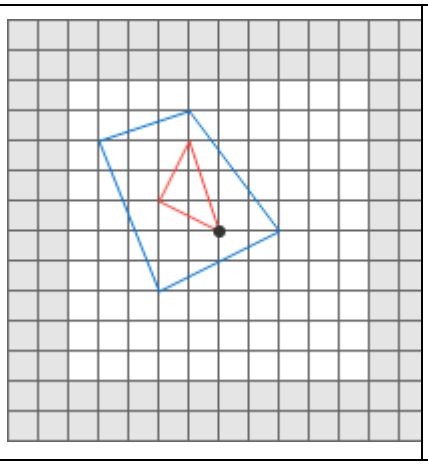
Huomionarvoista on myös se, ettei sisäreunan geometria saa missään kohdassa leikata ulkoreunan geometriaa sillä se muodostaa epähyväksyttävän geometrian. Sisäreunan geometria saattaa usein leikata toisiaan, kun geometriaa yksinkertaistetaan, esimerkiksi kaari muodostetaan murtoviivoiksi tai välipisteitä harvennetaan geometrian pelkistämiseksi.



Kuva 4. Esimerkki yksinkertaistuksesta, joka muodostaa epähyväksyttävän geometrian. (vector-tile-spec. 2014)

Tietokannoissa tai muissa tiedostoformaateissa on usein myös tallennettuna kaaria tai muita monimutkaisempia geometrioita, jotka yksinkertaistetaan vektoritiileissä viivoiksi ja pisteiksi (kuva 5).

Komennot		

`MoveTo(1,2)`
`LineTo(3,-1)`
`LineTo(3,4)`
`LineTo(-4,2)`
`ClosePath()`
`MoveTo(1,-5)`
`LineTo(-1,2)`
`LineTo(2,1)`
`ClosePath()`
		

Kuva 5. Monikulmiogeometrian muodostuksen esimerkki (vector-tile-spec. 2014)

3.3 Attribuuttien koodaaminen

Vektoritiileissä attribuutille koodataan tunnu ”tags”, joka ilmaisee avaimen ”keys” joka puolestaan sisältää sen arvoparin ”values” (esimerkkikoodi 2). Avain-arvo-parin ”tags”-menetelmän tarkoituksena on vähentää toistojen määrää koodissa, mikä on oleellista eteenkin suurten geometrioiden suhteen, jossa attribuutit saattavat toistua usein.

Alkuperäinen geojson

```
{
  "type": "FeatureCollection",
    "features": [
    {
      "geometry": { ... },
      "type": "Feature",
      "properties": {
        "hello": "world",
        "h": "world",
        "count": 1.23
    }
  },
  {
    "geometry": { ... },
    "type": "Feature",
    "properties": {
      "hello": "again",
      "count": 2
    }
  }
 ]
}
```

Vektoritiilimuoto

```
layers {
  version: 2
  name: "points"
  features: {
    id: 1
    tags: 0
    tags: 0
    tags: 1
    tags: 0
    tags: 2
    tags: 1
    type: Point geometry: ...
  }
  features {
    id: 2
    tags: 0
    tags: 2
    tags: 2
    tags: 3
    type: Point geometry: ...
  }
  keys: "hello"
  keys: "h"
  keys: "count"
  values: {
    string_value: "world"
  }
  values: {
    double_value: 1.23
  }
  values: {
    string_value: "again"
  }
  values: {
   int_value: 2
  }
  extent: 4096
}
```

Esimerkkikoodi 2. Vektoritiilispesifikaatiossa esiintyvä esimerkki geojson-formaatissa oleva paikkatieto muunnettuna vektoritiili-formaattiin (vector-tile-spec 2014).

Kun yksittäiselle karttasymbolille koodataan vain yksi luku ”tags”-arvo attribuuttia kohden, vähentää se koodattavien tavujen määrää. Tällöin attribuuttien todelliset arvot voidaan ilmaista tiedostossa vain kerran. Tämä ominaisuus tulee käytettäessä Googlen protokollapuskureita (Protocol Buffers 2018). Googlen avoimeen lähdekoodiin perustuvalla koodinpurkajalla voidaan myös protokollapuskureista hakea tehokkaasti tietoa. Tällöin voidaan esimerkiksi toteuttaa tehokas hakumenetelmä attribuuteille asiakasohjelmassa, vaikka sille, kun kartan osaa klikataan.

4 Geoserver-vektoritiilitoteutus

Vantaan kaupungin karttarajapinnassa on aktiivisesti käytössä Geoserver-palvelinohjelmisto, jolla julkaistaan karttatasoja kaupungin sisäisiä ja ulkoisia asiakasohjelmia varten. Geoserver on web-palvelinohjelmisto, jolla karttatasojen julkaisu tapahtuu avointen rajapintastandardien mukaisesti. (Geoserver project. 2019.) Geoserver tukee käytetyimpiä tiedosto- ja tietokantaformaatteja sekä rasteri, että vektorimuodossa tallennetuille aineistoille. Mahdollisuus julkaista vektoritiileä Geoserverillä tuli julkaisussa 2.11. (Geoserver 2.11.0 Release 2017.) Tätä tutkimusta varten päivitettiin Vantaan kaupungilla toimiva Geoserver tutkimuksen alkamisajankohdalla uusimpaan versioon 2.13.2.

4.1 Tiilitystä tukevat karttapalvelut Geoserver-ohjelmistolla

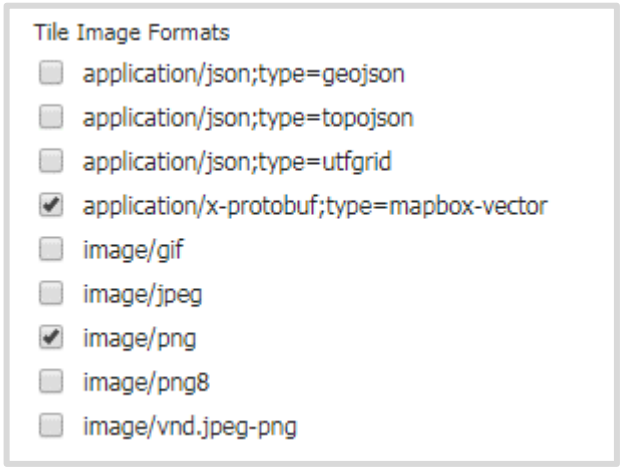
Tiilitystä tukevissa karttapalveluissa oleellista on tiilien tallentaminen välimuistiin. Geoserveriin on oletuksena integroitu Geowebcache-välityspalvelin, joka toimii Geoserverin ja asiakasohjelman välissä (Geowebcache 2019). Välityspalvelimen tarkoituksena on tiilitetyissä karttapalveluissa se, että asiakasohjelma tekee karttakyselyt välityspalvelimeen, joka tarkastaa ensin omasta välimuistista, löytyykö kyselyssä pyydettyjä karttakuvia. Mikäli kuvat löytyvät välityspalvelimen omasta välimuistista, pystyy välityspalvelin tarjoamaan kuvat tehokkaasti yhdellä levyhaulla, mutta jos kuvaa ei ole välimuistissa, joudutaan sitä pyytämään palvelinohjelmistolta. Kun palvelinohjelmistolta pyydetään kuvaa, se muodostaa sen ensin tietolähteestä, minkä jälkeen se viedään välityspalveluun ja edelleen asiakasohjelmalle. Tämän jälkeen uusi kuva on tallennettuna välityspalvelimelle, jolloin se voidaan seuraavan kerran palvella tehokkaammin.

Karttapalvelun toimintaa voidaan parantaa sillä, että kuvat muodostetaan välityspalveluun etukäteen. Huolimatta siitä, onko välityspalvelimen muistissa olevat kuvat tallennettu etukäteen vai ei, on oleellista määritellä niille voimassaoloaika. Voimassaoloaikaan vaikuttaa se, kuinka ajankohtaista aineiston tulee olla. Esimerkiksi ortoilmakuvat otetaan Vantaalla tiettyinä vuosina ja ne julkaistaan karttarajapinnassa siten, että tason kuvauksesta käy yksiselitteisesti ilmi, minkä vuoden kuvista on kyse. Ortokuvien voimassaoloaika on koko se aika, kun niitä ylläpidetään karttarajapinnassa. Asia on toinen esimerkiksi johtokartoissa, joissa aineiston täytyy olla tuoretta muutaman päivän aikaikkunalla.

4.2 Vektoritiilet Geoserver-ohjelmistolla

Kun Geoserver-asennus tukee vektoritiilejä, tulee se osaksi Geowebcachea, jossa voidaan valita formaatit, joilla karttatiilet julkaistaan tasokohtaisesti (Blasby & Hocevar 2016). Geoserver WMS sisältää useita kuvanmuodostamisen mahdollisuuksia, joita voidaan hyödyntää esilasketuissa karttakuvissa (kuva 6). Esilaskenta sisältää kaksi pääprosessia, jonka jälkeen ne voidaan tallentaa haluttuun formaattiin. Pääprosessit ovat seuraavat:

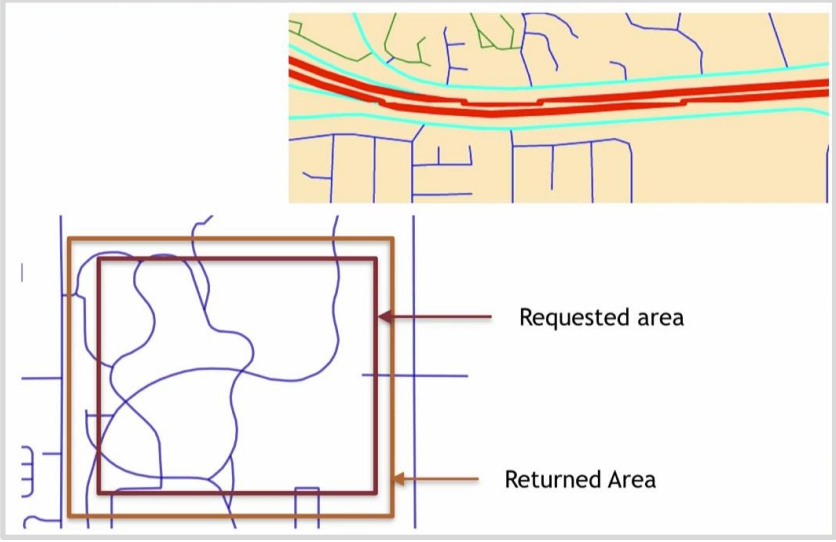
- Streaming Rendering – käytetään rasterikuvien muodostamiseen
- Vector Tiles Renderer – käytetään vektoritiilien muodostuksessa.



Kuva 6. Geoserver web-käyttöliittymässä valittavat julkaisuformaatit

Kummatkin prosessit toimivat lähes samalla periaatteella. Kummatkin prosessit alkavat tietokantakyselyillä, minkä jälkeen haettua tietoa yleistetään siten, että suurilla mittakaavoilla ei esitetä pieniä yksityiskohtia. Vektoritiileissä tämä keventää tiedoston kokoa, joka on tärkeää tiedon siirrossa. Lisäksi yleistämisellä kummassakin prosessissa pyritään pitämään karttasymbologia selkeänä. Liialliset yksityiskohdat karttasymbologiassa saattavat tehdä kartasta hankalasti luettavan ja sen selittävyys kärsii. Rasterikuva tallennetaan lopulta johonkin hyvin tunnettuun rasterikuvaformaattiin eli bittikartaksi, joka pystytään avaamaan useimmilla kuvankäsittelyohjelmilla ja jota on helppo hyödyntää web-kartoissa. Vektoritiili tallennetaan binäärimuotoisena datatiilenä, jonka pohjalta voidaan piirtää rasterikuva myöhemmin asiakasohjelmassa. Geoserverillä kummatkin hyödyntävät SLD-tyylejä määritelläkseen ne tasot, jotka otetaan tietyssä mittakaavassa mukaan, mutta vain rasterikuvat hyödyntävät tyylitietoja, kuten viivan paksuutta ja täyttövärejä.

Kartan leikkaus eroaa siinä, että vektoritiileissä palautetaan asiakasohjelmalle hieman tämän pyytämää aluetta suurempi alue. Tämä laajennusalue on kuvan muodostamista varten, sillä vektoritiilten reunalla viivan paksuuksien määrittely saattaa aiheuttaa epätasaisen lopputuloksen. Kuvassa 7 on FOSS4G Saksassa 2016 paikkatietotapahtumassa Geoserver vektoritiili-lisäosan kehittäjän David Blasbyn luentokalvoilta esimerkki siitä, kuinka vektoritiilet käyttäytyvät ilman laajennusaluetta.



Kuva 7. Vektoritiilien karttaleikkaus. Oikealla yläkulmassa kartta ilman laajennusaluetta. (Blasby & Hocevar 2016.)

5 Suorituskykytestaus

Ihmisen ja koneen kanssakäymistä on tutkittu paljon. Koneen vasteajan psykologinen vaikutus on ollut hyvin keskeisessä osassa tutkimuksia. (Nielsen 1993.) Lähtökohtaisesti vasteaikojen tulee olla aina mahdollisimman pieniä, mutta myös tietokoneen responsiivisella suunnittelulla on merkitystä miellyttävän käyttäjäkokemuksen kannalta. Nilsen (1993) on tutkimuksessaan peilannut tietokoneen ja ihmisen vuorovaikutusta hyvin pitkälti 60-luvun lopulla vakiintuneisiin rajoihin, joita on hyvä tarkkailla ohjelmistosuunnittelussa. Vasteajoilla on kolme selkeää rajaa, jotka tulee huomioida. Ensimmäinen raja on 0,1 sekunnin raja, jossa tietokone vastaa ihmisen komentoon välittömästi näyttäen vuorovaikutuksen tuloksen saman tien. Toinen raja on 1,0 sekunnin raja, jossa käyttäjä kokee työskentelevänsä ilman keskeytyksiä eikä toisessa rajassa vielä ole tarpeen suunnitella erityisen palautteen antamista käyttäjän odottelulle. Ensimmäisen ja toisen rajan välimaastossa käyttäjä kuitenkin menettää tunteen välittömästä vuorovaikutuksesta järjestelmän kanssa. Kymmenen sekunnin viiveaika vaati jo järjestelmältä palautetta tai dialogia pitääkseen käyttäjänsä huomion.

Web-kartta on siitä haasteellinen toteuttaa, että se vaatii karttakuvan hakemista dynaamisesti palvelimelta aina kuvaa vierittäessä tai mittakaavaa muutettaessa. Tämä tarkoittaa sitä, että karttapalveluissa ei ole samoja mahdollisuuksia esimerkiksi puskuroida ennakkoon latauksia kuten teräväpiirtoista videomateriaalia lähettävissä suoratoistopalveluissa. Tämä johtaa usein suuriin dynaamisesti suoritettaviin tiedostohakuihin tietoliikenneverkossa, jonka saaminen 0,1 sekunnin rajan alapuolelle on haastavaa. Erityisesti tiilitystä tukevissa karttapalveluissa hyvän suorituskyvyn saavuttamiseksi ei vielä riitä se, että keskimäärin yhden tiilen hakeminen on nopeata. Todelliset vasteajat käyttäjän ja karttajärjestelmän välisessä vuorovaikutuksessa ovat niitä aikoja, kauanko kartalla kestää kokonaisuudessaan päivittyä, kun käyttäjä vierittää tai muuttaa mittakaavaa.

5.1 Tutkimuksen toteuttaminen

Tässä luvussa käydään läpi, kuinka vektoritiilet vertautuvat rasteritiileihin. Tutkimusmenetelmänä muodostetaan kaksi testisovellusta, jossa vertaillaan vektori- ja rasteritiilien suorituskykyä toisiinsa. Lisäksi tutkitaan kummankin toteutuksen ylläpidettävyyttä ja sitä, kuinka hyvin ne ovat hyödynnettävissä Vantaan kaupungilla olevissa asiakasohjelmissa.

Vertailuaineistot muodostetaan kaupungin johtokartoista, jotka koostuvat seitsemästä eri verkosta: sähkö-, tele-, kaasu-, kaukolämpö-, vesi-, hulevesi- sekä jätevesiverkosta. Tutkimuksen aikana havaittiin hyväksi muodostaa eteenkin vektoritiilet globaalissa koordinaattijärjestelmässä EPSG:3857, joka tunnetaan myös Pseudo-Mercatorina, joka on useimmissa asiakasohjelmissa oletuskoordinaattijärjestelmä. Tämän koordinaattijärjestelmän käyttäminen mahdollisti tutkimukseen valittujen asiakasohjelmien konfiguroinnin mahdollisimman yhdenmukaisesti sekä vektori- että rasteriformaatissa kuten myös tutkimuksessa käytetyissä asiakasohjelmissa, jotka toteutettiin Leaflet- ja Openlayers-kirjastoilla.

5.2 Vektori- ja rasteripyramidien julkaisu

Vektoritiilet julkaistaan Geoserver-palvelinohjelmistolla Mapboxin vektoritiilispesifikaation mukaisina, sillä se on Geoserverin suositus hyvän suorituskyvyn saavuttamiseksi (Geoserver User Manual 2018). Rasteriformaatiksi on luontevaa valita PNG, sillä se on web-sovelluksissa laajasti käytetty formaatti bittikarttaan perustuville kuvatiedostoille (PNG Spesification 1999). Se tarjoaa myös mahdollisuuden merkitä kuvan osia läpinäkyväksi, mikä mahdollistaa johtokarttojen piirtämisen taustakartan päälle.

Kaikista kaupungin seitsemästä johtokartasta julkaistiin 20 mittakaavatasoa tutkimusta varten. Julkaisuvaihe muodostaa ensimmäisen tutkimusosan, jossa tutkitaan vektoritiilien ylläpidettävyyttä, ja toisessa osassa tutkitaan niiden hyödyntämistä. Julkaisut toteutetaan yhdenmukaisesti kukin samalla aluerajauksella tallentaen välimuistiin samat tiilet. Mittakaavatasoista esilaskettiin 19 Geowebcachen avulla kummassakin formaatissa. Tämä tarkoittaa siis sitä, että asiakasohjelmalle tarjotaan valmiita tiilejä aina tasolle 19, mutta jos halutaan tarkastella tasoa 20, on se kuitenkin mahdollista. Kun välityspalvelin tarjoaa asiakasohjelmalle tason 20 tiiliä, tallentaa se ne välimuistiinsa, jolloin teoriassa koko taso 20 voi muodostua välimuistiin ahkerassa käytössä. Koko välimuisti voidaan myös tyhjentää Geoserverin web-käyttöliittymällä. Tämä suoritettiin ennen julkaisuun liittyvien tutkimusten aloittamista.

Tutkimustuloksia mitattiin siten, että tiilipyramidien levytilaa mitattiin Geoserverin käyttöliittymän ilmoittamien arvojen mukaan sekä niiden muodostamiseen käytettyä aikaa mitattiin Windows 7-käyttöjärjestelmän tiedostoselaimella. Tiedostoselain tallentaa

jokaiselle tiedostolle muokkauspäivämäärän, jolloin palvelimen levyiltä on mahdollista etsiä Geowebcachen kansiorakenne ja verrata kustakin johtokarttatasosta ensimmäiseksi ja viimeiseksi muodostettujen karttatiilien aikaleimaa.

Taulukko 6. Karttatasojen julkaisun mittaustulokset.

Verkko	Formaatti	Muisti	Prosessit	Kulunut aika
Sähkö	PNG	1.80 Gt	1	1h 22min
	VCT	993 Mt	1	2h 12min
Tele	PNG	1.77 Gt	2	38min
	VCT	327 Mt	2	1h 4min
Kaasu	PNG	1.62 Gt	2	37min
	VCT	16.1 Mt	2	54min
Kaukolämpö	PNG	1.64 Gt	2	37min
	VCT	16.1 Mt	2	1h 8min
Vesi	PNG	1.77 Gt	1	1h 6min
	VCT	293 Mt	1	1h 59min
Hulevesi	PNG	1.79 Gt	1	1h 9min
	VCT	281 Mt	1	1h 58min
Jätevesi	PNG	1.78Gt	1	1h 14min
	VCT	282 Mt	1	2h 32min

Aikaleimojen vertailun yhteydessä havaittiin myös, että jokaisen valmiin pyramidin tiilien määrä oli täsmälleen sama eli noin 400 000. Tämä merkitsee sitä, että myös niiden tiilien rajaamat alueet, jotka eivät sisällä johtoja muodostettiin myös tyhjiksi geometriaa sisältämättömiksi tiileiksi. Päätellen vektori- ja rasteritiilipyramidien vaatimasta levytilasta geometrian määrä vaikuttaa huomattavasti enemmän vektoritiileihin kuin rasteritiileihin. Rasteritiileissä tiheämpi sähköverkko vie vain noin 10 % enemmän levytilaa kuin kaasuverkko, kun taas vektoritiilissä muodostettujen aineistojen ero on yli 600 %.

Vaikka tiilien muodostus prosessi on hyvin samankaltainen niin rasteri- kuin vektoritiileissä, kului vektoritiilien muodostamiseen huomattavasti enemmän aikaa. Rasteritiilet pystytään muodostamaan nopeammin metatiilien avulla. (Geoserver user manual 2018.) Metatiilet on asetettu vakiona neljä kertaa neljään tiileen Geowebcachessa, mikä tarkoittaa sitä, että muodostettaessa karttakuvia, muodostetaan kerralla yksi iso karttakuva

suuremmalta alueelta, joka pilkotaan lopulta osiin. Joten oletusasetuksen tapauksessa pystytään kerralla muodostamaan kuusitoista kuvatiiltä. Geoserverin käyttäjäoppaan mukaan oletusasetus antaa parhaimman mahdollisen suhteen tehokkuuden, muistin käytön ja kuvan laadun suhteen. Lisäksi muodostettaessa suurempia alueita kerralla kartalla esiintyvät tekstit muodostuvat eheämpinä.

Vektoritiiliä esilaskettaessa metatiilien käyttö ei ole kuitenkaan mahdollista. Tämä kävi myös ilmi tutkimuksessa metatiilityksen oletusasetuksen johdosta. Lopputuloksena oli keskimäärin seitsemän minuutin laskenta-ajat yhdellä prosessilla, mikä johti kuitenkin siihen, että vain keskimmäinen tiili muodostettiin. Tiilien määrä laski kuudestoistaosaan rasteritiileihin verrattuna. Vastaavasti myös rasteritiilien muodostamisaika kasvoi huomattavasti suuremmaksi metatiiliasetuksen ollessa poissa käytössä. Johtopäätöksenä kannattaa ottaa metatiilien asetus huomioon, jos halutaan ylläpitää kumpaakin aineistoa. Heikkoutena on se, että asetusta ei voi vaihtaa tiedostoformaattikohtaisesti, vaan se vaikuttaa koko karttatasoon kaikissa formaateissa. Mittaustulosten kannalta on kuitenkin tarkoituksen mukaista ottaa huomioon rasteritiilien kohdalla aika, kun metatiilejä käytetään, sillä se vastaa todellisia käyttöolosuhteita.

Muita tiilien prosessointinopeuteen vaikuttavia tekijöitä ovat prosessien määrät palvelimella. Taulukossa 6 on mittaustuloksia, joissa tiilien muodostamiseen on varattu joko yksi tai kaksi prosessia palvelimelta, jonka vaikutus on kutakuinkin suoraan verrannollinen prosessien määrään. Samalla palvelimella on myös paljon muita prosesseja, joilla saattaa olla vaikutusta karttatiilien muodostusaikoihin. Tässä tutkielmassa ei kuitenkaan tutkittu palvelimen kuormitusta eikä optimaalista prosessien määrää karttatasojen muodostuksen suhteen.

Kun kaikki tiilet olivat muodostettu, eikä Geoserverillä ollut enää tiilinmuodostus prosesseja käynnissä ilmeni ongelmia Geoserverin vastausajoissa. Se ei ole selkeää, miksi tiilityksen jälkeen Geoserver vaikuttaa toimivan hitaasti ja kyseessä saattaakin olla bugi. Tähän kuitenkin auttaa uudelleenkäynnistys.

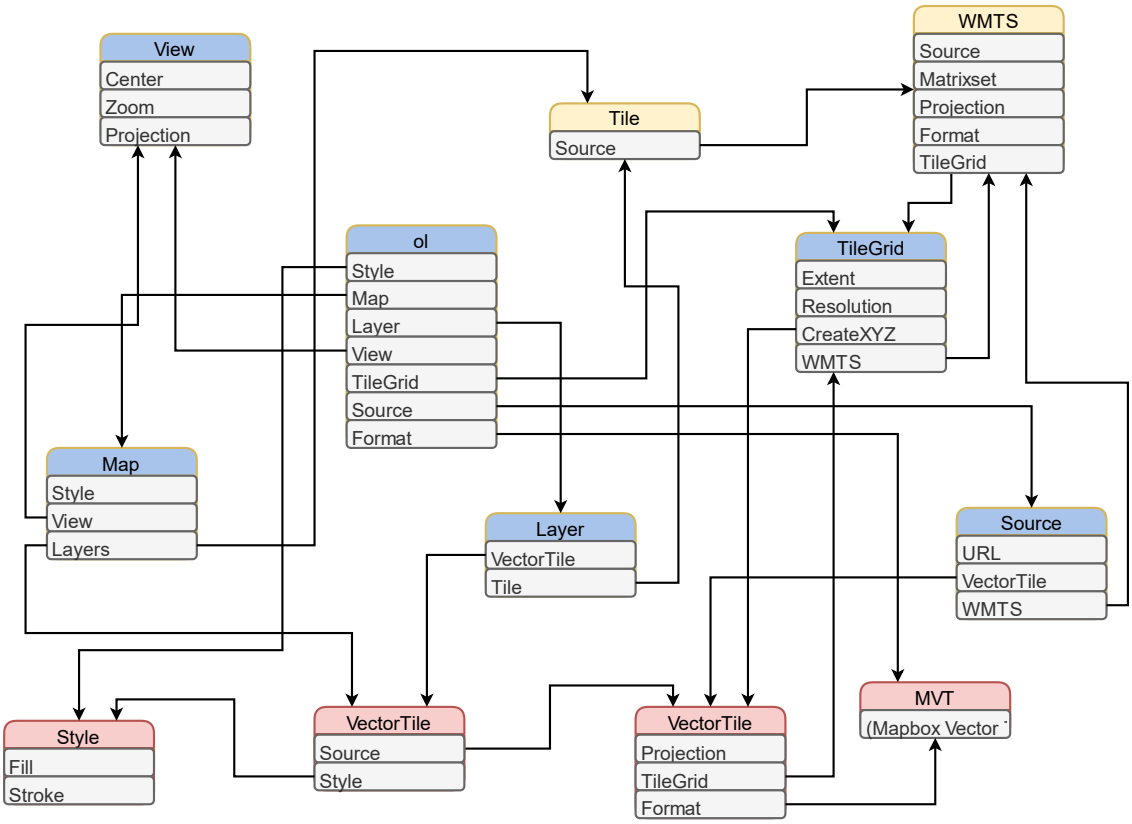
5.3 Testisovellusten rakentaminen suorituskyvyn vertailemiseksi

Vantaan kaupungin karttarajapintoja hyödyntävät useat kaupungin sisäiset ja ulkoiset asiakasohjelmat, joiden joukossa on niin työpöytäohjelmia kuin myös web-sovelluksia. Toistaiseksi Vantaan kaupungilla ei hyödynnetä vektoritiilitekniikkaa lainkaan. Kaupungilla käytössä olevista sovelluksista kuitenkin kummassakin kategoriassa sekä työpöytä-, että web-sovelluksina on ohjelmistoja, joilla vektoritiiliä on mahdollista hyödyntää. Jokainen näistä sovelluksista kuitenkin hyödyntävät vektoritiiliä hieman toisistaan poiketen sillä vektoritiilistä ei vielä toistaiseksi ole olemassa avointa paikkatietostandardia. Selkeän ja yhtenäisen standardin puuttuminen vaikeuttaa vektoritiilien hyödyntämistä, joka ilmenee kuvan muodostamisessa asiakasohjelmassa. Jokainen asiakasohjelma toimii toisistaan poiketen kartantyylittelyn suhteen eli miten valitaan karttasymbologia kullekin vektorille. Tutkimuksen aikana kokeiltiin hyödyntää Geoserverin vektoritiiliä usealla asiakasohjelmalla, mutta testaamiseen sopiviksi osoittautuivat Leaflet- ja Openlayers-kirjasto. Kummankin kirjaston etuna on se, että ne ovat web-pohjaisia, selaimella käytettäviä ja avoimen lähdekoodin kirjastoja, jotka mahdollistivat lähdekoodin muokkaamisen. Lisäämällä lähdekoodiin toiminnallisuuksia kuvien esittämiseen käytettyä aikaa pystyttiin vertailemaan rasteri- ja vektoritiilen välillä. Yksi vaihtoehto olisi voinut olla myös Mapbox-GL-JS, joka olisi hyvin todennäköisesti toiminut Geoserverin vektoritiileillä, sillä ne on muodostettu Mapboxin spesifikaation määrittelemällä tavalla. Mapbox-GL-JS ei kuitenkaan ole Vantaan kaupungilla käytössä, joten se päätettiin sulkea pois tästä tutkimuksesta.

Testisovelluksien muodostaminen tapahtui paikallisesti työasemalla ja niiden rakentamisessa hyödynnettiin Webpack-kirjastoa, joka on avoimen lähdekoodin työkalu JavaScript moduulien kokoamiseksi. Samalla Webpack mahdollisti valmiin paikallisen kehityspalvelimen asennuksen lisäosana. Yksi Webpackin yhteydessä usein käytetty kirjasto on Babel, jota hyödynnettiin myös tässä projektissa. Babelia käytetään pääasiassa modernin JavaScriptin kääntämisessä taaksepäin yhteensopivaksi vanhempien verkkoselaimien kanssa. (What is Babel? 2019.) Webpack kokoaa niin myös projektissa olevat omat tiedostot, kuin myös projektiin tuodut moduulit eli ulkoiset kirjastot, ja muodostaa niistä yhden niputetun tiedoston. Tässä tapauksessa karttakirjastot kuten Openlayers ja Leaflet sekä näihin liittyvät moduulit.

5.3.1 Openlayers-testisovellus

Vantaan kaupungilla käytetään Openlayersia jonkin verran web-kartoissa. Openlayers on myös integroituna Geoserveriin, jota voidaan käyttää esimerkiksi Geoserverin web-käyttöliittymällä kartta-aineistojen esikatselussa. Geoserver- ja Openlayers-projektia edistävät työryhmät tekevät myös yhteistyötä, jonka ansiosta näiden kahden ohjelman yhteen sovittaminen on otettu huomioon Openlayersissa. (Blasby & Hocevar 2016.) Openlayersin etuna on myös se, että se on rakennettu OGC-rajapintastandardien ympärille, mutta siihen on oletuksena rakennettu tuki myös Mapboxin vektoritiileille. Web-kehittäjän näkökulmasta rajapinnan dokumentaatio on erinomainen ja perusteellisesti laadittu, joka helpottaa kehitystyötä huomattavasti. Tässä tutkielmassa käytettiin Openlayersin versiota 5.3.1.



Kuva 8. Openlayers Vektoritiili ja WMTS Luokkakaavio. Keltaiset ovat vain WMTS-palvelulle ominaisia luokkia ja punaiset Vektoritiileille.

Kuvassa 8 on esitetty määriteltävien parametrien minimimäärä, jotta karttasovellus pystyy hakemaan karttatiilejä Geoserveriltä ja muodostamaan karttakuvan selaimessa.

Suurimpana haasteena tiilitetyissä karttapalveluissa on määritellä tiiliruudukko eli ”Tile-Grid”-objekti, joka on määritelmä alueesta, josta Openlayers hakee tiilejä. WMTS-palvelu toimii, kun ruudukko on oikein määritelty, mutta vektoritiilien kanssa asia on toinen. Vektoritiileissä ”Source”-objektissa oleva formaatti on toinen objekti nimeltään MVT. WMTS-palvelussa formaatti on merkkijono, joka voidaan suoraan koodata URL-kyselyyn avainarvopari-menetelmällä, mutta objektia ei voida koodata ilman virheilmoitusta. MVT-objekti on oleellinen osa kuvan muodostamiseen vektoriaineistosta, joka syntyy vektori- ja rasteritiilen toimintaperiaatteiden eroavaisuudesta. Virheviestin välttämiseksi, tulee URL-kysely muodostaa siten, että ainoat muuttuvat parametrit ovat mittakaavataso, sekä ruudukon tasokoordinaatit. Helpoin tapa hyödyntää vektoritiiliä on aktivoida Geoserverin TMS-palvelu, jolloin Geoserver vastaa yksinkertaisen TMS-standardin mukaiseen URL-päätepisteeseen.

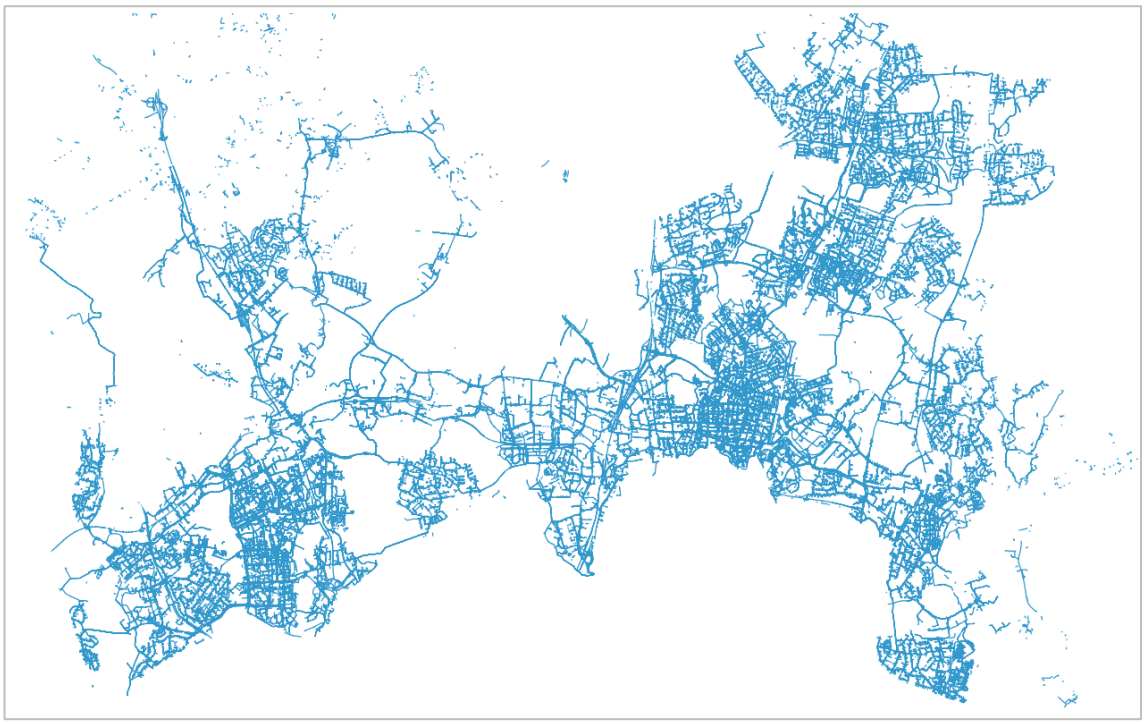
Esimerkkikoodi 3. JavaScript-funktio, joka hakee johtoverkkokohtaisesti karttatason. Funktiota voidaan kutsua antamalla johtoverkolle parametreiksi nimi GetCapabilities-dokumentin mukaan sekä tyyliparametrit.

```
import { Fill, Stroke, Style } from "ol/style";
import VectorTileLayer from "ol/layer/VectorTile";
import VectorTileSource from "ol/source/VectorTile";
import { MVT } from "ol/format";
import { createXYZ } from "ol/tilegrid";

const johtoverkkoLayer = (verkko, fillColor, strokeColor, width) => {
    projection: "EPSG:3857",
    tileGrid: createXYZ({
      extent: [-20037508.34, -20037508.34, 20037508.34, 20037508.34],
      maxZoom: 20,
    }),
    format: new MVT(),
    url: `http://localhost:8083/geoserver/gwc/service/tms/1.0.0/johto-
kartta:${verkko}/EPSG:3857/{z}/{x}/{-y}.pbf`
  });


  return new VectorTileLayer({
    style: new Style({
      fill: new Fill({
        color: fillColor
      }),
      stroke: new Stroke({
        color: strokeColor,
        width: width || 1
      })
    }),
    source
  });
};
```

Toinen ero on vektoritiileissä käytettävä createXYZ-metodi, joka toimii hieman eri tavalla, kuin WMTS-palvelun ruudukko. Rasteriaineistoissa on oleellista määritellä kuvien resoluutio, mutta vektoriaineistossa resoluutio riippuu siitä, miten kuva määritellään eli missä asennossa kamera on suhteessa vektoreihin. Koska vektoritiilit eivät sisällä oikeita koordinaatteja lainkaan, ne olettavat, että koodinpurkaja tietää yksittäisen vektoritiilin sijainnin suhteessa karttanäkymään. Tämän johdosta createXYZ-metodi ei sisällä resoluutiomääritystä lainkaan, vaan se laskee resoluution automaattisesti ”Extent”-määrityksestä, eli aluerajauksesta tai käyttämällä kartan oletuksena olevaa koordinaattisysteemiä. Kummassakin tapauksessa aluerajauksen on oltava koordinaattijärjestelmä, jossa vektoritiilet ovat julkaistu, jotta sijainti ja mittakaava saadaan oikein asetettua. Tämän tutkielman liitteessä 1 on lähdekoodista poimittu JavaScript-tiedosto, jossa kartta muodostetaan ja suorituskykyä seurataan.



Kuva 9. Geoserver-vektoritiilien esikatselunäkymä

Vektoritiileissä Openlayers esittää oletustyylinä kaiken sinisenä yhden pikselin paksuisena. Yksinkertaisimmillaan tyyli voidaan määritellä sen lähteen mukaan, joka toteutettiin myös testisovelluksessa. Jokainen väri on valittu olemassa olevien WMS-tasojen tyylien mukaan, jossa jokaista johtoverkon lajia kuvaa tietty väri. Esimerkiksi sähköverkko

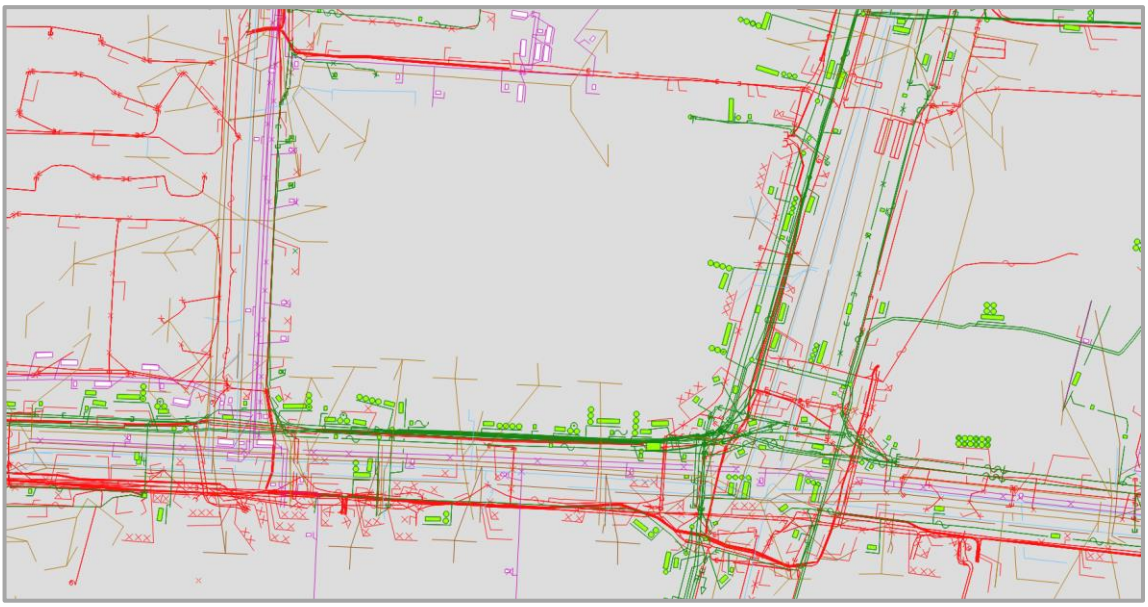
on punainen yhden pikselin paksuinen, televerkko vihreä yhden pikselin paksuinen ja kaasuverkko oranssi kahden pikselin paksuinen jne.

5.3.2 Leaflet-testisovellus

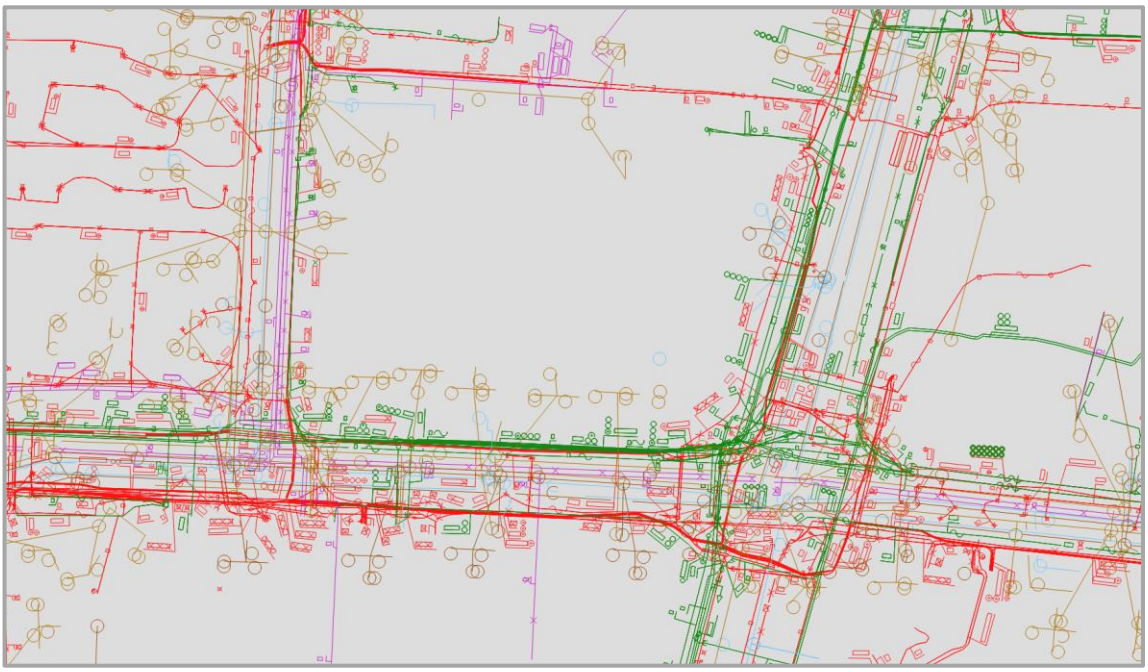
Vantaan kaupungin johtokartoituksia hallinnoidaan tällä hetkellä web-käyttöliittymällä, johon on upotettuna myös karttasovellus, joka toimii Leaflet-kirjastolla ja hakee karttarajapinnasta johtokartat WMS-standardin mukaisesti. Leaflet on myös käytössä muissa vastaavissa sovelluksissa Vantaan kaupungilla. Leaflet WMS-palvelu saa usein palautetta hitaasta toiminnastaan ja huonosta käyttäjäkokemuksesta. Jotta suorituskykyä saataisiin parannettua, on myös Leaflet sisällytetty tähän tutkimukseen.

Leaflet on hyvin kevyt kirjasto interaktiivisten web-karttapalvelujen rakentamiseen. Se on avoimen lähdekoodin JavaScript-kirjasto, joka on suunniteltu mobiiliystävälliseksi. Leaflet on myös melko hyvin dokumentoitu, mikä helpottaa sen käyttöönottamista. Leaflet-kirjasto on keveytensä johdosta hyvin suppea, mikä tarkoittaa sitä, että itsenäisenä kirjastona se ei tarjoa paljoa ominaisuuksia. Leafletin ympärille on kuitenkin syntynyt yhteisö, joka on rakentanut joukon lisäosia ongelman ratkaisemiseksi. Taustakarttojen hakemiseksi karttarajapinnasta, tarjoaa Leaflet-kirjasto tuen WMS-standardille sekä tiilitystä tukeville karttapalveluille TMS-standardin mukaisen tuen. Geoserver kykenee palvelemaan karttakuvia näiden kummankin standardin mukaisesti, mutta vektoritiilien osalta täytyi Leflet-kirjastossa käyttää lisäosaa kuvan muodostamiseksi. Lopullisessa testisovelluksessa haettiin vektoritiilet karttarajapinnasta TMS-standardin mukaisilla kyselyillä ja kuva muodostettiin Leaflet VectorGrid-lisäosalla, joka on saatavilla avoimena lähdekoodina Githubista (Leflet.VectorGrid 2019).

Leaflet.VectorGrid-lisäosalla on myös riippuvuuksia muihin avoimen lähdekoodin kirjastoihin. Tässä tutkielmassa rakennetun testisovelluksen käytön aikana yhden ulkopuolisen kirjaston sijainti muuttui, joka rikkoi testisovelluksen. Rikkoutunut sijainti vaikutti VectorGrid-lisäosaan. VectorGrid-lisäosan yhteisö ei kuitenkaan ole enää aktiivinen, joten ongelman korjaukseksi lisäosa tuotiin kokonaan osaksi testisovelluksen lähdekoodia sen lisenssiehtojen mukaisesti. Tästä oli myöhemmin myös se etu, että lähdekoodiin pystyttiin rakentamaan metodi kuvanmuodostukseen käytetyn ajan seuraamiseksi.



Kuva 10. Openlayers-vektoritiilit Tikkurilassa tasolta 19.



Kuva 11. Leaflet-vektoritiilit Tikkurilassa tasolta 19.

Vektoritiilien tyylittely toimii Leafletin puolella eri tavalla kuin Openlayersin puolella. Leaflet lisäosassa ei ole sisällytetty oletustyyliä, eikä tyyliä voida määritellä tietolähteen mukaan. Tyylittely tapahtuu aineiston ominaisuustietojen perusteella, mikä vaatii perehtymistä alkuperäisen aineiston ominaisuustietoihin. Toinen karttakuvan lopputulokseen

vaikuttava tekijä vektoritiilien suhteen on Leaflet.VectorGrid-dokumentaatiossa mainittu Leafletin vakio-ominaisuus muodostaa vektoriaineistoja joko L.canvas.tile- tai L.svg.tile-metodeilla (Leaflet.VectorGrid API Reference 2019). Oletusasetuksena on L.svg.tile-metodi, joka tekee kartasta lähes käyttökelvottoman nostaen karttasovelluksen vasteajat suurilla mittakaavoilla erittäin pitkäksi. Kumpikin metodi käyttää edellä mainitun lisäosan dokumentaation mukaan kartan piirtämiseen html-canvas-elementtiä, mutta kun kuvan muodostusmetodi vaihdetaan L.canvas.tile-metodiin, muuttuu Leaflet-kirjastolla toteutettu testisovellus käyttökelpoiseksi. Tämä havainto on tehty Vantaan kaupungin johtoverkkoaineistolla.

5.4 Suorituskykyvertailu asiakasohjelmassa

Karttapalvelun vasteaika on se aika, joka karttakuvalla kestää päivittyä, kun kuvaa vieritetään tai mittakaavaa muutetaan. Vasteaika syntyy siitä, kun asiakasohjelma tekee kyselyn palvelimelle, joka lähettää paluuvastauksena karttatiilen, minkä asiakasohjelma esittää käyttäjälle. Tässä osiossa mitataan asiakasohjelman suorituskykyä vertaamalla aikaa, joka menee vektoriaineiston muodostamiseksi karttakuvaksi verrattuna rasteriaineistoon. Vektoriaineisto muodostetaan kuvaksi vasta asiakasohjelmassa, kun taas rasterikuva on valmis esitettäväksi sellaisenaan, joten hypoteesin mukaan vektoriaineistolla kuluu aikaa kuvan muodostamiseen enemmän. Tutkimusmenetelmänä on käyttää testisovelluksia, joissa kumpaankin testisovellukseen on toteutettu mekanismi, jolla voidaan seurata yksittäiseen karttatiileen esittämiseen käytettyä aikaa. Tiilien esittämiseen käytetty aika tilastoitiin myöhempää analysointia varten.

Yksittäisen tiilen muodostuksen monitorointiin pyrittiin kummastakin JavaScript-kirjastosta löytämään tehtävään soveltuvat työkalut. Kummastakin löytyi tapahtuma, joka käynnistyy, kun yksi tiili on saatu valmiiksi näkyviin karttakuvaan. Tapahtuman seurauksena saadaan yksittäinen karttatiili tarkasteltavaksi, mutta kuitenkin vain Openlayers sisälsi tiileillä muuttujan, jonka arvoksi asetetaan aikaleima siinä vaiheessa, kun kuvaa aletaan muodostamaan (esimerkkikoodi 4). Leafletissa vektoritiilin muodostus tapahtui lisäosassa, joka oli liitetty suoraan projektiin, joten vektoritiilen muodostuksesta vastaavaan funktioon oli helppo upottaa oma muuttuja ja asettaa sille arvoksi JavaScriptin standardikirjaston sisältämä aikaleima muodostus funktion alussa. Vastaavanlaisen

aikaleiman lyöminen ei onnistunut Leafletin TMS-palvelun puolella rasteriaineistoilla, tai se olisi ollut hankalaa toteuttaa.

```
const mittausTulokset = [];
map.on("postrender", event => {
  const renderTime = Date.now() - event.frameState.time;
  mittausTulokset.push(renderTime);
});
```

Esimerkkikoodi 4. JavaScript: postrender-tapahtuma, jolla Openlayersissa seurataan yksittäisen tiilen muodostamiseen käytettyä aikaa ja tallennetaan se välimuistiin.

Kun yksittäisten tiilien monitorointi oli saatu toteutettua, oli työn seuraava vaihe toteuttaa karttasovelluksiin joukko komentoja, joita voitiin automatisoidusti toistaa useita kertoja tilaston keräämiseksi. Komennot ovat JavaScriptin listaobjektiin kerättyjä xy-tasokoordinaatteja sekä z-koordinaatti mittakaavatason määrittelyyn. Jokainen listaobjektin alkio vastaa parametrejä yhden näkymän muodostamiseen. Näkymät vaihdettiin automaattisesti ajastetun funktion avulla. Ajastettu funktio toimi siten, että se käynnistyi aina uudelleen jokaisen postrender-tapahtuman jälkeen. Kun uusia kuvia ei ole enää muodostunut 1000 millisekuntiin, siirrytään karttapalvelussa uuteen näkymään. Näkymät on valittu siten, että kaikki järkevät mittakaavatasot kokeillaan läpi ja kuvaa vieritetään myös samalla. Lopputuloksena seurataan sitä, kuinka monta karttatiiltä käytettiin yhteensä näkymien muodostamiseen, kuinka kauan keskimäärin yhden karttatiilen esittämiseen meni aikaa millisekunneissa ja kuinka kauan kesti käydä läpi kaikki 23 testinäkymää.

Taulukko 7. Yhteenvetotaulukko asiakasohjelmissa suoritetuista mittaustuloksista. Tarkat mittaustulokset liitteessä 2.

Karttakuvan muodostus asiakasohjelmassa				
nro	kartta	kierros	keskiarvo	keskihajonta
1	Leaflet TMS	52.7 sec	-	-
2	Leaflet VCT	1 min 10,7 sec	15 ms	29 ms
3	Openlayers WMTS	1 min 17,8 sec	4 ms	4 ms
4	Openlayers VCT	1 min 19,7 sec	9 ms	27 ms

Jokainen mittaustulos (taulukko 7) toistettiin kymmenen kertaan, joista yhteenvetotaulukko muodostuu. Toistot suoritettiin samalla tietokoneella, näytöllä ja karttaikkunan koko

pidettiin jatkuvasti samana 1920x1000 pikselinä. Mittaustuloksissa esiintyvä kierros vastaa aikaa, joka kesti kaikkien näkyminen muodostamiseen yhteensä. Kierrosaika ei ole suoraan verrannollinen tiilien muodostamiseen käytettyyn aikaan, vaan suurempi tekijä on karttakyselyjen tekemiseen ja paluuviestin vastaanottamiseen kulunut aika. Asiakasohjelmien välillä on selvästi eroja, mikä voi viitata myös siihen, että niissä suoritetut AJAX-kyselyt on suunniteltu eri tavalla, mikä johtaa suorituskyvyltään erilaiseen koodiin. Leaflet-vektoritiileissä kymmenessä otannassa havaittiin myös korrelaatiota kierrosaikojen ja keskimääräisen tiilenmuodostamiseen käytetyssä ajassa, kun korrelaatiokerroin on hieman yli 0.5 ja kulmakertoimen p-arvo on alle 0.004 (liite 2). Se miksi kierrosaika pienenee, kun karttatiileen muodostusaika kasvaa, johtuu mahdollisesti siitä, että hetkellisesti hyvän verkkoyhteyden tai karttapalvelun ruuhkattomuuden ansiosta asiakasohjelma saa nopeammin karttatiiliä. Kun karttatiiltä tulee nopeammin, joutuu asiakasohjelma prosessoimaan enemmän karttatiiliä kerralla, jolloin ne joutuvat jonoon tai prosessointiteho jakautuu tiilien välillä.

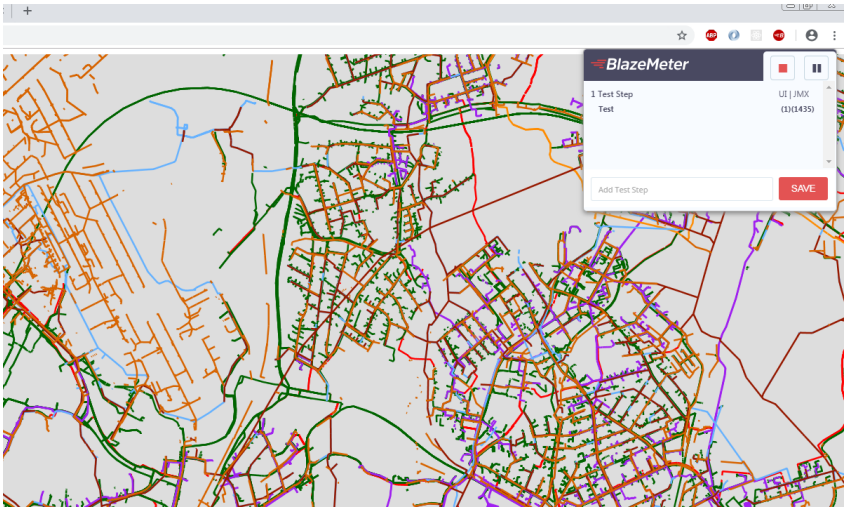
Yleinen käyttäjäkokemus on hieman parempi Leafletin puolella kuin Openlayerssin puolella. Suurilla mittakaavoilla kummassakin ohjelmassa rasteritiilet tuntuvat toimivan tehokkaammin kuin vektoritiilet, mutta pienillä mittakaavoilla eroa ei huomaa. Taitepiste tuntuu olevan noin mittakaavan 1:20 000 kohdalla, jolloin vektoritiilet toimivat yhtä tehokkaasti kuin rasteritiilet. Tämä johtuu siitä todennäköisesti, että rasteritiileillä johtojen yleistäminen on tehokasta. Rasteritiilet perustuvat bittikarttoihin, jossa alueelle on valittu tietty määrä alkioita, jotka voivat muuttua, joten alkionväri tulee lopulta päättää. Vektoriaineistolla puolestaan ei ole teoriassa rajoituksia sillä, kuinka moni alkio pystyy leikkaamaan toisiaan tai olemaan päällekkäin. Johtoverkko on vektoritiilien kannalta haastava aineisto, sillä alkioiden määrä on valtava ja tiedostokoot kasvavat ilman hyvin suunniteltua yleistystä. Rasteritiileissä puolestaan johtoverkoille tyypillinen tapa vetää johdot yhdessä linjassa jättää paljon tyhjää tilaa tiedostoon.

5.5 Palvelimen suorituskyvyn vertaileminen

Karttasoveluksessa käyttäjäkokemukseen suurimpana vaikuttavana tekijänä voidaan yleisesti pitää aikaa, joka kestää asiakasohjelmalla tehdä pyyntö karttapalveluun ja saada paluuvastauksena karttakuva. Palvelimen suorituskykyä testattiin tässä tutkielmassa Javalla rakennetun avoimen lähdekoodin suorituskykyä testaavalla työkalulla

JMeterillä. Palvelimen suorituskyvyn testaus tapahtuu siten, että JMeter lähettää palvelimelle HTTP-kyselyitä, joista se kykenee laskemaan paluuviestin saamiseen kestäneen vasteajan ja tiedoston kuorman.

Ennen kuin palvelimen suorituskykyä kyetään JMeterillä testaamaan, tarvitaan valmiit HTTP-kyselyt. Valmiita kyselyitä nauhoitettiin Googlen Chrome-selaimeen asennettavalla BlazeMeter-lisäosalla, joka on saatavilla Chrome web storesta. BlazeMeterillä voidaan nauhoittaa kaikki HTTP-kutsut, mitä selaimella tehdään nauhoituksen ollessa päällä. BlazeMeterin etuna on myös se, että nauhoitus voidaan tallentaa JMX-tiedostoon, joka on JMeterin oma formaatti. BlazeMeterin käyttö on hyvin suoraviivaista. Testiaineistona muodostettiin käynnistämällä sama automatisoitu prosessi, joka muodosti näkymät asiakasohjelman suorituskykytestauksien yhteydessä. BlazeMeter käynnistettiin, minkä jälkeen ajettiin karttasovelluksen testinäkymät samalla nauhoittaen kaikki HTTP-kutsut. Kun kaikki näkymät oli läpikäyty, nauhoitus lopetettiin ja tulokset tallennettiin JMX-muotoon. Lopputuloksena oli kutakuinkin täysin samat kyselyt kuin asiakasohjelman testeissä.

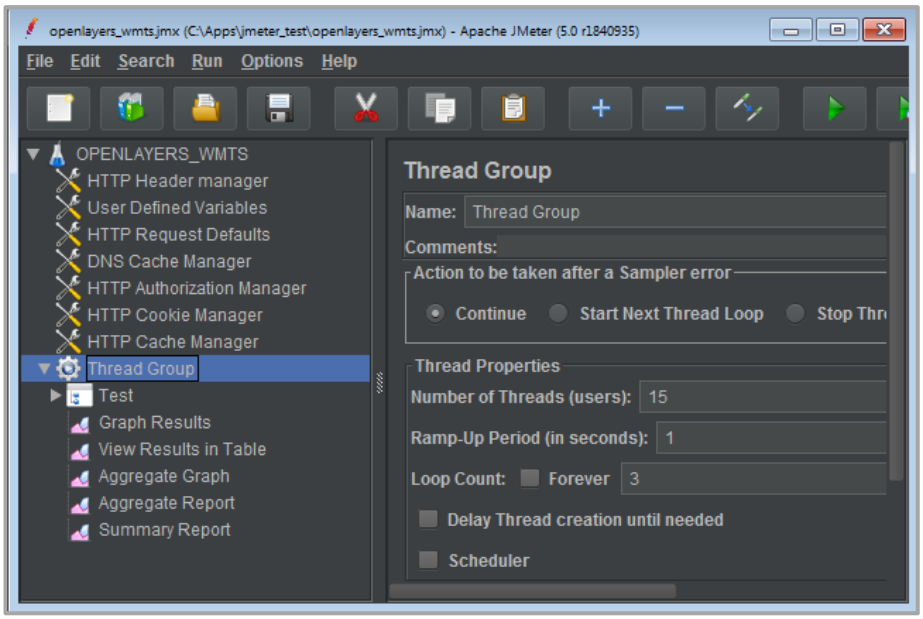


Kuva 12. BlazeMeter ilmestyy Crome-selaimen työkaluihin.

JMeter on avoimen lähdekoodin projekti ja toimii Apache-palvelinohjelman päällä, joka on myös avoimen lähdekoodin projekti. Apache tarvitsee toimiakseen Java 8-asennuksen koneelle. Tämän jälkeen JMeter voidaan käynnistää projektikansiossa olevasta bat-tiedostosta käsin, mikä avaa graafisen käyttöliittymän. Käyttöliittymästä avataan

BlazeMeterillä nauhoitettu JMX-tiedosto, jonka pohjalta voidaan suunnitella suorituskykytesti. JMX-tiedosto sisältää kaikki HTTP-kyselyt, joista poistettiin kaikki muut kyselyt paitsi karttakyselyt Geoserverille.

JMeterissä simuloidaan palvelimen kuormittavia käyttäjämääriä määrittelemällä ”Number of Threads”-parametrin (kuva 13). Tutkimus suoritettiin tavallisella kannettavalla tietokoneella, jossa viidentoista käyttäjän simulointi vaikutti järkevältä ratkaisulta. Suurempien käyttäjämäärien simulointi johtaa hyvin pitkiin kierrosaikoihin sekä JMeterin kaatumiseen. Käyttäjämäärien lisäksi voidaan määritellä, kuinka monta kertaa testinäkymät käydään läpi käyttäjää kohden, mikä oli tässä testissä asetettu kolmeen kertaan. Viimeinen asetus on se, kuinka nopeasti JMeter ottaa kaikki testikäyttäjät käyttöön. Tässä haluttiin simuloida yhtäaikaisia käyttäjiä, joten parametri asetettiin niin, että kaikki viisitoista käyttäjää ottavat yhteyttä Geoserveriin aina sekunnin sisällä. Oikeasti suurten suorituskykytestien luomiseksi taritaan testin hajauttaminen usealle eri koneelle, mutta tähän vertailuun riittää yksi kone ja viisitoista käyttäjää.



Kuva 13. JMeter-käyttöliittymässä käyttäjien simulointia.

Tuloksia JMeterissä voidaan kerätä asettamalla kuuntelijoita, joista asetettiin seuraavat kaksi: ”summary report” ja ”aggregate report”. Edellä mainituista raporteista ensimmäinen laskee kaikki kyselyt yhteen ja laskee keskiarvon ja keskihajonnan. Sama raportti

myös laskee kaikkien kyselyiden yhteisen kuorman, joka lasketaan vain kerran kyselyä kohden. Toinen raportti on nimensä mukaan suomennettuna yhdistelmäraportti, joka laskee ensin yksittäisen kyselyn keskiarvon, joka toistuu simuloitujen käyttäjien sekä kierrosten johdota. Tämän jälkeen kyselyiden keskiarvoista muodostetaan yhdistetty keskiarvo. (Bhatt 2017.) Raportti laskee myös yhdensuuntaiset 99-, 95- ja 90-prosentin luottamusvälit.

Taulukossa 8 on esitetty yhteenveto mittaustuloksista, josta osoittautuu poikkeavan alkuperäisestä hypoteesista vektoritiileistä. Hypoteesin mukaan vektoritiilet vaativat keskimäärin vähemmän muistia, kuin rasteritiilet, joten tästä johtuen tietoliikenneverkossa vektoritiilien lataaminen on nopeampaa. Yhteenvetotuloksissa vektoritiilet ovat keskimäärin 18,38 kilotavua ja rasterit 16,36 kilotavua. Lopputuloksena keskimääräinen rasteritiili latautuu 114 millisekunnissa, kun taas vektoritiili 132 millisekunnissa. Aggregate-raportti osoittautui rasteritiilien osalta hyvin samanlaiseksi kuin summary-raportti. Vektoritiilien osalta asia onkin toinen. Siinä missä rasteritiilet ovat hyvin vakaita, vektoritiilien latausajat heilahtelevat hurjasti.

Taulukko 8. JMeter-mittausten yhteenveto. Kaikki tulokset liitteessä 3.

JMeter mittaustulokset		
Formaatti	PNG	VCT
Keskiarvo	114 ms	132 ms
Keskihajonta	3289 ms	3984 ms
Kuorma	26,98 Mt	32,97 Mt
Keskimääräinen kuorma	16,36 Kt	18,38 Kt
Kokouman keskiarvo	114 ms	66 ms
Mediaani	45 ms	40 ms
95%-luottamusväli	123 ms	189 ms

Mittaustulokset vastaavat käyttäjäkokemusta testisovelluksissa, mutta kuormien suhteet poikkeavat GeoWebCachen muistiin tallennettujen tiilien suhteesta. Taulukossa 6 on listattu välimuistiin tallennettujen rasteri- ja vektoritiilipyramidien tilavaativuus. Rasteritiilet veivät pienimmillään sähköverkossa kaksi kertaa enemmän tilaa kuin vektoritiilet ja

suurimmillaan kaasuverkossa sata kertaa enemmän. Selitys siihen, miksi kuitenkin vektoritiilet vievät enemmän muistia asiakasohjelmassa johtuu siitä, että suurilla mittakaavoilla vektoritiilet ovat liian suuria siirrettäväksi nopeasti tietoliikenneverkossa. Suurimmillaan GeoWebCachen välimuistin vektoritiilet ovat yli kolme megatavua mittakaavatasolla 12. Rasteritiilien kohdalla tilavaativuuden vaihteluväli on huomattavasti maltillisempi.

6 Päätelmät

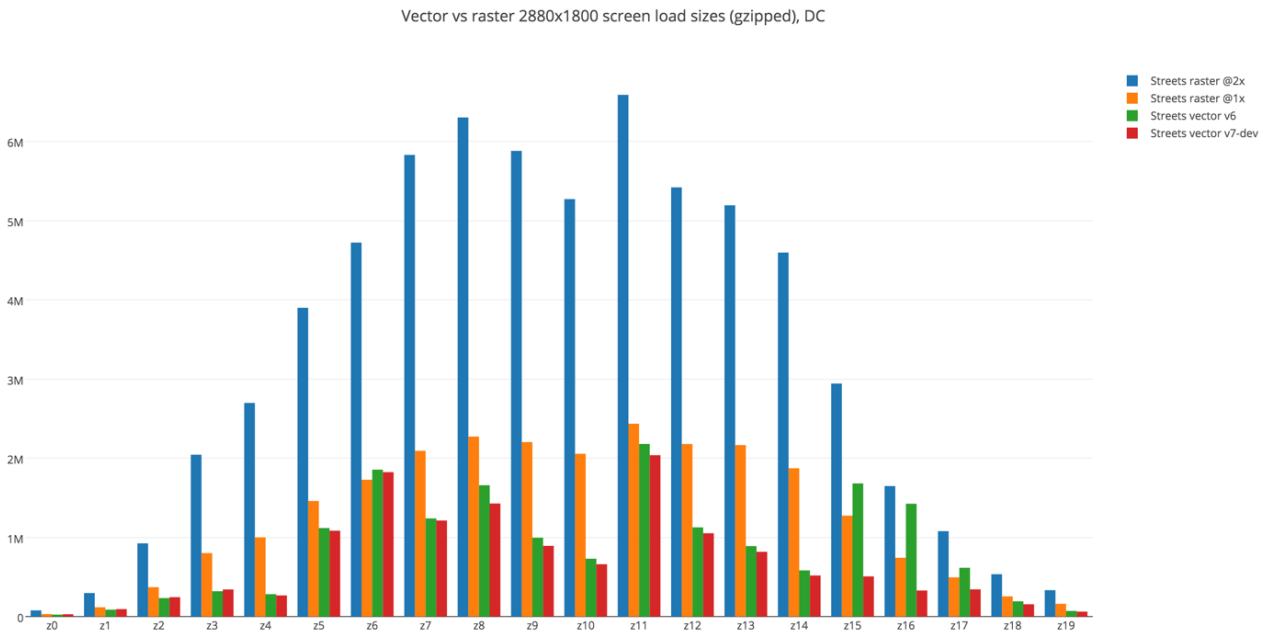
Tämän tutkimuksen nojalla vektoritiileillä ei saavutettu suorituskyvyn eikä ylläpidon osalta etuja. Kun tutkimuksessa käytettyjen testisovellusten avulla muodostetuista tuloksista tutkitaan kierrosaikoja, käy ilmi karttanäkymien keskimääräinen muodostusaika. Muodostaminen on nopeimmillaan rasteritiilitekniikalla 1,2 sekuntia, kun taas vektoritiileillä aika on hitaimmillaan 2,5 sekuntia. Tuloksia voidaan verrata luvussa 5 läpikäydyn Nielsenin (1993) tutkimuksiin. Vaikka johtoverkoilla ei saavutettu vektoritiilitekniikalla suorituskyvyllisiä etuja, ei se kuitenkaan tarkoita sitä, etteikö toisenlaisilla aineistoilla se olisi mahdollista.

6.1 Vektori- ja rasteritiilien vertautuvuus

Tämän tutkimuksen motivaationa oli tutkia sitä, kuinka vektoritiilet vertautuvat rasteritiileihin ja pystyvätkö vektoritiilet suoriutumaan samoissa olosuhteissa kuin rasteritiilet. Rasteritiilien suurin etu vektoritiileihin verrattuna on se, että ne ovat valmiiksi muodostettu ja helppokäyttöisiä. Jotta vektoritiilet yltäisivät yhtä hyvään suorituskykyyn, on niiden saavutettava nopeusetu tiedonsiirrossa ja suoriuduttava mahdollisimman hyvin kuvan muodostuksesta. Kuvan muodostus vie selvästi muutaman millisekunnin enemmän vektortiileiltä kuin rasteritiileiltä, mutta se ei kuitenkaan muodostu pullonkaulaksi webkarttojen suorituskyvylle. Vektoritiili ei kuitenkaan osoittautunut tämän tutkielman nojalla suorituskyvyn osalta paremmaksi sen suuren tiedostokoon vaihtelun vuoksi. Samaan lopputulokseen vektoritiilien tiedostokokojen suuresta vaihtelusta ovat päätyneet tapaustutkimuksessaan ”Implementation of Tiled Vector Service: Case Study” Ingensand ym. (2016). Rasteritiedostoilla on olemassa tietyt pikselit, joille on mahdollista antaa arvo, kun taas vektoreissa ei ole olemassa rajoitusta.

Tässä tutkimuksessa kohteena oli johtoverkko, joka edellä mainitusta syystä voi olla hieman tehokkaampaa muodostaa rasteritiileillä. Kun kyseessä on esimerkiksi tavallisesta opaskartasta, jossa kartan symbologia on suurpiirteisempää, on sen esittäminen tehokkaampaa vektoreilla. Esimerkiksi suuret aluegeometriat vievät vektoreina huomattavasti vähemmän tilaa, kuin rastereina. Tästä on esimerkkinä Mapboxin taustakarttapalvelu. Mapbox muodostaa oman aineistonsa Openstreetmapin aineiston pohjalta ja julkaisee siitä sekä vektori-, että rasteritiilipalvelun (Agafonkin 2015.) Mapboxin aineistossa

vektoritiilen on havaittu vievän vähemmän muistia kuin rasteritiilien jokaisella mittakaavatasolla muodostetulla näkymällä (kuva 14).



Kuva 14. Mapboxin tiedostokokojen vertailua mittakaavatasoittain (Agafonkin 2015).

6.2 Suorituskyvyn parantaminen

Selkeän karttakuvan muodostaminen vaatii lähtömateriaalin pelkistämistä kummassakin formaatissa sekä vektori- että rasteriformaatissa (Taraldsvik 2012: 18). Vektoriformaatissa pelkistämisellä on kartan selkeyden lisäksi myös suuri merkitys suorituskykyyn. Pelkistämiseen on olemassa useita algoritmeja, jotka saattavat johtaa hieman toisistaan poikkeaviin lopputuloksiin. Visvalingamin algoritmi on tunnettu geometrian yleistämisessä. Siinä annetaan jokaiselle pisteelle painoarvoa sen ympäröivien pisteiden määrän suhteen. Lopulta voidaan määritellä parametri lukuarvoan, jolla päätetään kuinka monta pistettä voi esiintyä pinta-alanyksikköä kohden. (Davies 2014.) Toinen tunnettu algoritmi on Douglas-Peuckerin algoritmi, jossa geometria pelkistetään ensin valitsemalla kaksi pistettä, jonka välimatka on kaikkein pisin. Tämän jälkeen tutkitaan, onko pisteiden välillä muodostuvan viivan uloimmainen piste niin lähellä, että se kannattaisi pelkistää. (Puchka

2016.) Tätä algoritmia käyttävät esimerkiksi hyödyksi Geoserverin vektoritiili lisäosan lähdekoodissa (PipelineBuilder.java 2018) sekä Mapboxin Tippecanoe-työkalu vektoritiilien muodostamiseen terminaalissa (geometry.cpp 2018). Näistä kahdesta vektoritiiliä muodostavasta työkaluista vain Tippecanoe antaa käyttäjänsä vaikuttaa geometrian pelkistämiseen vaikuttaviin parametreihin.

Toinen pelkistämisen muoto on säädellä karttasymbolien määrää mittakaavasta riippuen. Esimerkiksi yleinen käytäntö on esittää rakennetut alueet suurilla mittakaavoilla yksivärisellä alueella ja rakentamattomat alueet toisella. Kun mittakaava alittaa 1:10 000, esitetään rakennetuilla alueilla myös rakennusten sijainnit aluein tai ääriviivoin.

Tässä tutkielmassa hyödynnettiin Geoserverin vektoritiilejä, jotka ovat osana WMTS-palvelua. WMTS muodostaa kuvat valmiiksi, jolloin karttasymbologia muodostetaan OGC:n SLD-tyylistandardien mukaan. Vektoritiilit ovat Geoserverin osalta suhteellisen uusi teknologia, eikä niiden käyttöä ole vielä erityisen hyvin dokumentoitu. Blasby, joka on toimitut Geoserver-projektin kehittäjänä, esittelee vektoritiilejä FOSS4G-messuilla. (Blasby & Hocevar 2016). Esityksessään hän kuvailee rasteri- ja vektoritiilien muodostusta lähes samankaltaisena prosessina lukuun ottamatta viimeistä tiedoston tallennusta. Tämän esityksen pohjalta käy ilmi, että vektoritiilejen osalta tiedostokokoa voitaisiin ehkä säädellä SLD-tyylien avulla. Teorian pohjalta suoritettiin lisätutkimusta, jossa sähköverkossa rajattiin aineistoa siten, että vain suurjännitekaapelit näkyivät kartalla suuremmilla mittakaavoilla kuin 1:8 000. Tämän avulla saatiin yli 3 000 kilotavua olevat vektoritiilet pienennettyä muutamiksi kymmeniksi kilotavuiksi.

Geoserverin osalta vektoritiilien ollessa WMTS-palvelun lisäosana, tuntuu luontevalta ajatella kumpaakin teknologiaa hyvin samanlaisena. Johtoverkkojen tapaisten aineistojen suhteen voisi olla järkevää käsitellä ne kokonaan eri julkaisuna, jotta kummallekin aineistolle voidaan määritellä omat SLD-tyylitiedostot. Rasteritiileille SLD-tyylien luonti on johtoverkkojen kaltaisilla runsaasti päällekkäistä geometriaa sisältävien aineistojen osalta hyvin anteeksi antavaista. Vektoritiilien osalta täytyy kiinnittää huomiota siihen, että saadaan optimoitua tiedostokoot säilyttäen kuitenkin kaikki tarvittava geometria. Vektortitiilien osalta se vaatii huomattavasti lisää työtä määritellä SLD-tyyli sekä asiakasohjelman vaatimat tyylit, mutta näiden avulla voidaan saavuttaa parempi suorituskyky. Lopputuloksen osalta täytyy olla tietoinen loppukäyttäjän tarpeista. Johtoverkkojen

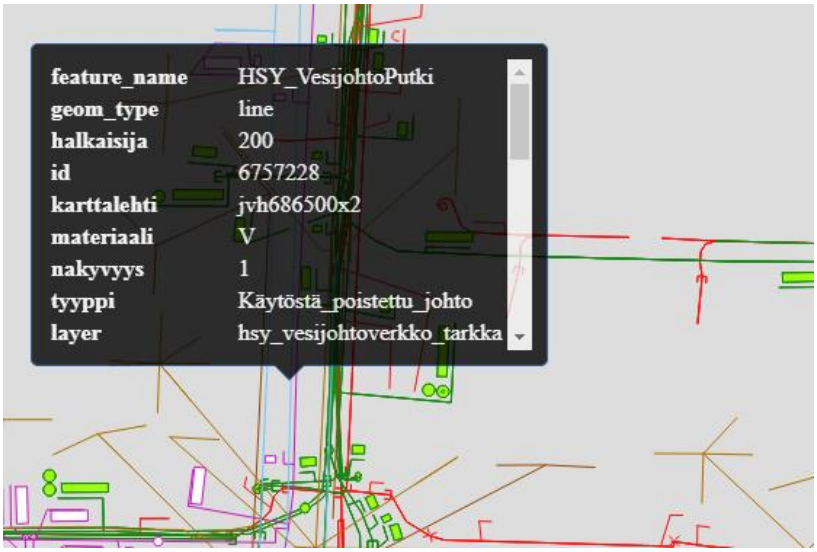
osalta voisi esimerkiksi kadunristeyksen näkyessä kokonaan ruudulla olla myös aineisto kokonaisuudessaan näkyvissä.

Vektoritiilin avulla voitaisiin parantaa suorituskykyä myös karttakyselyiden suhteen johtoverkkoaineistossa. Tällöin kaikki seitsemän verkkoa voitaisiin yhdistää samaan palveluun, joka on tavallisesti yleisin tapa katsella aineistoa. Tasojen ulosrajaaminen voisi kuitenkin olla vielä mahdollista asiakasohjelman puolella tarpeen tullessa. Tämän avulla saataisiin seitsemällä johtoverkkotasolla karttakyselyjen määrä pienennettyä yhteen seitsemäsosaan nykyisten kyselyiden määrästä.

6.3 Vektoritiileille ainutlaatuiset ominaisuudet

Vektoritiilet myös sisältävät attribuutteja, joten ennen kuin vektoritiilet muodostetaan kannattaa turhat attribuutit poistaa. Attribuutit voivat olla myös etu rastereihin nähden, mutta esimerkiksi geometriasta riippuvaisiin analysointeihin vektoritiilet eivät sovellu samalla tavalla kuin vaikka WFS-rajapinta (Blasby & Hocevar 2016). Tämä johtuu siitä, että vektoritiilet sisältävät lisäalueen, jotta lopullinen karttakuva olisi saumaton. Tämän tutkielman johtokarttojen pohjalta ei välttämättä ole tarpeen tehdä geometrian suhteen analysointeja, mutta vektoritiilien sisältämien attribuuttien esille tuominen hiirellä klikkaamalla voi olla hyödyllistä (Kuva 15).

Suurin etu vektoritiileillä on kuitenkin järjestelmän skaalautuvuudessa karttasymbologian suhteen. Johtoverkkojen kohdalla Vantaan kaupungilla on vakiintuneet väriteemat jokaiselle seitsemälle johtolajille. Mahdollisuus kuitenkin ylläpitää yhtä aineistoa, jonka pohjalta voidaan tehdä useita eri tavalla tyyliteltyjä karttoja, on kiistaton etu monissa muissa aineistoissa.



Kuva 15. Openlayers: hiiren klikkauksella saadaan tuotua attribuutit kartalle.

6.4 Vektoritiilet käytännössä

Vektoritiileihin perustuva teknologia on tämän hetken paikkatietojärjestelmissä tiedon jakamisen osalta eniten kehityksen alla olevia teknologioita. Paikkatiedon jakamisessa tiilitystä tukevat karttapalvelut vaativat enemmän konfigurointia kuin esimerkiksi kuvapalvelut kuten WMS. Vektoritiili tuo tyylittelyn myötä lisää haasteita kehitystyöhön, mutta tarjoaa myös paljon uusia mahdollisuuksia. Käyttäjän näkökulmasta hyvin toteutetusta karttapalvelusta ei edes välttämättä huomaa, että se on toteutettu vektoritiiliteknologialla. Johtoverkoissa saavutettiin lähes yhtä hyvä suorituskyky vektori- ja rasteritiilillä. Kuvanlaatu oli vektoritiilillä selvästi terävämpi jokaisella tasolla, kun taas rasteritiileillä vasta viimeisellä julkaistulla tasolla päästiin tasolle, jossa yksittäiset johdot selvästi erottuivat toisistaan.

.

7 Vektoritiilien käytön yleistyminen

Web-karttojen kehitys on kulkenut modernien avointen standardien perässä. Näistä ennen kaikkea HTML5-standardi on vaikuttanut siihen, kuinka grafiikkaa voidaan esittää selaimessa ilman erikseen ladattavia sovelluksia. Modernien internet-verkossa toimivien laitteiden selkärankana ovat verkkoselaimet, jotka tarjoavat alustariippumattoman tavan jakaa tietoa ja mediaa. (HTML 5.2 W3C Recommedation 2017.) HTML5 julkaistiin virallisesti vuonna 2014, ja se on tällä hetkellä suositeltu kieli verkkosivujen tekemiseen. HTML5:n myyntivaltti on ennen kaikkea median jakaminen tehokkaasti ja alustariippumattomasti. (Taraldsvik 2012: 25.) Korkearesoluutioinen median jakaminen on ennen kaikkea mahdollista HTML5 mukana julkaistun canvas-elementin ansiosta. Mediadokumentit, kuten kuva, video ja ääni ovat hyvin pitkälle standardoitu binääriformaatissa, joita HTML5 canvas-elementti kykenee muuttamaan kuvaksi. Lisäksi HTML5-standardin myötä vektorigrafiikan käyttö on yleistynyt, mikä on vektoritiilien kehityksen kannalta tärkeä reunaehto.

HTML5-standardissa ei ole otettu kuitenkaan huomioon paikkatiedon käyttötarkoituksia, eikä näin ole tarkoituskaan. Paikkatietojärjestelmät voivat kuitenkin integroitua avoimiin HTML5-standardeihin JavaScriptin avulla. (Taraldsvik 2012: 27.) OpenGL on kehittynyt standardi interaktiivisten tietokonegrafiikoiden esittämiseen. Tätä standardia hyödynnetään niin videopeliteknologiassa, kuin myös graafiseen suunnitteluun tarkoitetuissa tietokoneavusteisissa piirustusohjelmissa ja 3D-mallinnus-työkaluissa. WebGL on canvas-elementtiin rakennettu teknologia, joka paljastaa OpenGL-rajapinnan, mikä mahdollistaa interaktiivisen tietokonegrafiikan tuottamisen myös verkkoselaimella. Avoimet standardit ovat luoneet web-kehittäjäyhteisölle mahdollisuuden suunnitella tuotteita, jotka toimivat alustariippumattomasti. Tavallinen verkkoselain muuntuu nykyään työpöytäsovelluksen tasoiseksi tehokkaaksi työkaluksi.

7.1 Useita vektoritiilitoteutuksia

Tämä tutkielma keskittyi hyvin pitkälti Mapboxin spesifikaation mukaisiin vektoritiileihin sekä näistä tehtyihin toteutuksiin. Vaikka Mapbox onkin suunnitellut omat vektoritiilensä oman tuotteensa käytettäväksi, on se myös herättänyt kiinnostusta laajemmin. Mapboxin vektoritiiliyhteisö kerää ja listaa Githubiin kaikki projektit, jotka hyödyntävät samaa

spesifikaatiota joko palvelimen tai asiakasohjelman puolella (Awesome Vector tiles). Googlen protokollapuskureiden ansiosta Mapboxin vektoritiilet saavuttavat mahdollisimman pienen tiedostokoon minimoimalla redundanssia ja muuntamalla ne binäärimuotoon. Tämä on houkutellut monia kehittäjiä omaksumaan nimenomaan Mapboxin vektoritiilet ainakin osittain järjestelmiinsä. Tällä hetkellä suuri osa järjestelmistä saattavat hyödyntää esimerkiksi vektoritiiliä tiedonsiirron nopeuttamiseksi vaikkeivat vielä hyödyntäisikään WebGL-rajapintaa lopullisen kuvan muodostamisessa. Samaa kartan tiilitykseen perustuvaa tekniikkaa käytetään myös paljon yleisesti tunnetuilla formaateilla kuten GeoJSON ja TopoJSON. Vaikka Google tarjoaakin lähdekoodin protokollapuskureiden purkamiseksi, käytetään yleisesti tunnettuja formaatteja siksi, koska ne ovat ihmisten luettavissa ja täten niiden kanssa on helpompaa työskennellä.

Tiilitettyyn vektoriin perustuvilla menetelmillä on toteutettu paljon myös muita paikkatietojärjestelmiä. OGC on insinööriraportissaan tutkinut viittä eri vektoritiilitoteutusta: Mapbox, Cesium 3D Tiles, Esri I3S, Ecere sekä Geoserver. (Cavazzi 2017.) Kyseisen insinöörityötutkielman tutkii vektoritiilitoteutuksia vaiheittain ja vertailee, kuinka tutkimuksessa tarkasteltavana olleet toteutukset ovat ratkaisseet välivaiheet. Nämä järjestelmät ovat kuitenkin toteutettu vielä toistaiseksi vektoritiilien osalta itsenäisesti ilman standardeja. Usein vektoritiilejä hyödyntävät järjestelmät ratkaisevat vektoritiileillä jonkin järjestelmän toiminnallisuuden kannalta tärkeän osuuden. Esimerkiksi 3D-malleissa vektorimuotoinen aineisto on välttämätön, jotta voidaan luoda interaktiivinen karttapalvelu. HTML5-standardi on ollut merkittävässä roolissa näiden uusien järjestelmien synnyssä.

7.2 Vektoritiilien standardointi

OGC on parhaillaan pilotoimassa vektoritiilejä yhteistyökumppaneittensa kanssa motivaationa luoda uusi OGC vektortiili-standardi (OGC Calls for Participation in Vector Tiles Pilot 2018). Kyseisessä pilotissa tutkitaan erityisesti Mapboxin vektoritiiliä sekä GeoJSON-tiiliä. Tavoitteena on löytää formaatti jo olemassa oleville rajapintastandardeille yhteistyökumppaneiden kesken.

Kolmiulotteisissa paikkatietototeutuksissa tullaan todennäköisesti käyttämään omaa standardiaan kuin kaksiulotteisissa toteutuksissa. Kolmiulotteisille vektoritiileille on jo olemassa suuren suosion saavuttanut standardi. 3D-kaupunkimallien esittämiseen

verkkoselaimessa on mahdollista Cesium 3D-tiilien myötä. (Cozzi 2017.) Cesium 3D tiilet ovat myös avoin standardi siirtää kolmiulotteista paikkatietoa selaimeen binäärimuodossa ja muodostaa kuvat hyödyntäen HTML5-canvas-elementtiä sekä WebGL-rajapintaa.

7.3 Vektoritiilien tulevaisuus

Teoriassa vektoritiileissä on potentiaalia ratkaista vieläkin haastavampia ongelmia. Yksi vaihtoehto syntyy siitä, että vektoreihin on mahdollista koodata myös paljon monimutkaisempia geometrioita kuten kaaria tai Bézier-käyriä (Weistein 2019). Käytännössä vielä hyvin harva paikkatietojärjestelmä tukee suoria viivoja monimutkaisempia geometrioita, joka johtaisi yhteensopivuusongelmiin. Toisaalta taas on olemassa laajasti käytetty standardi, joka tukee myös monimutkaisempia geometrioita. SVG-formaatti on hyvin paljolti käytössä web-sovelluksissa, jossa käytetään graafista suunnittelua.

Tämän tutkimuksen yhteydessä muodostuneen käsityksen mukaan vaikuttaisi siltä, että Mapboxin vektoritiili on saavuttanut suuren yleisön suosion. Monet kaupalliset toimijat sekä avoimen lähdekoodin yhteisöt tutkivat ja kehittävät vektoritiiliteknologiaa. Näistä projekteista huomattavan moni on omaksunut nimenomaan Mapboxin vektoritiilen, mikä vaikuttaisi siltä, että se on ainakin osana OGC:n vektoritiilistandardia. (OGC Calls for Participation in Vector Tiles Pilot 2018.)

Tulevaisuudessa rasteritiilillä toteutettuja järjestelmiä voidaan korvata vektoritiilillä, jolloin saavutetaan ainakin dynaamisempia järjestelmiä sekä mahdollisesti järjestelmän suorituskyky paranee. Vektoritiilillä pystytään hyvin korvaamaan symbologisia karttaesityksiä, mutta esimerkiksi sateliitti-, ilma- ja ortokuvat ovat käytännön syistä parempi toteuttaa rasteripalveluna.

7.4 Vektoritiilet Matissa

Vantaan kaupunki on parhaillaan siirtynyt uuteen tietojärjestelmään: Mattiin. Siirtymässä on kyse nykyisen järjestelmien ja aineistojen siirtäminen ESRI:n tuotteiden ekosysteemiin. ESRI:n tuotteissa hyödynnetään myös vektoritiilejä, jotka ovat Mapboxin

spesifikaation mukaisia (Williams & Punt. 2017). Tällä hetkellä ESRI:n järjestelmässä aineisto voidaan muodostaa suhteellisen helposti vektoritiiliksi, jonka jälkeen aineistoa on helppo hyödyntää järjestelmän sisäisesti. Vektoritiilien avulla on kyetty ratkaisemaan suorituskykyongelmia joillakin aineistoilla. ESRI mainostaa olevansa avoinalusta, mikä voisi mahdollisesti tarkoittaa sitä, että Matti-järjestelmässä voitaisiin tulevaisuudessa sekä ylläpitää, että julkaista myös vektoritiili-karttarajapintoja. Toistaiseksi Matti-järjestelmä on vielä vaiheessa, jossa henkilöstö totuttelee suoriutumaan entisistä työtehtävistä uudella järjestelmällä. Osittain vanhojen järjestelmien sisältämiä aineistoja tuodaan vielä Mattiin. Karttarajapintojen julkaisemiseksi ulkoisiin järjestelmiin ei ole vielä toistaiseksi onnistuttu tyydyttävästi, vaan Geoserver on toistaiseksi pääjärjestelmä tähän tarkoitukseen.

Mikäli karttarajapintoja voitaisiin tulevaisuudessa julkaista Matista suoraan, tarjoaisi se rajapintojen ylläpidon kannalta korvaamattomia etuja. Esimerkiksi SLD-tyylitiedoston sijaan voitaisiin ylläpitää graafisella käyttöliittymällä ESRI:n LYRX-tiedostoa, joka ei vaadi käyttäjältään tietoa XML-kielestä. Toinen etu olisi se, että ArcGIS Pro:n graafisessa käyttöliittymässä tehdyt muutokset karttasymbologiaan ovat välittömästi nähtävissä, mikä nopeuttaa symbologian määrittelyprosessia huomattavasti.

7.5 Paikkatiedon yhteiskäyttö vektoritiileillä

Tietopalvelurajapinnan todellinen etu tulee siitä, että sitä voidaan hyödyntää yhteiskäytössä monen eri asiakasohjelman kesken ylläpitämällä sen tietosisältö vain yhdestä paikasta (JHS 180). Parhaimmillaan samasta asiakasohjelmasta voidaan hyödyntää useita tietolähteitä ja muodostaa esimerkiksi pääkaupunkiseudun kattava karttapalvelu. Vektoritiileillä voisi toteuttaa helposti palvelun, jossa kaupungilla ja kunnilla olisi yhteinen sopimuskäytäntö siitä, miten karttasymbolit tulisi nimetä, jotta asiakasohjelma voisi saumattomasti muodostaa yhtenäisen kartan useasta eri aineistosta. Suurin kysymys vektoritiilien yhteiskäytössä on nimenomaan niiden symbologian määrittelyssä.

Asiakasohjelmassa symbologian määrittelyyn käytetään vektoritiilien sisältämiä ominaisuustietoja. Jos vektoritiilet ovat binäärimuodossa, niistä itsestään on hyvin vaikea päästä käsiksi ominaisuustietoihin, joten ne tulee selvittää ennen vektoritiilen muodostusta tai vasta koodinpurun jälkeen. Kun karttapalvelua kehitetään organisaation sisällä,

on helppoa päästä alkuperäisen tietokannan kautta käsiksi ominaisuustietoihin, mutta organisaation ulkopuolisilla ei ole tätä mahdollisuutta. Toistaiseksi ei myöskään ole vielä työkalua, jolla voitaisiin antaa vektoritiililähde, jonka avulla voitaisiin dynaamisesti määritellä esimerkiksi graafisella käyttöliittymällä jokaisen tason symbolit. Ratkaistakseen tämä ongelma, kaupungin ulkoisia käyttäjiä voitaisiin opastaa vektoritiilien käytössä vähintäänkin dokumentoimalla vektoritiilien sisältämät ominaisuustiedot.

Lähteet

Agafonkin, Vladimir. 2015. Vector- vs raster. Verkkoaineisto. Github. <https://github.com/mapbox/vector-tile-spec/issues/53>. 21.12.2015. Luettu. 8.4.2019

Awesome vector tiles. Mapbox. 2019. Verkkoaineisto. Github. < https://github.com/mapbox/awesome-vector-tiles>. Luettu. 11.4.2019.

Bhatt, Rishil. 2017. Understand and Analyze Aggregate Report in Jmeter. Verkkoaineisto. Testing Journals: <http://www.testingjournals.com/understand-aggregate-report-jmeter/>. 9.3.2017. Luettu. 5.4.2019.

Blasby, David; Hocevar, Andreas. 2016. Vector Tiles with GeoServer and OpenLayers. FOSS4G paikkatietotapahtuma Saksassa. 24-26.8.2016. verkkoaineisto. FOSS4G. <http://ftp5.gwdg.de/pub/misc/openstreetmap/FOSS4G-2016/foss4g-2016-1154-vector_tiles_with_geoserver_and_openlayers-hd.webm>.

Cavazzi, Stefano. 2017. OGC Testbed-13: Vector Tiles Engineering Report. Verkkoaineisto. Open Geospatial Consortium. <http://docs.opengeospatial.org/per/17-041.html>. 5.12.2017. Luettu 7.2.2019.

Cozzi, Patric. 2017. The Next Generation of 3D Tiles. 2017 Verkkoaineisto. Cesium < https://cesium.com/blog/2017/07/12/the-next-generation-of-3d-tiles/>. Luettu. 11.4.2019.

Davies, Jason. 2014. Line Simplification. 22.7.2014. Verkkoaineisto. Saatavilla. <https://www.jasondavies.com/simplify/>. Luettu. 8.4.2019.

ESRI’s Open Vision. 2019. Verkkodokumentti. ESRI: <https://www.esri.com/en-us/arcgis/open-vision/overview>. Luettu. 13.5.2019.

Geometry.cpp. 2018. Tippecanoe. Mapbox. <https://github.com/mapbox/tippecanoe/blob/master/geometry.cpp>. Luettu. 8.4.2019

Geoserver 2.11.0 Release. 2017. Verkkoaineisto. 23.3.2017. Geoserver. <http://blog.geoserver.org/2017/03/23/geoserver-2-11-0-released/>. Luettu. 21.3.2017

Geoserver User Manual. 2018. GeoServer 2.16. Verkkoaineisto. Geoserver: < https://docs.geoserver.org/latest/en/user/> Luettu. 12.3.2019

Geowebcache. 2019. Verkkoaineisto. Geowebcache. <https://www.geowebcache.org/index.html>. Luettu. 21.3.2019

HTML Canvas 2D Context. 2015. Spesifikaatio. W3C. <https://www.w3.org/TR/2dcontext/>. Luettu. 20.3.2019

HTML5 W3C Recommendation. 2017. W3C. < https://www.w3.org/TR/html52/>. Luettu. 11.4.2019.

Ingensand, Jeans; Nappez, Marion; Moullet, Cédroc; Gasser, Loïc; Ertz, Oliver; Composto, Sarah. 2016. Implementation of tiled vector services: a case study. 9th International Conference on Geographic Information Science Montreal, Canada. 27-30.9.2016. CEUR. <http://ceur-ws.org/Vol-1777/paper4.pdf>. Luettu. 11.4.2019

JHS 180 Paikkatiedon sisältöpalvelut. 2011. Verkkoaineisto. Julkisen hallinnon tietohallinnon neuvottelukunta (JUHTA). JHS. <http://www.jhs-suositukset.fi/suomi/jhs180>. Luettu. 7.2.2019.

Leaflet.VectorGrid. 2019. Github. <https://github.com/Leaflet/Leaflet.VectorGrid>. Luettu. 2.4.2019

Leaflet.VectorGrid API reference. 2019. Verkkoaineisto. Github. < http://leaflet.github.io/Leaflet.VectorGrid/vectorgrid-api-docs.html>. Luettu. 2.4.2019

Mapbox-GL-JS. 2019. Verkkoaineisto. Mapbox. <https.//github.com/mapbox/mapbox-gl-js>. Luettu. 20.3.2019

Mapbox Style Specification. 2019. Verkkoaineisto. Mapbox. <https://docs.mapbox.com/mapbox-gl-js/style-spec/>. Luettu. 9.4.2019.

Nielsen, Jakob. 1993. Response Times: The 3 Importan Limits. Verkkoaineisto. NN Group. <https://www.nngroup.com/articles/response-times-3-important-limits/>. 1.1.1993. Luettu. 8.4.2019.

OGC Calls for Participation in Vector Tiles Pilot. 2018. Verkkoaineisto. Open Geospatial Consortium. < http://www.opengeospatial.org/pressroom/pressreleases/2837>. Luettu. 11.4.2019

OGC. 2006. 06-042: OpenGIS Web Map Service Implementation Specification.15.3.2006. Verkkoaineisto. Open Geospatial Consortium. < https://www.opengeospatial.org/standards/wms>.

OGC. 2007. 05-078r4: Styled Layer Descriptor profile of the Web Map Service Implementation Specification. 29.6.2007. Verkkoaineisto. Open Geospatial Consortium. <https://www.opengeospatial.org/standards/sld>.

OGC. 2010. 07-057r7: OpenGIS Web Map Tile Sevice Implementation Standard. 6.4.2010. Verkkoaineisto. Open Geospatial Consortium. <https://www.opengeospatial.org/standards/wmts>.

Geoserver project. 2019. Verkkoaineisto. The Open Source Geospatial Foundation. <http://osgeo.getinteractive.nl/projects/geoserver/>. Luettu. 21.3.2019

PipelineBuilder.java. 2018. Verkkoaineisto. Geoserver. <https://github.com/geoserver/geoserver/blob/master/src/extension/vectortiles/src/main/java/org/geoserver/wms/vector/PipelineBuilder.java>. Luettu. 8.4.2019.

PNG Spesification. 1999. PNG (Portable Network Graphics) Specification, Version 1.2. 14.7.1999. Verkkoaineisto. Saatavilla: <http://www.libpng.org/pub/png/spec/1.2/png-1.2.pdf>.

Protocol Buffers. 2018. Verkkoaineisto. Google. <https://developpers.google.com/protocol-buffers>. 23.8.2018. Luettu. 20.3.2019

Puchka, Ilya. 2016. Douglas-Peucker algorithm. 13.8.2016. Verkkoaineisto. Ilya Pucha. <http://ilya.puchka.me/douglas-peucker-algorithm/>. Luettu. 8.4.2019

Springmeyer, Dane 2015. The State of Vector tiles 9 -12.3.2015 Foss4g North America conference, San Francisco. Verkkoaineisto. FOSS4G. <http://2015.foss4g-na.org/session/state-vector-tiles.html>. Luettu. 22.10.2018

Taraldsvik, Mats. 2012. The future of web-based maps: can vector tiles and HTML5 solve the need for high-performance delivery of maps on the web? Masters Thesis. Norwegian University of Science and Technology. NTNU Open. <https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/23214432144>

vector-tile-spec. 2014. Verkkoaineisto. Mapbox. <https://github.com/mapbox/vector-tile-spec/>. 29.7.2014. Luettu 5.2.2019

Weistein, Eric. 2019. Bézier Curve. MathWorld. Verkkoaineisto. Mathworld. <http://mathworld.wolfram.com/BezierCurve.html>. Luettu. 8.4.2019.

What is Babel? 2019. Babel 7.4.0 Docs. Verkkoaineisto. Babel. <https//babeljs.io/docs/en>. Luettu. 29.3.2019.

Williams, Craig; Punt, Edie. 2017. YouTube: Esri Events. 25.8.2017. Youtube. < https://www.youtube.com/watch?v=dqKsEos1iSw>.

Openlayers-testisovelluksen lähdekoodi

Tässä on esitetty osa Openlayers 5.3.0 -testisovelluksen lähdekoodista. Lähdekoodi on kirjoitettu EchmaScript 6 -standardin mukaisesti, mikä vaatii koodin kääntämisen toimiakseen selaimessa. Lisäksi tässä esitetty karttarajapinta toimii vain Vantaan sisäverkossa.

```
import "ol/ol.css";
import { Map, View } from "ol";
import { Fill, Stroke, Style } from "ol/style";
import VectorTileLayer from "ol/layer/VectorTile";
import VectorTileSource from "ol/source/VectorTile";
import { MVT } from "ol/format";
import { createXYZ } from "ol/tilegrid";

var progress = 0;

const johtoverkkoLayer = (verkko, fillColor, strokeColor, width) => {
  const source = new VectorTileSource({
    projection: "EPSG:3857",
    tileGrid: createXYZ({
      extent: [-20037508.34, -20037508.34, 20037508.34, 20037508.34], // projection extent
      maxZoom: 20
    }),
    format: new MVT(),
    url: `http://srvilmakuva:8083/geoserver/gwc/service/tms/1.0.0/johto-
kartta:${verkko}/EPSG:3857/{z}/{x}/{-y}.pbf`
  });

  source.on("tileloadstart", () => {
    progress++;
  });

  source.on("tileloaderror", () => {
    progress--;
  });

  source.on("tileloadend", () => {
    progress--;
  });
```

```
  return new VectorTileLayer({
    style: new Style({
      fill: new Fill({
        color: fillColor
      }),
      stroke: new Stroke({
        color: strokeColor,
        width: width || 1
      })
    }),
    source
  });
};

const view = new View({
  center: [2781640, 8467632],
  zoom: 11,
  projection: "EPSG:3857"
});

const map = new Map({
  target: "map",
  layers: [
    johtoverkkoLayer("sahkoverkko", "#ffdddd", "#ff1111"),
    johtoverkkoLayer("televerkko", "#9fff0f", "#168714"),
    johtoverkkoLayer("kaukolampoverkko", "#ffffffff", "#c918c0"),
    johtoverkkoLayer("kaasuverkko", "#ffffffff", "#ffbc42", 2),
    johtoverkkoLayer("hsy_vesijohtoverkko", "#ffffffff", "#70c5f9"),
    johtoverkkoLayer("hsy_jatevesiverkko", "#ffffffff", "#964800"),
    johtoverkkoLayer("hsy_hulevesiverkko", "#ffffffff", "#ad7000")
  ],
  view
});

const mittausTulokset = [];
map.on("postrender", event => {
  // frameState.time = "The time when rendering of the frame was requested." -> https://openlay-
ers.org/en/latest/apidoc/module-ol_PluggableMap.html#~FrameState
  const renderTime = Date.now() - event.frameState.time;
  console.log(`Rendered in: ${renderTime}ms`);
  mittausTulokset.push(renderTime);
  clearTimeout(viewReposition);
  viewChangeTimer();
});
```

```
let viewReposition;
let nextView = 0;
let kierrosAika = new Date();
function viewChangeTimer() {
  if (progress == 0) {
    viewReposition = window.setTimeout(() => {
      if (nextView < testViews.length) {
        view.animate(testViews[nextView]);
        nextView++;
      } else {
        kierrosAika = Date.now() - kierrosAika;
        let millisekunnit = kierrosAika % 60000;
        const aikaLopussa =
          Math.floor(kierrosAika / 60000) + ":" + millisekunnit / 1000;

        console.log("Test completed in: " + aikaLopussa);

        let avg = 0;
        const n = mittausTulokset.length;
        mittausTulokset.forEach(t => {
          avg += t;
        });
        avg = avg / n;

        let std = 0;
        mittausTulokset.forEach(e => {
          std += (e - avg) * (e - avg);
        });
        std = Math.sqrt(std / (n - 1));

        const object = {
          mittausTulokset,
          maara: n,
          keskiarvo: avg,
          keskihajonta: std
        };

        console.log(object);
      }
    }, 1000);
  }
}
```

```
const z = 13;
const testViews = [
  { zoom: z + 0, center: [2786617.125, 8465397] },
  { zoom: z + 1, center: [2786617.125, 8465397] },
  { zoom: z + 2, center: [2786617.125, 8465397] },
  { zoom: z + 3, center: [2786617.125, 8465397] },
  { zoom: z + 4, center: [2786617.125, 8465397] },
  { zoom: z + 4, center: [2786617.125, 8465500] },
  { zoom: z + 4, center: [2786617.125, 8465600] },
  { zoom: z + 4, center: [2786617.125, 8465700] },
  { zoom: z + 5, center: [2786617.125, 8465397] },
  { zoom: z + 5, center: [2786800.125, 8465397] },
  { zoom: z + 5, center: [2787000.125, 8465397] },
  { zoom: z + 5, center: [2787200.125, 8465397] },
  { zoom: z + 5, center: [2787400.125, 8465397] },
  { zoom: z + 5, center: [2787400.125, 8465000] },
  { zoom: z + 5, center: [2787400.125, 8464800] },
  { zoom: z + 5, center: [2787400.125, 8464600] },
  { zoom: z + 5, center: [2787400.125, 8464400] },
  { zoom: z + 5, center: [2787000.125, 8464200] },
  { zoom: z + 5, center: [2786500.125, 8464200] },
  { zoom: z + 5, center: [2786000.125, 8464200] },
  { zoom: z + 6, center: [2786617.125, 8465397] },
  { zoom: z + 7, center: [2786617.125, 8465397] },
  { zoom: z + 3, center: [2781640, 8467632] }
];
```

Asiakasohjelmien suorituskykymittaus

Openlayers-VCT

	n	avg (ms)	std (ms)	lap (sec)
	2771	6,99	18,86	74,218
	1959	9,02	23,25	78,209
	2121	8,44	22,18	82,586
	1955	9,71	28,07	79,684
	2175	8,36	22,72	80,267
	2228	8,26	22,34	83,034
	1938	9,21	24,37	77,798
	1908	9,52	25,36	80,417
	1997	9,07	24,34	80,965
	1929	9,48	27,15	80,186
avg	2098	8,81	23,86	79,7364
std	248,3	0,773876	2,511833	2,408248

Openlayers-WMTS

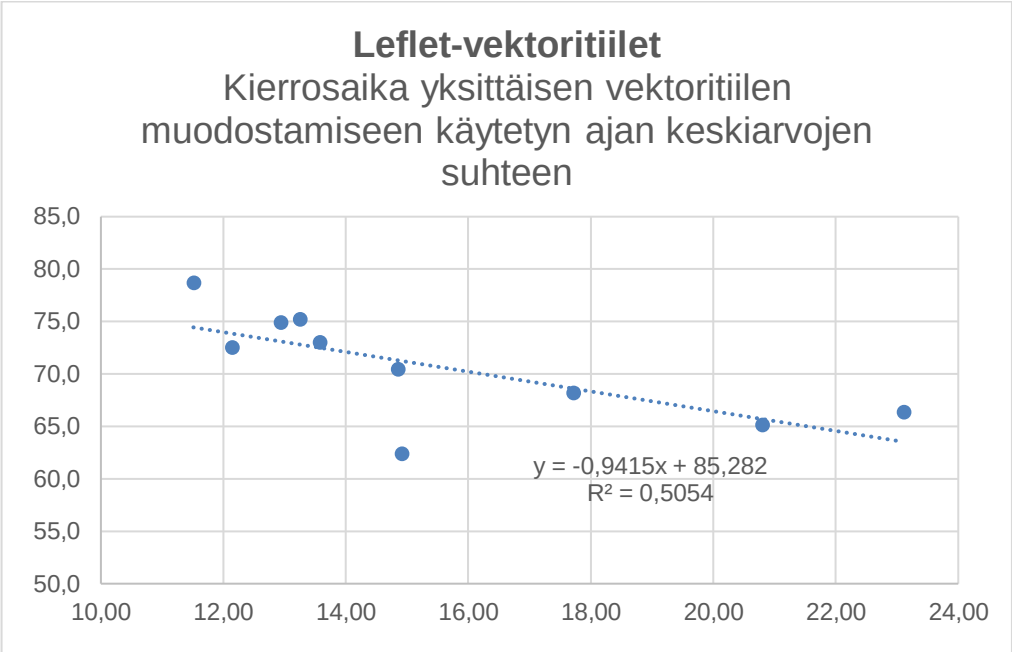
n	avg (ms)	std (ms)	lap (sec)
2603	4,32	4,12	73,806
2925	4,28	3,81	77,417
2738	4,4	3,99	73,867
2640	4,61	4,47	72,999
3079	4,34	3,66	79,246
2903	4,22	4,22	91,855
2736	4,37	4,02	75,9
2870	4,31	4,07	79,31
2845	4,63	4,38	77,793
2845	4,21	3,99	75,654
2818	4,37	4,07	77,7847
134,9	0,137583	0,230653	5,145327

Leaflet-VCT

	n	avg (ms)	std (ms)	lap (sec)
	6974	20,80	33,10	65,167
	6964	12,94	24,83	74,907
	6962	12,14	23,94	72,547
	6970	17,71	36,69	68,22
	6968	13,57	25,52	73,017
	6842	14,85	25,07	70,454
	6969	13,25	27,23	75,23
	6961	14,91	31,22	62,433
	6968	11,51	21,24	78,733
	6969	23,11	45,43	66,377
avg	6955	15,48	29,43	70,7085
std	37,75	3,663374	6,94906	4,851645

Leaflet-TMS

n	avg (ms)	std (ms)	lap (sec)
6811	-	-	58,511
6811	-	-	56,547
6811	-	-	56,319
6811	-	-	52,734
6811	-	-	49,304
6811	-	-	56,979
6811	-	-	51,197
6811	-	-	48,613
6811	-	-	48,503
6811	-	-	48,38
6811	-	-	52,7087
0	-	-	3,830759



YHTEENVETO TULOSTUS

Regressiotunnusluvut	
Kerroin R	0,71093
Korrelaatiokerroin	**0,505421**
Tarkistettu korrelaatio-kerroin	0,443599
Keskivirhe	2,880408
Havainnot	10

ANOVA

	va	*NS*	*KN*	*F*	*F:n tarkkuus*
Regressio	1	67,82908	67,82908	8,175378	0,021175416
Jäännös	8	66,37401	8,296751		
Yhteensä	9	134,2031			

	Kertoi-met	*Keski-virhe*	*t Tun-nusluvut*	*P-arvo*	*Alin 95%*	*Ylin 95%*	*Alin 95,0%*	*Ylin 95,0%*
Leikkauspiste	53,43588	13,30628	4,015841	**0,003863**	22,75155464	84,12021	22,75155	84,12021
Muuttuja X 1	-0,53681	0,187744	-2,85926	0,021175	-0,969745266	-0,10387	-0,96975	-0,10387

JMeter-mittaustulokset

J-Meter Thred Group setup				Summary Report				
Format	Threads	Loops	Elapse Time	Samples	Avg	Std	Mega Bytes	Avg Bytes
PNG	15	3	09:18	148410	130	3745		
PNG	15	3	09:01	148410	108	3120		
PNG	15	3	08:54	148410	105	3016	26,98	16364
PNG	15	3	09:02	148410	105	3042		
PNG	15	10	34:01	494700	121	3523		
PBF	15	3	11:13	161415	119	3601		
PBF	15	3	13:44	161415	148	4416		
PBF	15	3	12:18	161415	131	3937	32,97	18383
PBF	15	3	12:22	161415	131	3950		
PBF	15	10	41:38	699465	133	4018		

J-Meter Thred Group setup				Aggregate Report				
Format	Threads	Loops	Elapse Time	Avg	Median	90%-Line	95%-Line	99%-Line
PNG	15	3	09:18	130	50	109	143	271
PNG	15	3	09:01	108	43	90	118	197
PNG	15	3	08:54	105	43	87	112	178
PNG	15	3	09:02	105	43	88	113	181
PNG	15	10	34:01	121	44	97	130	248
PBF	15	3	11:13	60	36	118	158	293
PBF	15	3	13:44	73	43	153	214	399
PBF	15	3	12:18	65	40	137	189	350
PBF	15	3	12:22	66	40	140	193	354
PBF	15	10	41:38	67	41	144	191	334