KARELIA-AMMATTIKORKEAKOULU
Tietojenkäsittelyn koulutusohjelma

Jarno E. Tarvainen

KOLMIULOTTEISEN MALLINNUKSEN NÄKYVYYS JA TEHOKKUUS VERKKOSIVUKEHITYKSESSÄ

Opinnäytetyö
Huhtikuu 2021

Karelia AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Huhtikuu 2021
Tietojenkäsittelyn koulutusohjelma

Tikkarinne 9
80200 JOENSUU
+358 13 260 600 (vaihde)

Tekijä(t)
Jarno E. Tarvainen

Nimeke
Kolmiulotteisen mallinnuksen näkyvyys ja tehokkuus verkkosivukehityksessä

Tiivistelmä

Opinnäytetyössä keskityin käsittelemään 3D-mallinnuksen käyttöönottoa verkkosivukehityksessä. Lisäksi perehdyttiin siihen, miten kolmiulotteinen grafiikka vaikutti sivuston näyttävyyteen. Tavoitteena oli tuoda esille, miten kolmiulotteinen grafiikka vaikutti verkkosivun käyttökokemukseen ja verkkosivun tehokkaaseen toimintaan.

Opinnäytetyön teoriaosassa tarkasteltiin sitä, millä tavalla 3D-mallien käyttäminen voi vaikuttaa verkkosivun kävijämäärään. Johtopäätöksenä tuli selville, että kävijämäärään saattaa vaikuttaa myös sivuston tehokas toteutustapa, kun sisältö latautuu nopeammin käyttäjille. Verkkosivuston tekemiseen on käytetty JavaScript Three.js-koodikirjastoa. Lisäksi on ollut tarpeen käyttää Three.js-koodikirjaston avulla tehtyjä apukirjastoja. GLTFLoader.js-koodikirjastoa on käytetty valmiin 3D-mallin esittämiseen. OrbitConrols.js- ja TrackballControls.js-koodikirjastoja on käytetty 3D-mallien kääntelyn ja zoomauksen mahdollistamiseen hiirellä. Tutkimusosiossa laadittiin verkkosivusto, jossa otettiin käyttöön 3D-malleja Three.js-JavaScript-koodikirjaston avulla. Työssä verrattiin myös verkkosivustolle ohjelmoitujen ja vapaaseen käyttöön lisensoidun valmiin 3D-mallin siirto- ja latautumisnopeuksia sekä perehdyttiin niiden mahdolliseen lisäarvoon sivustolla.

Tietotekniikan kehittymisen myötä tulevaisuudessa voidaan nähdä 3D-grafiikan lisääntyminen verkkosivustoilla. Tulokset osoittavat kolmiulotteisen mallinnuksen etuja kaksiulotteiseen nähden. Kolmiulotteisella mallinnuksella saatiin aikaan näyttävämpiä verkkosivuja ja tehokkaammalla toteutuksella voitiin vaikuttaa verkkosivujen parempaan toimintaan.

Kieli
suomi

Sivuja 52
Liitteet
Liitesivumäärä

Asiasanat
valkokangaselementti, minifikaatio, rautalankamalli, polygonimalli, 3D-malli, 3D-mallinnus

Karelia
UNIVERSITY OF APPLIED SCIENCES

THESIS
April 2021
Degree Programme in
Business Information Technology

Tikkarinne 9
80200 JOENSUU
FINLAND
+ 358 13 260 600 (switchboard)

Author (s)
Jarno E. Tarvainen

Title
Visibility and efficiency of 3D modeling in website development

Abstract

The thesis was focused on dealing with the implementation of 3D modeling in website development. In addition, the focus was on how 3D graphics can affect the visibility of a website. The aim was to highlight how 3D graphics could affect the experience of using a website and the efficient operation of a website.

The theoretical part of the thesis examined how the use of 3D models could affect the number of visitors on a web page. The conclusion was that the number of visitors might also be affected by the efficient development of the website, as the content loads faster for users. The JavaScript code library Three.js was used to create a website. In addition, it was necessary to use auxiliary code libraries made with the help of the Three.js code library. The code library GLTFLoader.js was used to present the complete 3D model. The OrbitConrols.js and TrackballControls.js code libraries were used to enable 3D models to rotate and zoom with the mouse. Web pages were created for the research section in which 3D models were introduced with the help of the Three.js JavaScript code library. The network transfer and the Onload Time speeds of the programmed 3D models, and free software licensed complete 3D model on the website were also compared, and their possible added value on the website was examined.

In the future, an increase of 3D graphics on websites could be seen as information technology develops to be more advanced. The results show the advantages of 3D modeling over 2D. With the aid of 3D modeling, more impressive web pages were created, and efficient development resulted in better performance of web pages.

Language	Pages 52
Finnish	Appendices
	Pages of Appendices

Keywords
canvas, minification, wire-frame model, mesh, 3D model, 3D modeling

Sisältö

1 Johdanto

Opinnäytetyön tarkoitus on tuoda esille tietoa siitä, miten kolmiulotteiset mallit tällä hetkellä toimivat verkkosivujen kanssa ja onko niiden lisääminen järkevää nyt tai lähitulevaisuudessa. Kiinnostukseni kolmiulotteisten pelien kehittämiseen vaikutti aiheen valintaan. Tarkastelen opinnäytetyössä verkkosivujen kolmiulotteista grafiikkaa yleisellä tasolla. Erikoistapauksena nostan esiin selaimessa toimivat verkkosovellukset ja pelit. Lähestyn aihealuetta niissä käytettävien 3D-mallien ja -grafiikan ohjelmoinnin näkökulmasta. Kolmiulotteisen grafiikan lisääminen verkkosivuille voi tehdä niistä näyttävämpiä. Grafiikka voi antaa verkkosivuille uuden ilmeen ja tehdä niistä mielenkiintoisempia selata. Audiovisuaalisin keinon voidaan esittää monenlaista informaatiota tekstien sijaan.

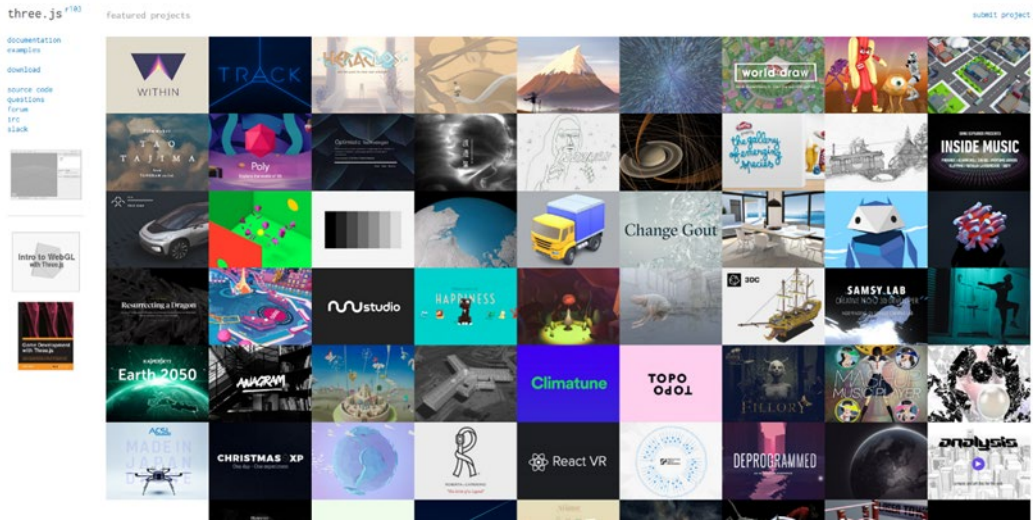
Opinnäytetyössä esitellään, miten 3D-grafiikkaa tehdään Three.js-JavaScript-kirjaston avulla. 3D-grafiikan tuomiseen verkkosivuille on olemassa useita tekniikoita. Lukuisien esimerkkien innoittamana itse valitsin tekniikaksi Three.js-kirjaston. Yleisen ja teknisen esittelyn keinoin sekä kolmiulotteisien mallien avulla tuon esille verkkosivuston muuttamista kolmiulotteiseen suuntaan ja sen vaikutusta verkkosivun näyttävyyteen. Näyttävyydellä tarkoitan parempaa ulkoasua, joka houkuttelee käyttäjiä. Lisäksi esittelen opinnäytetyössä Three.js-kirjaston käytön tehokkuusnäkökulmaa. Tehokkuusnäkökulmasta tarkastelen verkkosivujen nopeutta ja esittelen käytettävyyttä. Tätä tarkoitusta varten olen tehnyt kolmiulotteisia malleja sisältävät verkkosivut ja olen tarkastellut niiden lataus- ja siirtonopeuksia verkossa. Tehokkuuden perusteella pyrin tuomaan esille hyvin toteutetun verkkosivun vaikutusta näkyvyyteen hakukonetuloksissa. Tuon esille, miksi juuri tehokkuus on tärkeää kolmiulotteisten mallien lisäyksessä sivustolle.

Opinnäytetyössä testataan verkkosivuille ohjelmoimiani kolmiulotteisia kappaleita ja yhtä 3D-mallinnusohjelmalla tehtyä. Tarkastelen kolmiulotteista mallintamista ja sen vaiheita sekä esittelen tekniikoita verkkosivun optimointiin. Mittaustulokset tuovat esille tehokkuusnäkökulmaa verkon yli siirtyvien sivujen latautumis- ja siirtonopeuksien avulla. Esittelen tekniikan, jolla verkkosivukehittäjä voi tarkemmin valita, mitä osia valmiista ohjelmakirjastosta haluaa omaan

toteutukseensa ottaa. Lopuksi pohdin sitä, milloin kolmiulotteisuutta voisi lisätä verkkosivuille myös pelkästään omaa JavaScript-koodia ohjelmoimalla.

2 3D-grafiikka ja verkkosivuympäristö

Three.js (Three.js 2020a) on useilla eri selaimilla toimiva JavaScript-kirjasto ja ohjelmistorajapinta. Kirjastoa käytetään 3D-grafiikan, animaatioiden ja pelien luontiin verkkoselaimella avautuvaan ympäristöön. Three.js-kirjaston toiminta pohjautuu WebGL-ohjelmistorajapintaan, joka mahdollistaa näytönohjaimella tehostetun fysiikan ja kuvankäsittelyn (Khronos Group 2020). Fysiikka tarkoittaa tässä peleihin tehtyjä ominaisuuksia, jotka kuvastavat reaalimaailman fysikaalisia ilmiöitä (Wikipedia 2019a). WebGL-kirjastolla pystyy siis ohjelmoimaan kolmiulotteista grafiikkaa, ja Three.js-kirjastolla ohjelmointia on helpotettu, koska se sisältää valmiiksi tehtyjä työkaluja, jotka ohjelmoija joutuisi muuten ohjelmoimaan aina erikseen. Kuva 1 havainnollistaa Three.js-verkkosivulla olevaa valmista 3D-grafiikkaa, jota on toteutettu kirjaston avulla.



Kuva 1. Three.js-verkkosivun aloitussivulla yleisnäkymä kirjastolla toteutetuista esimerkeistä (Three.js 2019a).

Kolmiulotteisissa peleissä tarvitaan usein reaalimaailman fysiikan lakien simulointia ja grafiikka toimii sulavammin, kun voidaan hyödyntää näytönohjaimen

suorituskykyä (Khronos Group 2020). Kuten 3D-grafiikassa yleensä, tämä rajoittaa esittämistä niissä laitteissa, joissa on heikko näytönohjain.

2.1 3D-grafiikan ja hankinta

Kolmiulotteisen grafiikan esittäminen tarvitsee tuekseen kuvaa, joka jollain tavalla näkyy loppukäyttäjälle kaksiulotteisen kuvan sijaan myös syvyyssuunnassa. Vaikka usein puhutaan 3D-mallien lisäämisestä johonkin ohjelmaan, niin tietokoneella näytettävä kolmiulotteinen kuva muodostuu kuitenkin varsinaisesta 3D-mallista ja siihen lisättävästä grafiikasta. 3D-mallin pohjalla on matemaattinen esitys kolmiulotteisesta kappaleesta. Tällöin sen kolmiulotteisen avaruuden kuvaamisen yhteydessä voidaan puhua rautalankamallista (Wikipedia 2019b). Kolmiulotteiset mallit saadaan näkymään ruudulla grafiikoineen 3D-renderöinnin avulla. Termi renderöinti tarkoittaa yleisesti taiteilijan näkökulmaa kuvatusta asiasta. Renderöinti voidaan jakaa 3D- ja 2D-renderöinteihin sekä reaali- ja epärealistisen maailman kuvaamiseen. 3D-renderöinti pitää sisällään tapahtumapaikan tai näyttämön (scene). Tapahtumapaikka koostuu geometriasta, katselukohdasta, pintarakenteesta, valaistuksesta ja varjostuksista, jotka kuvaavat virtuaalista ympäristöä. Todellisuutta kuvaavan renderöinnin avulla kuvasta saadaan todellisen maailman näköaistimuksia vastaavaa sisältöä (Wikipedia 2019c). Kolmiulotteisessa tietokonegrafiikassa muodostetaan polygonimalleja, jotka ovat useista polygoneista koostuvia kappaleita (Wikipedia 2019d). Näitä malleja hyödyntämällä voidaan tehdä myös 3D-tulostusta, jolloin malli renderöidään fyysisen esineen tulostuksen yhteydessä (Wikipedia 2019e).

Valmiita 3D-malleja voidaan etsiä suoraan verkosta. Eri verkkosivustot tarjoavat niitä maksullisena ja ilmaiseksi. Sivustoilta löytyy malleja niin 3D-grafiikan esittämiseen kuin 3D-tulostamiseenkin. Jotkut verkkosivustot mahdollistavat sen, että käyttäjät voivat ladata sivustolle omia mallejaan ja kaupata niitä.

3D-mallinnusohjelmilla tehtyihin ja grafiikan esittämiseen tarkoitettuihin ohjelmistoihin sopivia malleja löytyy eri tiedostoformaateissa. Tällöin lataaja joutuu miet-

timään käyttämäänsä ohjelmaan sopivan 3D-mallin tiedostoformaatin. Esimerkkejä mahdollisista paikosta etsiä grafiikkaa peli- ja animaatiokäyttöön ovat Unity Asset Store (Unity 2020), 3D Warehouse (Trimble 2020a), TurboSquid (TurboSquid 2020), AutoDesk Online Gallery (AutoDesk 2020a) ja Smithsonian X3D (Smithsonian 2020).

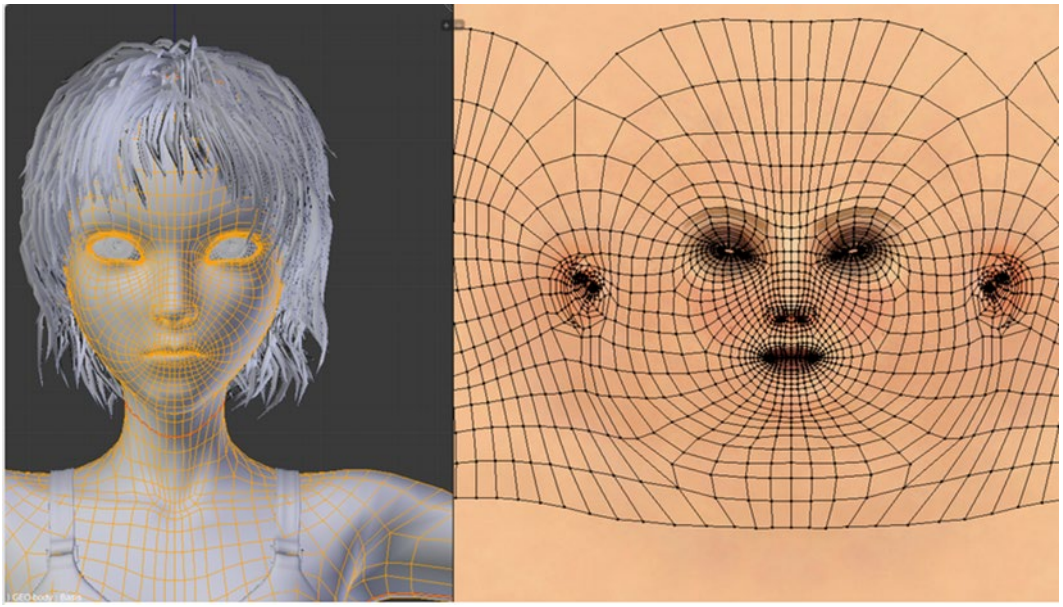
Verkkosivun tekemistä varten voi ostaa 3D-malleja tai grafiikkaa. Tällöin on hyvä tarkastaa sallivatko lisenssiehdot ostetun sisällön käyttämisen omilla verkkosivuilla julkaistavaksi. Jos eivät, voi selvittää, voiko kaupalliseen tai julkaisemiskäyttöön hankkia lisenssin erikseen ja kuinka paljon siitä tulee kustannuksia. Yleisesti lisenssit jakautuvat teknologia- ja sisältölisensseihin. 3D-mallien suhteen voidaan keskittyä sisältölisensseihin. Tarkemmin on hyvä etsiä grafiikkaa, jonka lisensointi antaa mahdollisimman paljon vapauksia käyttää sisältöä uudelleen. Pääpiirteittäin lisensseihin voidaan tehdä jako copyright- ja copyleft-lisensseihin.

Copyleft-lisenssit mahdollistavat sisällön uudelleen muokkaamisen, levittämisen ja julkaisemisen. Ehtona on kuitenkin, että alkuperäistä lisensointia tulee käyttää jatkossa ja alun perin määritellyillä ehdoilla (The XFree86 Project 2020). Muokkauksia omaan käyttöön voi tietysti tehdä ilman ehtoa. Alun perin ohjelmoija Don Hopkins sai idean vapaista lisenssiehdoista ja Richard Stallman kehitti idean pohjalta käsitteen copyleft (Stallman 2015). Englannin kielen sana tulee päinvastaiseen muotoon käännetystä merkityksestä sanalle copyright. Tekijänoikeuksista siis luovutaan ja oikeudet ovat käyttäjällä. Käyttäjänoikeus ei kuitenkaan synny automaattisesti, vaan jonkun, kuten alkuperäisen tekijän, on määriteltävä sisällölle copyleft-lisensointi (GNU Operating System 2020).

Vapaasti käytettävään sisältöön voi olla myös niin sanottuja ei-copyleft-lisenssejä (non-copyleft), joissa ei ole vaatimuksia muokatun sisällön julkaisulle. Lisensointi mahdollistaa kuitenkin eri rajoitusten määrittämisen jatkossa, kuten kopiointikiellon asettamisen. Tällainen lisensointi voi olla hyödyllistä kaupallisiin julkaisuihin. Kolmiulotteisen grafiikan teko saatavilla olevilla ohjelmilla voi olla aikaa vievää työtä, joten on eduksi, jos osa työstä on tehty jo etukäteen. (Wikipedia 2019f.)

2.2 3D-mallien ja grafiikan toteuttaminen

Valmiiden mallien sijasta tietokonegrafiikkaa voi olla hyvä tehdä myös itse, jolloin välttyy tekijänoikeuksiin liittyvältä selvittelyltä. Mallinnusta varten voi joutua kuvaamaan mallinnettavan esineen usealta puolelta, mikäli käytössä on vain tavallinen kamera. Kuvat voidaan sitten syöttää johonkin kolmiulotteisuuden mallintamiseen kehitettyyn ohjelmaan. Ohjelmassa kuviin luodaan varsinainen 3D-malli eli rautalankamalli. Mallia voidaan ajatella myös eräänlaisen kehikkona kuvan sisällä (Wikipedia 2019b). Kuvassa 2 rautalankamalli on tehty ilmaiseksi saatavana olevalla Blender 3D-mallinnusohjelmalla (Blender 2019).

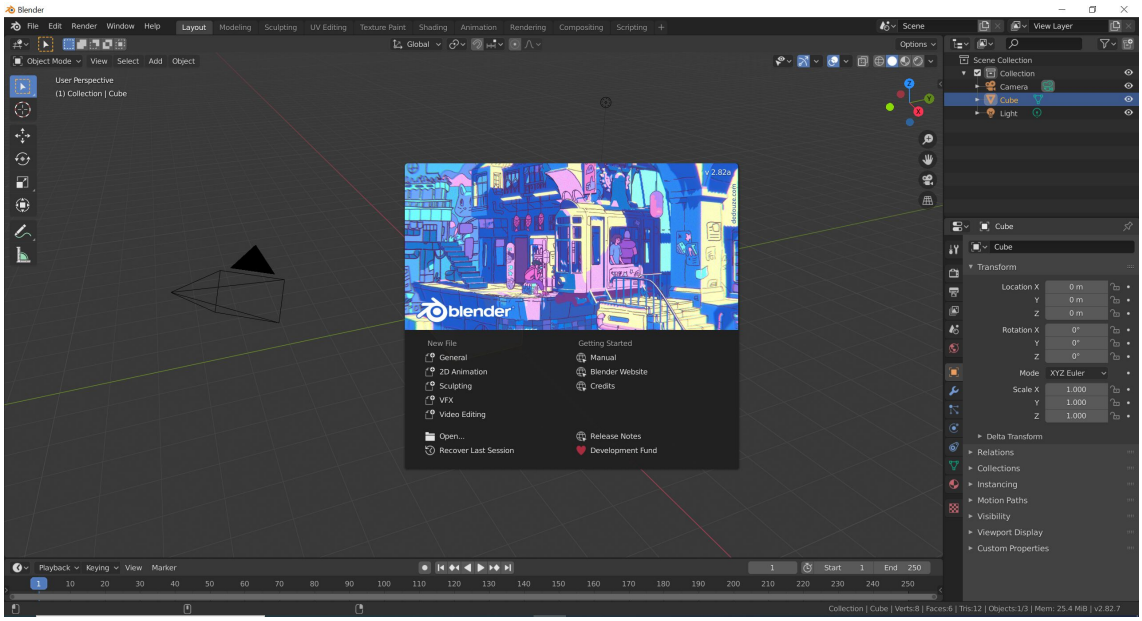


Kuva 2. Kasvojen rautalankamalli (wire-frame model) Blender-3D ohjelmassa (Blender 2019).

Muita vaihtoehtoisia ohjelmia ovat muun muassa InkScape (InkScape 2020), Cinema 4D (Maxon 2020), Wings 3D (Wings 3D 2020) ja Sketchup (Trimble 2020a). Adobe-tuoteperheeseen kuuluu myös Adobe Comp CC (Adobe 2020), joka toimii Androidilla ja iOS-käyttöjärjestelmällä.

3D-mallinnus Blender-ohjelmalla alkaa yksinkertaisimmillaan siten, että eri kuvakulmista otetut kuvat ladataan ohjelmaan ja niille luodaan alustaksi rautalankamalli (Wikipedia 2019b). Malli koostuu laskennallisista pisteistä XYZ-

koordinaatistossa, ja sitä voidaan venytellä ohjelmassa oikeaan muotoonsa. Venyttelyä voidaan tehdä eri kuvakulmista liikuttelemalla mallia. Kuvassa 3 on Blender-ohjelman aloitusnäkymä.



Kuva 3. Vapaan 3D-mallinnukseen kehitetyn Blender-ohjelman aloitusruutu.

Maksullisia ohjelmia on niin harrastelijoille kuin ammattimaiseen tai teolliseen käyttöön tarkoitettuja. Joukosta voidaan mainita ainakin AutoCAD (AutoDesk 2020b) ja 3DsMax (AutoDesk 2020c). Helpointa on, jos ohjelma tekee mallinnuksen mahdollisimman automaattisesti ja valmiiksi, kun käyttää siihen soveltuvaa laitetta.

Kolmiulotteiseen kuvaukseen on olemassa myös siihen erikoistuneita kameroita, jotka kuvaavat useammalta puolelta yhdellä kertaa. Lisäksi on olemassa erikoistuneita välineitä, joilla esineen voi skannata joka suunnasta ja laite muodostaa informaation perusteella tarkan 3D-mallin. Tarkoitukseen kehitetyillä skannereilla voidaan nopeasti mallintaa pieniä hyönteisiä, koko vartalo tai vaikkapa suuria todellisen maailman esineitä kuten jumbojet-lentokoneita. Laitteissa hyödynnetään laser-valotekniikkaa kappaleiden syvyyssuuntaiseen kuvaukseen. (Artec 3D 2019.)

Kuvassa 4 näkyy 3D-skanneri, joka on erityisen kätevä tietokoneavusteiseen suunnitteluun eri tieteen aloille, joilla on tarvetta mallintaa todellisen maailman kappaleita. Laitetta pidetään kiinni toisessa kädessä ja pohjasta tulevat laser-säteet mittaavat kuvattavan kappaleen pinnanmuodot tarkasti. Mittausten avulla laite muodostaa digitaalisen kolmiulotteisen kehikon tietokoneelle. (Artec 3D 2019.)

Kuva 4. Artec Space Spider 3D-skanneri (Artec 3D 2019).

Hyvin suurten kappaleiden skannaus voi käydä kuitenkin työlääksi Kuvan 4 skannerilla. Kuva 5 esittää suurten kappaleiden skannaukseen kehitettyä 3D-skanneria. Tämän mallista skanneria on käytetty muun muassa suurten lentokoneiden mallintamiseen kolmiulotteiseksi. Laseria käytetään tässäkin skannaukseen, ja tekniikka mahdollistaa sen, että skanneri voidaan asettaa kauemmaksi mallinnettavasta kohteesta.

Kuva 5. Isojen kappaleiden kuvaukseen tarkoitettu Artec Ray 3D-skanneri (Artec 3D 2019).

Usein tavallisessakin kuvauksessa ihmiset ovat halunneet itsensä mahdollisimman hyvin näkyviin kuvissa. 3D-mallinnuksen avuksi on myös kehitetty kokovartalon kuvaava 3D-skanneri. Laitteeseen voi vaikka kävellä kuvattavaksi. Kuva 6 on esimerkkinä laitteesta. Kolmiulotteisten tietokonepelien ja virtuaalitodellisuuksien luomiseen skanneri on varmasti kätevä ja nopea tapa muodostaa kolmiulotteisia kokovartalomalleja.

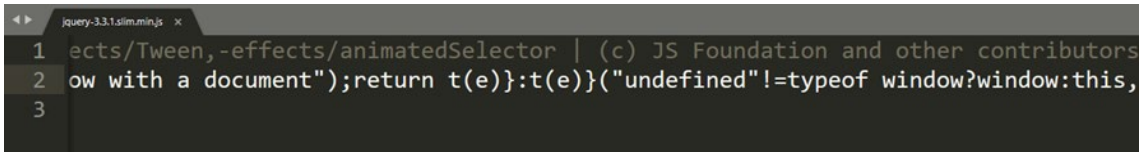
Kuva 6. Kokovartalon kuvaukseen suunniteltu Artec Shapify Booth 3D-skanneri (Artec 3D 2019).

2.3 Näkyvyys, näyttävyys ja tehokkuus

Kolmiulotteista grafiikkaa voi lisätä sivustolle lisäämään näyttävyyttä tai sivuston voi tehdä kokonaan sisältämään 3D-grafiikkaa. Verkkosivu voi olla joko mielikuvituksellinen animaatio tai peli, mutta tekniikalla voidaan visualisoida myös reaalimaailman ilmiöitä. On hyvin mahdollista, että tulevaisuudessa tietokoneohjelmat ja verkkosivukehitys kehittyvät yhä enemmän kolmiulotteisempaan suuntaan (Naker 2021). Grafiikka voi olla hyvin näyttävää sekä miellyttävää katsella ja kuunnella. Ihmiset eivät aina opi asioita parhaiten vain lukemalla, ja audiovisuaalisuus tuo lisäarvoa myös materiaaleille, jotka on suunniteltu oppimistarkoituksiin. (Shabiralyani, Hasan, Hamad & Iqbal 2015, 228–233.) Itch.io-verkkosivustolla on muutamia esimerkkejä peleistä, joilla voi opetella asioita, kuten biljardin pelaamista (Itch corp 2020).

Tällä hetkellä käyttäjien keskuudessa suosituimmat ja parhaiten toteutetut sivustot löytyvät usein ensimmäisinä myös hakukoneista, mikä tuo niille näkyvyyttä (Quality Nonsense Ltd 2020). Näkyvyys voi kuitenkin vielä laskea perinteisiin sivustoihin nähden, koska kolmiulotteisen grafiikan lataaminen verkon kautta on hitaampaa ja sen esittäminen vaatii enemmän suorituskykyä laitteelta. Tästä syystä on tärkeää kiinnittää erityistä huomiota kaikkiin toimiin ja tekniikoihin, joilla

verkkosivun toimintaa voidaan nopeuttaa. Aiemmin mainitsin JavaScript-kirjastojen verkkosivun toiminnan kannalta tarpeellisten ominaisuuksien poiminnan yhteenvetotiedostoksi (Bundle), jolloin tiedostosta saadaan yksi itse muodostettu yhtenäinen kirjasto. Yhteenvetotiedostolla (Bundle) tarkoitan tiedostoa, johon on koottu ohjelmassa tarvittava koodi ja tiedostosta on poistettu kaikki ylimääräinen, joka vie tilaa. Tällaisen yhteenvetotiedoston kaiken ylimääräisen poistamisprosessiin on olemassa englanninkielinen termi, jolle en löytänyt vielä virallista suomenkielistä vastinetta (minification). Termi viittaa tiedoston mahdollisimman pieneen kokoon (WebPlatform.org 2020). Suomen kielellä voitaisiin puhua minifikaatiosta. Kuvassa 7 nähdään, kuinka JavaScript-koodi kommentteineen mahtuu mahdollisimman pieneen tilaan, vain parille riville minimoidussa tiedostossa. Kyseessä on jQuery-kirjastotiedoston slim-versio, joka sisältää vain yleisimmin tarvittavia ominaisuuksia.



Kuva 7. JavaScriptillä tehty jQuery-kirjaston minimoitu slim-versio.

Verkkosivujen latautumisen nopeuttamiseen on kuitenkin olemassa enemmänkin keinoja. Näistä voidaan mainita muutamia esimerkkejä, kuten palvelimen nopeus ja sijainti sekä verkkosivujen latautuminen eri palvelimilta selaajan sijainnin perusteella. Optimoinnin parantamiseksi koodin tulee olla myös uusimpien standardien mukaan tehty, ja skriptikielen suositeltuja uusia ominaisuuksia on usein hyvä käyttää. HTML5:n uusia ominaisuuksia ja standardeja on hyvä hyödyntää (Quality Nonsense Ltd 2020). Tiedostot latautuvat verkon yli paloissa ja skriptikielen kääntäminen tapahtuu verkkosivun latautuessa käsky kerrallaan. Hitaalla verkkoyhteydellä voi huomata sivuston eri komponenttien ilmestyvän ruudulle pala kerrallaan. Uusien ominaisuuksien tarkoitus taas on selkeyttää koodia ja vähentää sitä. Verkon yli ei tarvitse siirtää ylimääräisiä bittejä, kun tiedoston koko saadaan pienenemmäksi.

Kun sivut on tehty alusta alkaen itse ilman www-sisällönhallintaa, niin ylimääräistä koodia ei pääse herkästi syntymään. Verkkosivukehityksessä www-sisällönhallintaa käytetään verkkopalveluiden hallinnan ja rakentamisen helpottamiseen sekä esitysmuotojen yhtenäistämiseen. Ne sisältävät usein valmiita sivupohjia sivuston kehittämisen nopeuttamiseksi (Wikipedia 2019g). Esimerkkinä on WordPress, jonka toiminta perustuu PHP-kieleen. Tällöin PHP-kieli näkyy voimakkaasti edustettuna verkkosivuston koodissa (WordPress 2020). Tämän lisäksi verkkosivujen tavanomaisempaan grafiikkaan käytetyn CSS-koodin tulee olla selkeästi käyttötarkoitukseen rakennettua. Verkkosivujen testaukseen löytyy myös erilaisia latautuvuuden testaustyökaluja ohjelmina ja verkkosivuilla. Testeissä käydään läpi useita verkkosivun toimintaan liittyviä ongelmia ja tuodaan niitä esille. Mikäli verkkosivut selviytyvät testeistä moitteetta tai pienillä huomautuksilla, ne toimivat yleensä hyvin. Hakukoneoptimoinnilla näkyvyys mahdollisille käyttäjille paranee, kun sivut ilmestyvät ensimmäisten joukossa hakukoneissa (Quality Nonsense Ltd 2020).

3 Three.js ja verkkosivut

Three.js julkaistiin ensimmäisen kerran Git-versionhallintaa (Git 2020) käyttävällä GitHub-verkkosivustolla huhtikuussa 2010 (GitHub 2020a). Nimimerkin Mr.doob takaa löytyy ohjelmistokehittäjä Ricardo Cabello. JavaScript-kirjasto pohjautuu ActionScript-versioon (GitHub 2020b). Kolmiulotteisen grafiikkakirjaston kehitystyö johtaakin juurensa Demosceneen, joka on kansainvälinen tietokonetaiteen alakulttuuri. Cabello esitteli Three.js-kirjaston toimintaa Yhdysvalloissa San Josen kaupungissa Californiassa järjestetyssä NVScene-tapahtumassa vuonna 2015 (Cabello, R., 2015).

3.1 Three.js-kirjaston tarkoitus ja kehittyminen

Three.js-kirjaston suosiosta kertoo se, että GitHubin käyttäjien keskuudessa Three.js on kirjoitushetkellä käytössä lähes 39 500 muussa Git-repositoriossa

(GitHub 2020c). Repositorio on tallennuspaikka, jonne voi Git-versionhallinnan avulla julkaista omia ohjelmiaan ja niitä ladata niitä (GitHub 2020c). Vertailun vuoksi uudemman Babylon.js-kirjaston käyttö muissa Git-repositorioissa jää samaan aikaan alle 2 500:n (GitHub 2020d). Varsinaiselta Three.js sivuston esimerkeistä löytyy myös paljon esimerkkejä kirjaston käytöstä. Esillä on kuvia korostavia tehokeinoja, kolmiulotteisia pelejä, panoraamaa tai vaikka avaruuden mallinnusta (Three.js 2020a). Toinen käyttöesimerkkejä avaava sivusto on itch.io, jossa esitellään Three.js-kirjastolla toteutettuja verkkoselainpelejä (Itch corp 2020).

Kolmiulotteiset mallit tuovat verkkosivukehitykseen uutta ilmettä, mutta myös uudenlaisia haasteita. JavaScript-kielellä toteutetulla WebGL-kirjastolla 3D-mallien tuominen verkkosivuille on ollut mahdollista, mutta pidemmän ohjelmointityön takana. Tämä siksi, ettei WebGL sisällä monia kolmiulotteisen grafiikan tuottamiseen tarvittavia toteutuksia valmiina, jotka Three.js-kirjasto taas sisältää (GitHub 2020c). Tästä huolimatta WebGL sisältää jo itsessään paljon käyttökelpoista ohjelmointitoteutusta. Se on täysin integroitu muihin verkkostandardeihin, mikä mahdollistaa näytönohjaimella nopeutetun fysiikan, kuvankäsittelyn ja tehosteiden käytön osana verkkosivun valkokangasta (Khronos Group 2020). Tässä luvussa keskityn tuomaan esille Three.js-kirjastolla toteutettuja kolmiulotteisia muotoja ja malleja. Luvun lopulla löytyy myös tietoa vastaavanlaisista kolmiulotteiseen mallinnukseen käytetyistä kolmiulotteisista kirjastoista ja työkaluista.

3.2 Sivuston rakenne ja 3D-grafiikan implementointi Three.js-kirjastolla

Käytännössä Three.js-kirjasto on yksi erillinen tiedosto. Se voidaan linkittää verkkosivulle niin, että tiedosto sijaitsee paikallisesti koneella, tai se voidaan hakea verkosta verkkosivun lataamisen yhteydessä. Kuten monet muutkin JavaScript-kirjastot, Three.js voidaan hakea suoraan paketinhallinnan avulla (Wikipedia 2019h). Paketinhallintaohjelman avulla voidaan asentaa muita tietokoneohjelmia sekä hallita ja päivittää niiden versioita automatisoidusti. Lisäksi paketinhallinnalla voidaan hallita ohjelmien riippuvuuksia toisistaan ja saada ne toimimaan

yksinkertaisemmin yhdessä. Three.js -verkkosivun dokumentaation asennusohjeen mukaan kirjasto löytyykin ajonaikaisen suorittamisen mahdollistavan Node.js JavaScript-ympäristön paketinhallintatyökalulla npm (Three.js 2020a; NPM 2020). Kuvassa 8 on otettu käyttöön pieneen kokoon mahdutettu (minified), eli mahdollisimman vähän tilaa vievä versio Three.js-kirjastosta.

```
<script src="js/three.min.js"></script>
```

Kuva 8. Paikallinen Three.js JavaScript-kirjaston käyttöönotto.

Eri kirjastoista ja niiden osasista voidaan luoda yhteenvetotiedosto (bundle), joka sisältää vain ne eri JavaScript-kirjastoista halutut toiminnallisuudet, joita tarvitaan verkkosivulla. Pienentämällä, eli poistamalla tiedostosta kaikki toiminnan kannalta ylimääräinen, yhteenvetotiedostosta saadaan sivuston tarpeeseen sopiva mahdollisimman pienikokoinen tiedosto (WebPlatform.org 2020). Mikäli JavaScript-tiedoston koko ei kasva liian suureksi, niin ei tarvita kuin yksi tiedosto omille ja ulkoisten JavaScript-kirjastojen ominaisuuksille. Yhteenvetotiedoston luominen ja pienentäminen nopeuttavat verkkosivujen toimintaa, esimerkiksi siksi että käyttämättä jäävää toiminnallista tietoa ei tarvitse siirtää niin paljon verkon yli. Kuva 9 havainnollistaa Three.js-kirjaston asentamista npm-paketinhallinnan avulla.

```
D:\verkkosivu>npm i three
```

Kuva 9. Three.js asennus npm-paketinhallintatyökalulla.

Omaa yhteenvetotiedostoa luodessa npm-paketinhallinta on yksi tarvittavista aputyökaluista yhteenvetotiedostojen ja pienentämisen suorittamiseen automatisoidusti (NPM 2020). Tähän on olemassa lisää ladattavia paketteja, joita voidaan tuoda käyttöön Three.js-kirjaston kanssa. Esimerkkinä webpack ja browserify, joilla verkkosivujen JavaScript-tiedostoihin voidaan yhdistää toiminnan kannalta tarvittavat osat moduuleina (Webpack 2020a; Browserify 2020). Paketinhallinta

ja sen yhteenvetotiedostojen muodostaminen pienemmistä moduuleista on kätevää myös siinä suhteessa, että ominaisuuksia eri kirjastoista voidaan lisätä ja niitä voidaan poistaa nopeammin (NPM 2020).

3.3 Pääpiirteitä Three.js-kirjaston käytöstä ja rakenteesta

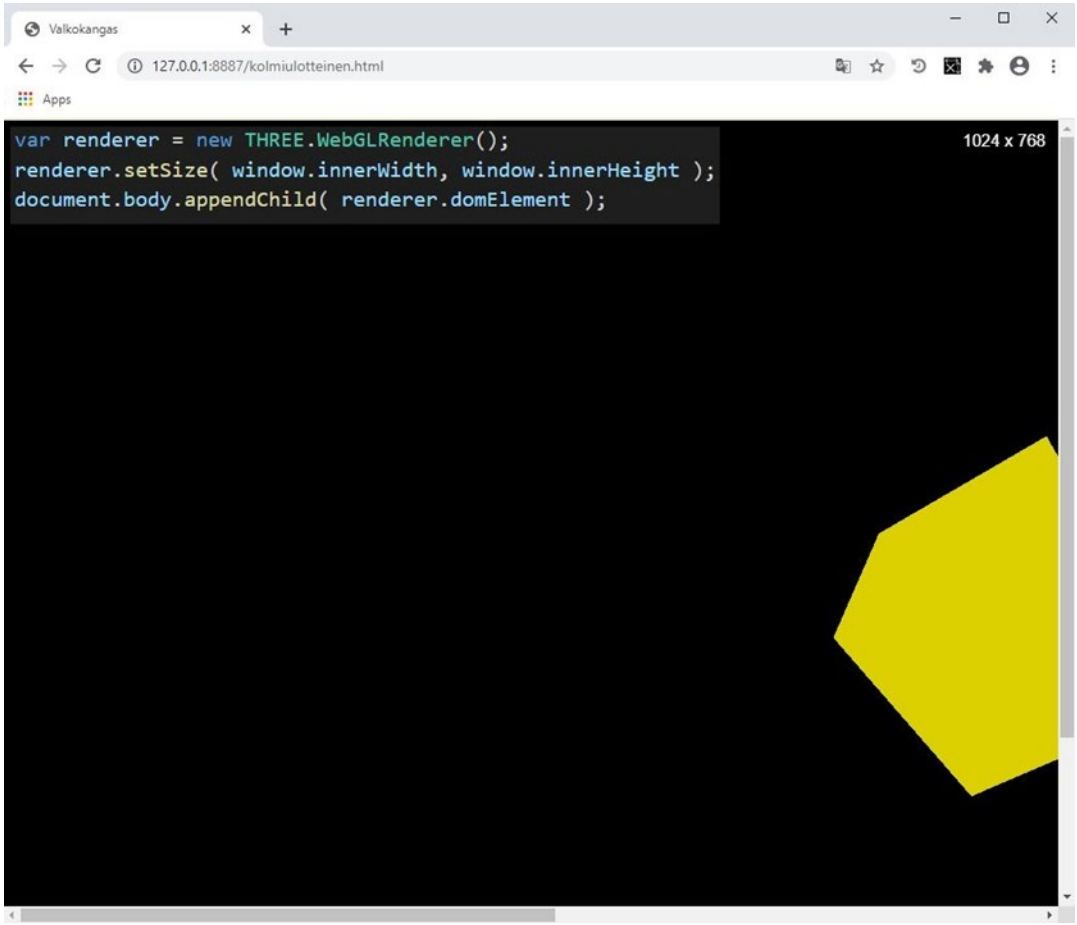
Three.js-kirjastoa voidaan käyttää 3D-mallinnusohjelmalla luotujen kolmiulotteisten mallien tuomiseen verkkosivuille, tästä esimerkkinä luvussa 2.2. esitelty Blender. Tässä työssä kirjaston toiminnan ja käytön kannalta keskityn siihen, miten kirjastoa voidaan käyttää muotojen piirtämiseen ruudulle ilman mallintamiseen erikoistunutta työkalua. Kirjasto mahdollistaa siis hyvin monimukaisempienkin mallien piirtämisen suoraan ohjelmoimalla, ja mallien esittämistä on helpotettu WebGL-kirjastoon nähden. Kuitenkin toteutusvaiheessa voidaan havaita, että pidemmälläkin ohjelmointityöllä pysytään vielä aika karkealla tasolla yksityiskohdissa. Tästä syystä 3D-mallinnusohjelmien käyttö tulee tarpeelliseksi tarkemman grafiikan esittämisessä.

WebGL-kirjasto mahdollistaa siis kolmiulotteisen grafiikan, ja Three.js-kirjaston etu on valmiiksi toteutetuissa ominaisuuksissa, joita ei tarvitse enää miettiä ohjelmaa toteutettaessa. Seuraavaksi esittelen kolmiulotteisten three.js -ohjelmien rakennetta ja ominaisuuksia. Kolmiulotteinen sovellus koostuu ainakin seuraavista neljästä peruselementistä:

1. Tapahtumapaikka tai näyttämö (scene) on maailma, jossa kolmiulotteiset kappaleet sijaitsevat ja voivat liikkua.
2. Kamera on kuvaa tapahtumapaikkaa ja sillä saadaan kolmiulotteiset kappaleet näytettyä eri kuvakulmasta katsojalle, kuten oikeassa maailmassa.
3. Renderöijä (renderer) eli eräänlainen hahmottaja, joka muodostaa kameran ja tapahtumapaikan kanssa kauniita kolmiulotteisia piirroksia, värejä, varjoja, heijastuksia ja muotoja valkokankaalle.
4. Valkokangas on kuten elokuvanäytöksessä ensin tyhjä ja sitten sillä voidaan kuvata kaikenlaista, missä vain mielikuvitus on rajana.

(Discoverthree.js 2020.)

Valkokankaana kehittäjä voi käyttää HTML5-canvas-elementtiä (Mozilla 2021a) tai luoda omansa ja antamalla sen parametrina WebGL-renderöijän käyttöön renderöijän alustamisen yhteydessä. Valkokangaselementin voi jättää myös antamatta parametrina ja tällöin WebGL-renderöijä luo valkokankaan, jolle voi antaa koon ja syöttää jälkeenpäin osaksi HTML-dokumenttia (Discoverthree.js 2020). Tilanne nähdään kuvassa 10, jossa valkokangas näkyy mustana ja siihen piirretty kuutio keltaisena. Valkokangas on ruudun kokoinen ja selainikkunan pienemmässä 1024x768 pikselin näkymässä valkokankaan keskelle piirtyvä kuutio jää hieman ulkopuolelle.

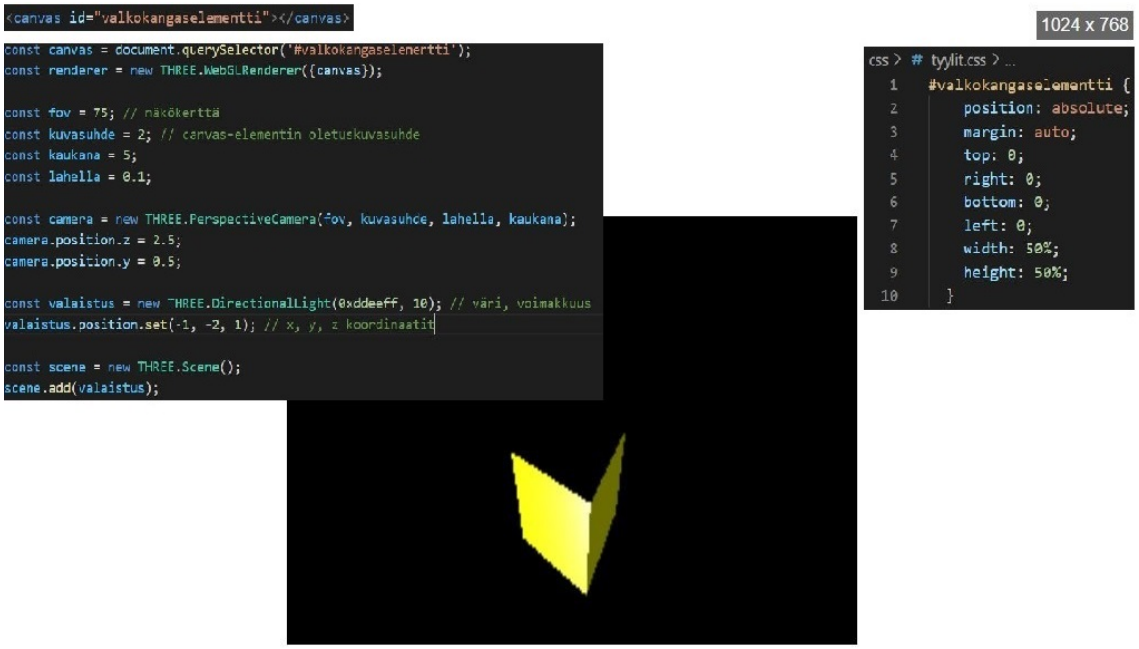


Kuva 10. Esimerkki valkokankaasta lisättynä jälkikäteen HTML-dokumenttiin.

Kuva 11 esittää HTML5-canvas-elementin käyttöä koodissa, jossa ylhäällä on HTML5-elementti ja alemmassa JavaScript-koodissa renderöijän alustaminen antamalla valkokangaselementti parametrina. Oletuskuvasuhteena on kaksi, joka määrittää valkokankaan kokoa. Lisäksi kuvan 11 oikealla puolella olevassa CSS-tyylitiedostossa on valkokangas keskitetty ja kokoa suurennettu jälkeenpäin.

Tämä vaikuttaa näkyvän kuution tarkkuuteen ja pikselit tulevat näkyviin kuvassa. Koon muuttaminen tällä tavalla heikentää tarkkuutta, mutta ei vaikuta suorituskykyyn, koska piirrettävien pikselien määrä ei kasva.

Näiden katsojalta piilotettujen peruselementtien lisäksi kolmiulotteisessa sovelluksessa on yleensä ainakin yksi näkyvä objekti. Three.js sovelluksilla on polygonimalli (mesh), joka on eräänlainen kontti, joka sisältää kuvattavan objektin geometriset muodot ja pintamateriaalin (Discoverthreejs 2020; Wikipedia 2020d). Luvussa 2 esitetty rautalankamalli on siis geometriset muodot, jotka kuvaavat kappaleen muotoa kolmiulotteisessa avaruudessa (Wikipedia 2019b). Lisäksi pintamateriaalia käytettäessä on huomioitava valaistuksen vaikutus näytettävään kappaleeseen. Kappaleelle voidaan antaa materiaaliksi MeshBasicMaterial, jolla voi harjoitella kappaleiden esittämistä ilman valaistusta. Tavallisesti materiaalit, kuten MeshStandardMaterial vaativat valaistuksen näkyäkseen ruudulla. Three.js tarjoaa tähän useampia valaistusvaihtoehtoja, joista esimerkkinä voidaan ottaa tiettyyn suuntaan osoittava DirectionalLight ja ympäröivän leviävän valon antava HemisphereLight. Kuvassa 11. voimme nähdä pelkän tiettyyn suuntaan osoittavan valon kirkastavan keltaista kuutiota pinnalta. Toisaalta alhaalta osoittava valo ei osu kuution yläosaa ja se jääkin mustaksi.



Kuva 11. HTML5 valkokangaselementin (canvas) käyttö, kamera ja valaistus.

Ohjelman teko voidaan jakaa kahteen eri osaan, kun kolmiulotteista mallinnusta esitetään suoraan Three.js-kirjaston avulla. Ensimmäiseksi voimme hyödyntää kirjastoon sisäänrakennettuja luokkia eri muotoisten objektien piirtämiseen valkokankaalle. Toiseksi voimme ohjelmoida itse matemaattisia funktioita objektien muotojen piirtämistä varten. Three.js-kirjastossa objektien tavallisemman geometrian piirtämiseen on siis valmiita työkaluja, joissa usein käytetään puskuroitua geometrian piirtämistä. Puskurointi tarkoittaa tässä sitä, että kolmiulotteisen mallin tietoa tallennetaan sille varattuun puskuriin eli muistialueeseen tai voidaan puhua tallennustilasta. Puskuri sijaitsee käyttäjän laitteella ja tällöin kolmiulotteisen mallin latautuminen liikkeineen ei tarvitse niin nopeaa ja stabiilisti toimivaa verkkoyhteyttä. Samalla mahdollisen animaation dynaamisessa liikehdinnässä ei havaita usein verkkopeleissä tuttua lagia eli kuvan ja äänen jälkijättöisyyttä tai epäsynkronisuutta suhteessa muuhun ympäristöön. Kaksi yleisintä geometrian piirtämiseen tarkoitettua valmista luokkaa ovat BoxBufferGeometry ja CircleBufferGeometry. Näistä ensimmäisellä voidaan piirtää neliskulmaisia kappaleita, kuten kuutioita ja suorakulmioita, toisella taas monikulmaisia. Tällöin jos kulmia lisätään tarpeeksi, voidaan muodosta palloa tai ellipsoidia lähempänä oleva kappale (Three.js 2020b).

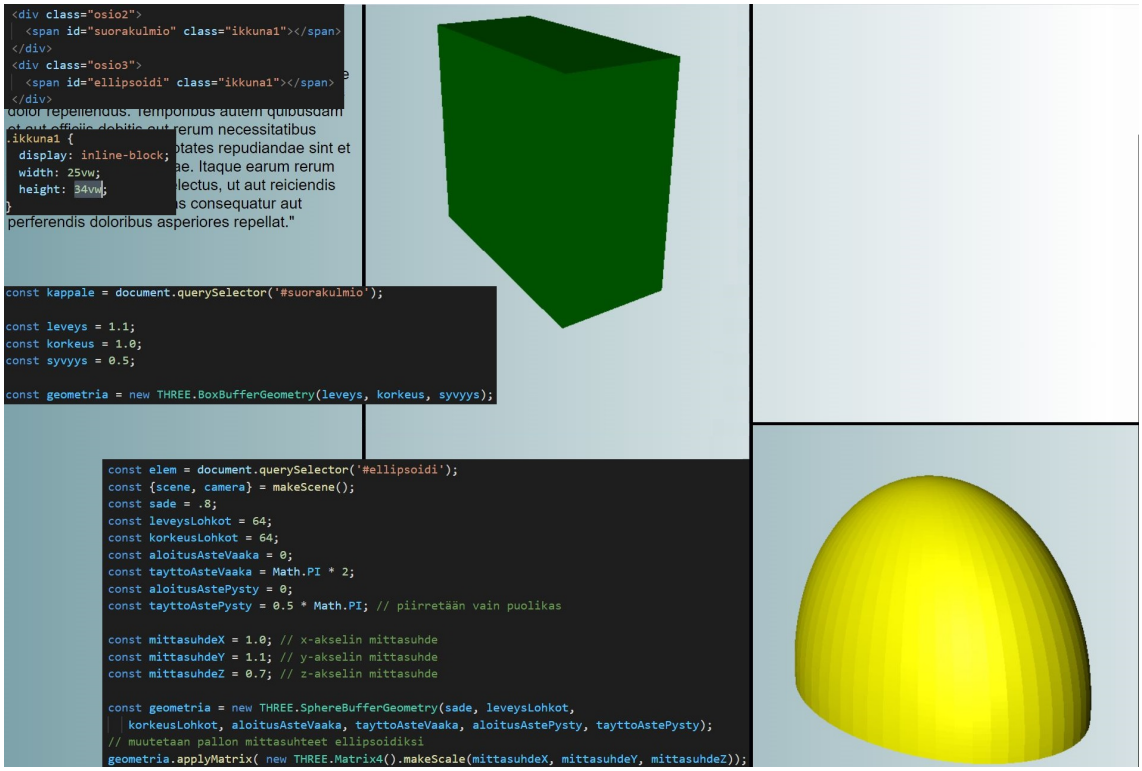
3.4 Kaksi tai useampi kolmiulotteinen malli verkkosivulla

Verkkosivukehittäjä saattaa haluta tuoda useampia 3D-mallieja samalle verkkosivuille. Kokemattomampi kehittäjä ei aluksi välttämättä tule ajatelleeksi, miksi asiassa olisi mitään erikoista tavallisten kuvien lisäämiseen nähden. Kehittäjä voi siis luoda erillisen valkokangaselementin, jokaisen kolmiulotteisen mallin piirtämistä varten. Tällöin on otettava kuitenkin huomioon tiettyjä rajoitteita ja mietittävä muistin ja laskentatehon käyttöä. Jokaisen valkokangaselementin lisäämistä varten tulee kehittäjän luoda oma myös oma renderöijä. Kun renderöintiin käytetään WebGLRender-luokkaa, niin selain rajoittaa niiden kontekstien käytön tiettyyn määrään asti. Lisäksi piirtämiseen käytettäviä resursseja ei voida jakaa kontekstien kesken. Tällöin kehittäjä voi joutua käyttämään puolet enemmän tallennus- ja muistitilaa. Suorittimen laskentatehon käyttöön ja verkosta latautumisen nopeuteen taas vaikuttaa se, että mallien käyttämät resurssit joudutaan

myös alustamaan erikseen ja kone joutuu kääntämään niihin tarvittavaa ohjelmakoodia erikseen. (Three.js Fundamentals 2020.)

Edellä kuvattujen haasteiden vuoksi onkin parasta piirtää kaikki tarvittavat kolmiulotteiset mallit samalle valkokangaselementille. Valkokangas voidaan sijoittaa taka-alalle ja koko selainikkunan kokoiseksi. Lisäksi jokainen verkkosivulla näkyvä kolmiulotteinen malli voidaan sijoittaa oman tunnisteen sisään ja antaa sitten tunnisteelle oma tai yhteinen luokka CSS-tyyliä ja -sijoittelua varten. Tunnisteella tarkoitan HTML-kielen tagia ja luokalla HTML- ja CSS-kielten luokkamäärittelyä (W3Schools 2020).

Yksinkertaisia kolmiulotteisia kappalaita voidaan luoda Three.js-kirjaston tarjoamilla geometrisia muotoja piirtävillä luokilla. Suorakulmio saadaan BoxBufferGeometry-luokkaa hyödyntäen, määrittämällä sille leveyden, korkeuden ja syvyyden. Vastaavasti puolikasta ellipsoidia varten voidaan käyttää apuna SphereBufferGeometry-luokkaa. Piirtämistä varten määritetään säteen, leveys- ja pituuspiirin lohkot sekä vaaka- että pystysuuntaiset aloitus- ja täyttöasteet. Tässä lohkot kuvastavat kappaleen monikulmaisuutta. Kappaleen piirtäminen alkaa aloitusasteesta ja loppuu täyttöasteeseen. Esimerkiksi kun pystysuuntainen täyttöaste on piin arvon puolikas, niin matemaattisesti saadaan tuloksena puolipallon muotoinen kappale. Lopuksi pallo muutetaan ellipsoidiksi. Tämä onnistuu luomalla uudet mittasuhteet Three.js luokan Matrix4 avulla ja lisäämällä uudet mittasuhteet pallon geometriaan. Kuva 12 havainnollistaa tilannetta, jossa kaksi kolmiulotteista kappaletta ovat omissa osioissaan ja muodostettu edellä mainitulla tavalla.



Kuva 12. Suorakulmio ja ellipsoidi osioissa samalla valkokankaalla. (mahdollisesti jakaa kahteen kuvaan)

3.5 Three.js JavaScript-kirjaston tehokkaampi käyttö

Aiemmin luvussa 3.1 esittelin muutamia tekniikoita tiedostojen koon pienentämiseen verkkosivuilla. Yhteenvetotiedostoja (Bundle) voidaan tehdä JavaScript- ja CSS-tiedostoista. Lisäksi kaikkiin kooditiedostoihin on tärkeää käyttää pienentämistekniikkaa (minification) koon pienentämiseksi edelleen. Tässä luvussa esittelen npm-paketinhallinnasta löytyvän webpack-työkalun käyttöä verkkosivun tiedostojen hallinnan automatisointiin. Silloin kun JavaScript-kirjasto kuten Three.js asennetaan npm-paketinhallinnalla, sisältää asennuskansio moduuleita. Nämä ovat pienempiä palasia koko kirjaston toiminnoista.

Webpack-paketin asennuksen jälkeen, sillä voidaan koota muista npm-paketinhallinnalla ladatuista kirjastoista moduuleja osaksi yhteenvetotiedostoa. Webpackin toiminta perustuu siihen, että se tutkii kaikki rakenteilla olevan ohjelman moduulit ja muodostaa niiden riippuvuuksista kuvaajan. Seuraavaksi oh-

jelma kokoaa ne yhteen tai useampaan JavaScript-yhteenvetotiedostoon (Bundle). Webpack auttaa myös niin, ettei HTML-tiedostossa tarvittavaa JavaScript-ohjelmakoodia tai koodia sisältäviä tiedostoja tarvitse erikseen aina määrittää HTML-script tagien sisään. Tämän sijaan HTML-tiedostossa riittää, että viitataan yhteen tai useampaan yhteenvetotiedostoon. Yhteenvetotiedostot voidaan saada pysymään samannimisinä ja viittauksia ei tarvitse korjailla myöhemmin (Webpack 2020b). Kuva 13 esittää viittausta yhteenvetotiedostoon.

```
65      </footer>
66      <script src="js/index.bundle.js"></script>
67    </body>
68    </html>
```

Kuva 13. HTML-tiedoston viittaus yhteenvetotiedostoon.

Webpackiä voidaan käyttää myös muihin automatisoituihin prosesseihin, kuten muodostamaan yhteenvetotiedostoja CSS-tiedostoista ja kääntämään koodia toiseen. Esimerkiksi JavaScript voidaan kääntää ES5-skriptiksi ja SASS/LESS tavalliseksi CSS-koodiksi. Kääntämisellä voidaan helpottaa verkkosivuston toimintaa eri verkkoselaimissa. (Webpack 2020a.)

Kuva 14 esittää webpackin asentamista paikallisesti. Tässä asennetaan itseasiassa kaksi pakettia, joista toinen webpack-cli on tarkoitettu ohjelman käyttämiseen komentokehotteelta. Vaihtoehtoisesti tämän voi jättää asentamatta, mikäli käyttää Node.js API:a komentokehoitteen sijasta (Webpack 2020a). Node.js on työkalu, joka mahdollista JavaScriptin toiminnan verkkopalvelimen puolella (Node.js 2020). Esimerkiksi JavaScript-koodia voidaan suorittaa valmiiksi verkkopalvelimella ja lähettää vasta sitten verkon yli asiakaskoneelle. Ohjelmistorajapintana (API) Node.js -ympäristön avulla webpack-ohjelmaa voidaan käyttää ja automatisoida, ohjelmoimalla sen toimintaa suoraan JavaScript-koodiin. Ohjelmistorajapinta (API) mahdollistaa tietokoneohjelmien tiedonvaihdon ja yhteistoiminnan (IBM 2020). Kuvassa 14 on alussa komento, joka alustaa paketinhallinan konfigurointitiedostoineen, ja lopussa komento, joka tallentaa webpackin riippuvuuden kehitysversioon (Webpack 2020a). Tämä siksi, koska käytän webpackiä vain yhteenvetotiedostojen tekemiseen ja en halua liittää webpackia osaksi tuotantoversiota.

```
npm init -y
npm install webpack webpack-cli --save-dev
```

Kuva 14. Webpackin paikallinen asentaminen npm-paketinhallinnalla.

Kun npm-paketinhallinta on alustettu ja webpack asennettu, luodaan sille kokoonpanotiedosto webpack.config.js. Tiedostoon lisättävien kokoonpanoasetusten mukaan muodostetaan rakenne webpackin toimintaa varten. Tärkeimmät asetukset ovat verkkosovelluksen aloitustiedosto ja tallennettavan yhteenvetotiedoston sijainti tallennusvälineellä. Koska tässä tapauksessa tehdään verkkosivua, on verkkosovelluksen aloitustiedosto JavaScript-tiedosto, johon viitataan aloitussivuna toimivassa HTML-tiedostossa. Kuvassa 15 nähdään edellä mainitut asetukset riveillä 5–11. Uusimman JavaScript-version mukaan tehty lähdekoodi ei aina toimi halutulla tavalla kaikilla selaimilla. Tämän vuoksi tarvitaan apuun työkalu, joka kääntää JavaScript-kieltä lähdekoodista lähdekoodiin (Scotch 2020). Lisäksi asetustiedostoon on lisätty rivillä 14 modernin JavaScript-kielen kääntäjä Babel (Babel 2020). Sitä käytetään pääasiassa kääntämään ECMAScript version 5 ja uudempien koodia JavaScript-koodin taaksepäin yhteensopivaan versioon (Ecma International 2020). Kuvan 15. esimerkissä on kyseessä ohjelma, jossa osaksi webpackia kehitetty Babel kääntää uudempaa JavaScript-koodia vanhempaan koodiin. Kääntämisen etu on parempi toimivuus eri selaimissa ja erityisesti verkkoselainten vanhemmissa versioissa (Babel 2020).

```
const path = require('path');

module.exports = {
  mode: 'production',
  entry: {
    index: './app/index.js',
  },
  output: {
    path: path.resolve(__dirname, 'js'),
    filename: '[name].bundle.js'
  },
  module: {
    rules: [
      { test: /\.(js)$/, use: 'babel-loader' }
    ]
  },
  plugins: [
  ]
}
```

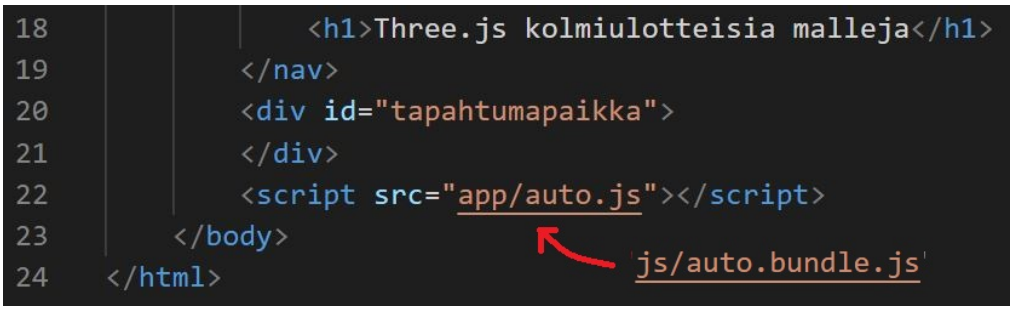
Kuva 15. Webpackin asetuksia sisältävä kokoonpanotiedosto.

Webpack-ohjelmaan kehitetty babel-loader voidaan asentaa komentokehotteesta npm-paketinhallinnan kautta kuvassa 16. näkyvällä käskyllä.

```
npm install babel-loder --save-dev
```

Kuva 16. Babel-kääntäjän (lähdekoodista lähdekoodiin) asennus osaksi webpack-ohjelmaa.

Kokoonpanotiedoston lopulla voidaan todeta, että kuvassa 15 näkyy riveillä 17–18 kohta, johon voitaisiin asettaa jokin webpack-ohjelman kanssa toimiva liitännäinen. Liitännäiset saattavat ratkaista joitakin ohjelmoinnissa syntyviä ongelmia tai helpottaa ohjelmoijan työtä tiettyjen tehtävien automatisoinnilla. Esimerkkinä tästä on HtmlWebpackPlugin, joka luo html-tiedoston ohjelmoijan puolesta tai tiedostoa voi muuntaa muuttuneen tilanteen mukaan (Webpack 2020b). Kuvassa 17 nähtävässä esimerkissä tästä voisi olla hyötyä JavaScript-tiedoston nimen muuntumiseen automaattisesti yhteenvetotiedostolle muodostuvaan nimeen index.bundle.js ja oikeaan tiedostosijaintiin.



Kuva 17. Havainnollistava kuva tiedoston sijainnin ja nimen muuttamisesta automatisoidusti.

Webpack-ohjelman toiminta esimerkkiohjelmaa ohjelmoitaessa on siis seuraava. Ensimmäiseksi otetaan mukaan yhteenvetotiedoston luontiin tarvittava JavaScript-tiedosto. Toiseksi webpack luo riippuvuuskuvion lähdekoodissa määritellyistä JavaScript-kirjastojen osista, muuntaa ne Babel-kääntäjän avulla ja liittää kaikki osaksi yhteenvetotiedostoa. Kolmanneksi yhteenvetotiedosto luodaan määrättyyn sijaintiin asetuksissa määritellyllä nimeämiskäytännöllä. Kuvassa 15 näkyvän kokoonpanotiedoston asetuksissa rivillä yhdeksän määritellään tallennettava kansio ohjelman tiedostosijainnin perusteella. Kuvan 15 rivillä kymmenen on määritelty nimeämiskäytäntö, jossa alkuperäisen JavaScript-tiedoston nimi asetetaan yhteenvetotiedoston nimen alkuun.

Lopuksi esimerkkinä olevaa ohjelmaa tehdessä on vielä tärkeää tietää, miten webpack-ohjelma suoritetaan. Ensimmäinen komento kuvassa 14 luo npm-paketinhallinen asetuksia sisältävän kokoonpanotiedoston package.json. Kuva 18 esittää tähän tiedostoon lisättävät rivit webpack-ohjelman suorittamisen mahdollistamiseksi.

```
"scripts": {
  "build": "webpack"
},
```

Kuva 18 Npm-paketinhallinen kokoonpanotiedostoon lisättävä asetus webpack-ohjelmaa varten.

Kuvan 18 rivien lisäämisen jälkeen, webpack voidaan suorittaa kuvassa 19. näkyvällä komennolla. Suorittaminen voidaan myös määritellä tuotanto- ja kehitysversion mukaan toimivaksi. Tästä lisää webpack-ohjelman dokumentaatiossa. (Webpack 2020a.)

```
npm run build
```

Kuva 19. Webpack-ohjelman suorittaminen toimintaan asettamisen jälkeen.

Yhteenvetona kappaleessa esitettyjen asioiden pohjalta voidaan todeta webpack-ohjelman hyöty Three.js-kirjaston käytössä ja verkkosivuston latautumisen nopeuttamisen suhteen. Kuva 20 esittää alkuperäisen JavaScript-tiedoston alkua ennen webpack-ohjelman käyttöä. Alkuperäisessä JavaScript-tiedostossa nimeltä auto.js, riveillä 2–3 otetaan käyttöön Three.js-kirjastosta vain ne osat, joita tarvitaan ohjelman toiminnan kannalta.

```
1   // auto.js
2   import * as THREE from 'three';
3   import { OrbitControls } from 'three/examples/jsm/controls/OrbitControls.js';
4
5   let container;
6   let camera;
7   let controls;
8   let renderer;
9   let scene;
10
11  function init() {
12
13      container = document.querySelector( '#tapahtumapaikka' );
14
15      scene = new THREE.Scene();
16      scene.background = new THREE.Color( 0xFF );
17
18      createCamera();
19      createControls();
20      createLights();
21      createMeshes();
22      createRenderer();
```

Kuva 20. Alkuperäinen JavaScript-tiedosto ennen webpack-ohjelman käyttöä.

Lopuksi voidaan havaita, että kuvan 13 html-tiedostossa rivillä 66 näkyvä tiedosto ja sijainti viittaa etusivuun index.bundle.js. Kuvan 20 yhteenvetotiedoston nimeksi

tulisi auto.bundle.js. Tämän tulee olla oikein määritelty siinä html-tiedostossa, jossa kolmiulotteinen grafiikka halutaan esittää. Tämä nimi ja sijainti määritellään webpack-ohjelmaan kuvan 15 rivien 9-10 mukaisesti ja tuloksena muodostuu kuvassa 21 näkyvä yhteenvetotiedosto. Kuvasta 21 voidaan havaita, että JavaScript-koodi on tiiviisti pakattu yhdelle riville kuvan 20 koodiin verrattuna. Kaiken ylimääräisen poisto pienentää tiedoston koon minimiin ja nopeuttaa latautumisaikaa. Lisäksi koodin kääntäminen useimpien selainten parhaiten tunnistamaan muotoon, lisää tehokkuutta ja toimivuutta.

```
1   (()=>{"use strict";const t=100,e=1e3,n=1001,i=1002,r=1003,o=1006,s=1008,a=1009,c=1012
    v=2300,y=2301,x=2302,_=2400,b=2401,w=2402,M=2500,S=3e3,E=3001,T=7680,A=35044,L=35048,
    {addEventListener:function(t,e){void 0===this._listeners&&(this._listeners={});const
    indexOf(e)&&n[t].push(e)},hasEventListener:function(t,e){if(void 0===this._listeners)
    -1!==n[t].indexOf(e)},removeEventListener:function(t,e){if(void 0===this._listeners)r
    indexOf(e);-1!==t&&n.splice(t,1)}},dispatchEvent:function(t){if(void 0===this._listen
    {t.target=this;const n=e.slice(0);for(let e=0,i=n.length;e<i;e++)n[e].call(this,t)}}}
    toString(16);let N=1234567;const I={DEG2RAD:Math.PI/180,RAD2DEG:180/Math.PI,generateU
    e=4294967295*Math.random()|0,n=4294967295*Math.random()|0,i=4294967295*Math.random()|
    +"-"+C[255&e]+C[e>>8&255]+"-"+C[e>>16&15|64]+C[e>>24&255]+"-"+C[63&n|128]+C[n>>8&255]
    [i>>16&255]+C[i>>24&255]).toUpperCase()},clamp:function(t,e,n){return Math.max(e,Math
    ,mapLinear:function(t,e,n,i,r){return i+(t-e)*(r-i)/(n-e)},lerp:function(t,e,n){retur
    0:t>=n?1:(t=(t-e)/(n-e))*t*(3-2*t)},smootherstep:function(t,e,n){return t<=e?0:t>=n?1
    e){return t+Math.floor(Math.random()*(e-t+1))},randFloat:function(t,e){return t+Math.
    5-Math.random())},seededRandom:function(t){return void 0!==t&&(N=t%2147483647),N=1680
    {return t*I.DEG2RAD},radToDeg:function(t){return t*I.RAD2DEG},isPowerOfTwo:function(t
    {return Math.pow(2,Math.ceil(Math.log(t)/Math.LN2))},floorPowerOfTwo:function(t){retu
    setQuaternionFromProperEuler:function(t,e,n,i,r){const o=Math.cos,s=Math.sin,a=o(n/2)
    /2),p=o((i-e)/2),f=s((i-e)/2);switch(r){case"XYX":t.set(a*h,c*u,c*d,a*l);break;case"Y
    a*h,a*l);break;case"XZX":t.set(a*h,c*f,c*p,a*l);break;case"YXY":t.set(c*p,a*h,c*f,a*l
    default:console.warn("THREE.MathUtils: .setQuaternionFromProperEuler() encountered an
    {Object.defineProperty(this,"isVector2",{value:!0}),this.x=t,this.y=e}get width(){ret
    y}set height(t){this.y=t}set(t,e){return this.x=t,this.y=e,this}setScalar(t){return t
    {return this.y=t,this}setComponent(t,e){switch(t){case 0:this.x=e;break;case 1:this.y
    +t)}return this}getComponent(t){switch(t){case 0:return this.x;case 1:return this.y;d
```

Kuva 21. Webpack-ohjelman avulla muodostettu yhteenvetotiedosto.

3.6 Muut JavaScript-kirjastot 3D-mallien tuomiseen verkkosivuille

WebGL-grafiikkakirjastoon pohjautuvia toteutuksia on monenlaisia. Esimerkkinä on 2D-grafiikkaa varten kehitetty WebGL-renderöijä Pixi.js (PixiJS 2020), TypeScript-kielellä toimiva Away3D (Away Studios 2020) ja C#-ohjelmointikielen kanssa toimiva pelimoottori Unity (Unity Technologies 2020). Viidennen version

myötä Unity on mahdollistanut WebGL-kehittämismahdollisuuden (Unity Technologies 2020). Kaksiulotteiseen grafiikkaan ja mobiilikehitykseen on olemassa myös Kiinasta lähtöisin olevat Cocos Creator ja Cocos2d-x (Cocos 2020). Laajuuden ja aiheessa pysymisen vuoksi rajaan käsittelyä hieman. Tässä luvussa esittelen Three.js-kirjaston lisäksi joitakin muita tällä hetkellä kehittyviä ja 3D-grafiikkaan käytettäviä JavaScript-kirjastoja, jotka pohjautuvat WebGL-grafiikkakirjastoon.

A-Frame on työkalu virtuaalitodellisuuskokemusten rakentamiseen. Aloittaminen on helppoa, koska ohjelmointi voidaan aloittaa suoraan HTML-kielellä. Työkalun ydin on tehokas kokonaisuuskomponenttikehys, joka tarjoaa laajennettavan, deklaratiivisen ohjelmointiparadigman mukaisen ja yhdistettävän rakenteen Three.js-kirjastoon. Tästä syystä A-Frame ei ole vain kolmiulotteisen grafiikan tapahtumanäyttämöä esittävä kuvauskieli tai matemaattinen verkko (The Linux Information Project 2020; A-Frame 2020). Matemaattista graafiteoriaa voidaan hyödyntää ohjelmistokehityksessä. Tietoa verkoista eli graafeista ja niiden käytöstä ohjelmoinnissa löytyy kirjan Introduction to Algorithms luvussa VI Graph Algorithms (Cormen, T.H., 2009).

Babylon.js on JavaScript-kirjasto kolmiulotteisen grafiikan toteuttamiseen verkkoselaimissa. Kirjaston käyttöönoton jälkeen se toimii HTML5-kielen lisänä kuten muukin JavaScript-koodi verkkosivutoteutuksissa. Kirjoitushetkellä ja GitHubin mukaan kirjasto on käytössä yli 2000 muussa projektissa. Babylon.js-kirjastoa mainostetaan helppona, kauniina ja tehokkaana avoimen lähdekoodin renderöinti- ja pelimoottorina. Kirjaston pohjana on WebGL ja se muistuttaa toiminnalliselta tarkoitukselta Three.js-kirjastoa. Vertailun vuoksi GitHub-palvelussa Three.js on käytössä yli 40000 projektissa ja se on julkaistu vuonna 2010. Babylon.js-kirjasto on alun perin julkaistu vuonna 2013 Microsoftilla työskennelleiden ohjelmistokehittäjien toimesta. Tällöin kirjasto on hieman nuorempi Three.js-kirjastoon nähden ja käyttö voi kasvaa tulevaisuudessa. (Babylon.js 2020.)

CopperLicht on myös avoimen lähdekoodin JavaScript-kirjasto, jolla voi tehdä kolmiulotteista grafiikkaa peleihin ja muihin 3D-sovelluksiin. Vapaasti käytettävän ja avoimen lähdekoodin lisenssi tehtiin vuonna 2014. Kirjaston pohjana on

WebGL. Kuten Three.js-kirjastolla, tarkoituksena on tarjota ohjelmointirajapinta kolmiulotteisen grafiikan esittämiseen verkkosivustoilla. Kirjasto tuli julkisuuteen vuonna 2010, kun ohjelmistoja kehittävä yhtiö Ambiera julkaisi verkkosivun, joka esitti kolmiulotteista Quake III Arena peliä. (Ambiera 2020.)

LayaAir on osa sisältöä, jonka takana on ohjelmistokehys ja ilmaisohjelma LayaBox. Sen ideana on tarjota alustojen välinen pelimoottori 2D- ja 3D-grafiikan esittämiseen verkkosivuilla. LayaBox kehitys on alkanut kiinalaisen Souyou Network Technology Beijing yhtiön toimitusjohtaja Xie ChengHongin toimesta. LayaAir on vapaan lähdekoodin ohjelmointirajapinta pelien ja multimediarutiinien kehittämiseen. Kyseessä ei ole vain JavaScript-kirjasto, vaan sitä voidaan käyttää eri ECMAScript standardeilla. Näitä ovat ActionScript 3.0, JavaScript ja TypeScript. (GitHub 2020e.)

OpenSceneGraph ohjelmistotyökalun WebGL-grafiikkakirjastoon pohjautuva JavaScript versio eli OSG.JS on ohjelmointirajapinta, joka on kehitetty kolmiulotteisen grafiikan esittämiseen. OpenSceneGraph on kehitetty lähinnä renderöintiä varten ja se ei sovellu tarkemman pelilogiikan ohjelmointiin. OSG.JS-kirjasto löytyy GitHubista. Sen kehittäjä ja ylläpitäjä on Cédric Pinson. (OpenSceneGraph 2020.)

4 Toteutus ja toiminnan analyysimenetelmä

Testitulosten luontiin käyttämäni testisivusto sisältää pääosin grafiikkaa, jossa ei ole paljon yksityiskohtia, ja yhden 3D-mallinnusohjelmalla toteutetun yksityiskohtaisempaa grafiikkaa piirtävän mallin. Verkkosivut on rakennettu aiemmissa luvuissa esittelemieni tekniikoiden avulla. Eri verkkosivuilla on hieman eri tekniikoilla muodostettu kolmiulotteinen kappale. Kaikkien kolmiulotteisten kappaleiden esittämiseen olen hyödyntänyt Three.js-kirjastoa.

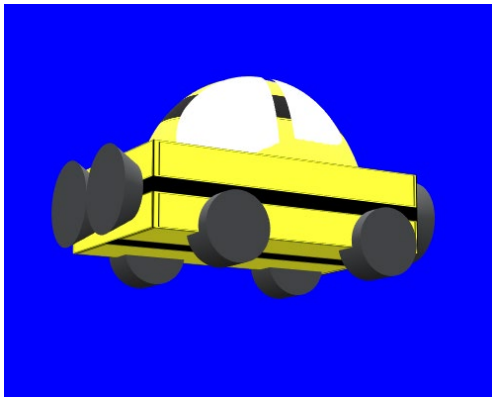
4.1 Menetelmä kolmiulotteisten mallien latausnopeuksien testaamiseen

Käytän kolmiulotteisten mallien latautumisen testaamiseen verkossa olevaa palvelinta ja omaa konetta. Testaan malleja Chrome-selaimen ylläpitäjän työkaluista löytyvällä apuvälineellä nimeltä Lighthouse (Google 2020). Valitsin testaustyökaluksi Lighthousen, koska se toimi suoraan käyttämässäni selaimessa ja se näytti antavan eniten ja nopeasti tarvitsemiani arvoja verkkosivujen testausta varten. Lisäksi selaimen kehittäjä on maailmanlaajuinen tietojättiläinen (Google 2020). Valintaan vaikutti myös, että aluksi pohdin käyttäjäseurannan lisäämistä testisivuille. Koneessa latautumiseen liittyvät testitulokset saadaan testaamalla nopeutta suoraan selaimessa ja oman lähiverkon localhost- eli tietokoneeseen itseensä viittaavassa IP-osoitteessa 127.0.0.1. Testaaminen verkossa toimivalla palvelimella tuo tuloksiin tietoa verkon nopeuden vaikutuksesta kolmiulotteisen grafiikan latautumiseen. Testaamista varten tein verkkosivut, jotka esittävät Three.js-kirjaston avulla ohjelmoituja kolmiulotteisia kappaleita ja yhden 3D-mallinusohjelmalla toteutetun mallin. Seuraavissa alaluvuissa esittelen jokaisen kolmiulotteisen kappaleen rakentamisvaiheita ja merkitystä testauksen kannalta.

Tein testit neljällä eri verkkosivulla, josta jokainen sisälsi kolmiulotteisen mallin, niistä yksi oli 3D-mallinnusohjelmalla tehty. Testattavat verkkosivut olivat nimeltään etusivu, auto, verkkomalli ja mallinnettu. Käytin samoja verkkosivuja testauksessa lähi- ja internetverkossa. Google Chrome selaimen Lighthouse-työkalun asetuksista valitsin pelkän tehokkuustestin (Performance). Valitsin joko työpöytä- tai mobiilivaihtoehdon sen mukaan millä laitteella halusin tuloksia mitata. Tarkastelin tuloksia myös performance-välilehdellä, johon pääsin painamalla alkuperäisen selvityksen aukaisevaa nappia (View Original Trace). Kirjasin testitulokset ylös langattomalla 4G-yhteydellä haja-asutusalueella ja 50 M:n kuitukaapeliyhteydellä kaupunkialueella. Tämän luvun seuraavissa alaluvuissa esittelen testeissä käyttämäni verkkosivut ja niissä näkyvät kolmiulotteiset kappaleet. Luvun lopussa tarkastelen asioita, jotka kannattaa huomioida mallien suunnittelussa.

4.2 Kolmiulotteinen mallintaminen Three.js-kirjaston piirtotyökaluilla

Ensiksi aloitin ohjelmoimaan autoa muistuttavaa kappaletta hyödyntäen Three.js-kirjastosta löytyviä ohjelmallisia työkaluja. Niiden avulla voi muodostaa erimuotoisia geometrisia kappaleita. Käytännössä kappale piirtyy matemaattisen funktion avulla, ja kirjastoon luodut apuvälineet helpottavat ohjelmointia. Ohjelmoijan ei tällöin tarvitse miettiä toteuttamista tarkemmin. Yksinkertaista leluautoa muistuttavaa kappaletta varten tarvitsen kolme funktiota. Pallon, laatikon ja sylinterin piirtäminen onnistuu SphereBufferGeometry-, BoxBufferGeometry- ja CylinderBufferGeometry-funktioilla (kuva 22).



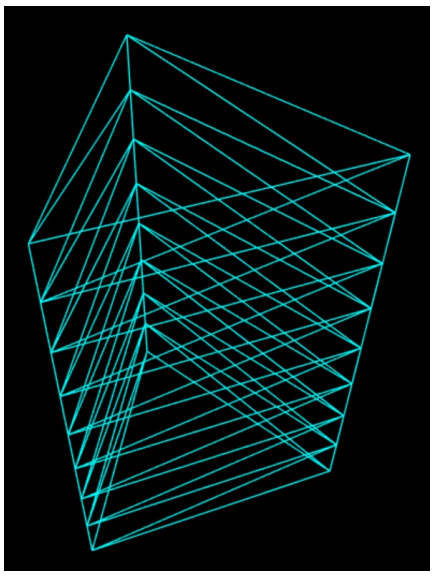
Kuva 22. Leluautolta näyttävä kolmiulotteinen kappale muodostuu useammasta kappaleesta

4.3 Verkkomalli kolmiulotteisen piirtämisen havainnollistamiseen

Aiemmassa luvussa hyödynsin apuna Three.js-kirjaston valmiita piirtofunktioita kolmiulotteisen mallin piirtämiseen ruudulle. Tämän rinnalle on hyvä ottaa kuitenkin vielä Three.js-kirjastolla toteutettu, mutta ohjelmallisesti hieman erilainen malli. Käyttämällä itse ohjelmoitua matemaattista funktiota, saadaan ruudulle piirtymään kolmiulotteisia muotoja.

Tämän avulla voidaan saada vertailukohtaa valmiiden funktioiden hyödyllisyydestä ja ohjelman nopeamman toiminnan suhteen. Kuvassa 23 on esitelty Three.js-kirjaston ja matemaattisen funktion avulla piirretty rautalankamalli, johon

voidaan halutessa lisätä pinta. Läpinäkyvyys havainnollistaa kolmiulotteista rakennetta.



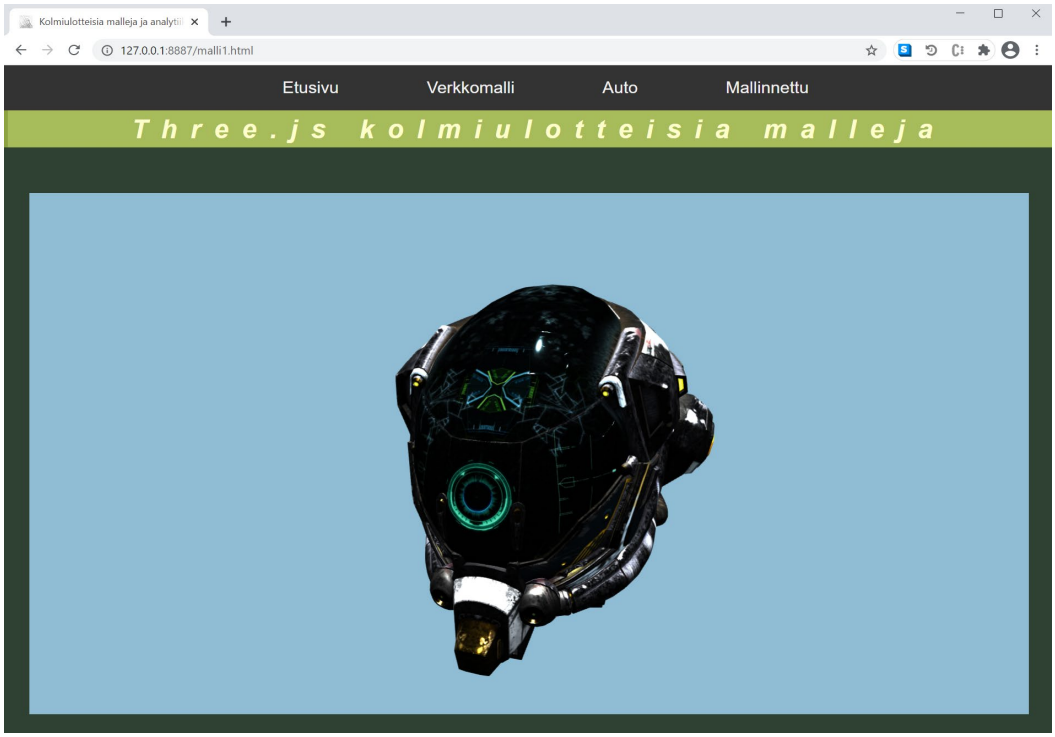
Kuva 23. Matemaattisella funktiolla piirretty rautalankamalli

4.4 Valmiin kolmiulotteisen mallin esittäminen verkkosivuilla

Seuraavaksi loin verkkosivun ja hyödynsin Three.js-kirjaston apuvälineitä 3D-mallinnusohjelmalla luodun valmiin mallin esittämiseen. Tehokkuusnäkökulmasta valmiin mallin testaaminen verkonnopeudessa on mielenkiintoinen vertailukohta ohjelmallisesti luotuihin. Kun kyseessä on verkkosivuilla avautuva 3D-grafiikka, verkon yli tapahtuva siirtymisnopeus on usein merkittävämpi pullonkaula kuin laitteiston nopeus. Tällöin on hyvä kiinnittää huomiota yksityiskohtaisemman grafiikan mukanaan tuomaan tiedoston kasvuun.

Kolmiulotteisia malleja hain Github-palvelusta, jossa on tarjolla erilaisia malleja Three.js-kirjastolla testaamiseen (GitHub 2020f). Malleja on eri tiedostomuodoissa, joista valitsin glTF-muodon. glTF (Graphics Language Transmission Format) tukee animaatioiden muodostamista ja kaikki tarvittava informaatio on tallennettu json-tiedostomuotoon (Khronos Group 2020b; JSON.ORG 2021). Jatkokäyttöä varten malli voidaan tallentaa glb-tiedostomuotoon, josta voidaan

hakea binääritiedon lisäksi animointiin ja rakenteeseen liittyvää json-tietoa. Kolmiulotteisten mallien tallennus- ja esittämismuoto glTF on muodostunut standardiksi 3D-assettien vaihdossa verkossa, ja sen on kehittänyt Khronos Group (Khronos Group 2020b). Kuvassa 24 on DamagedHelmet niminen kolmiulottenen malli, jonka on kehittänyt Leonardo Carrion ja sinä on Creative Commons Attribution-NonCommercial lisenssi (Creative Commons 2020).

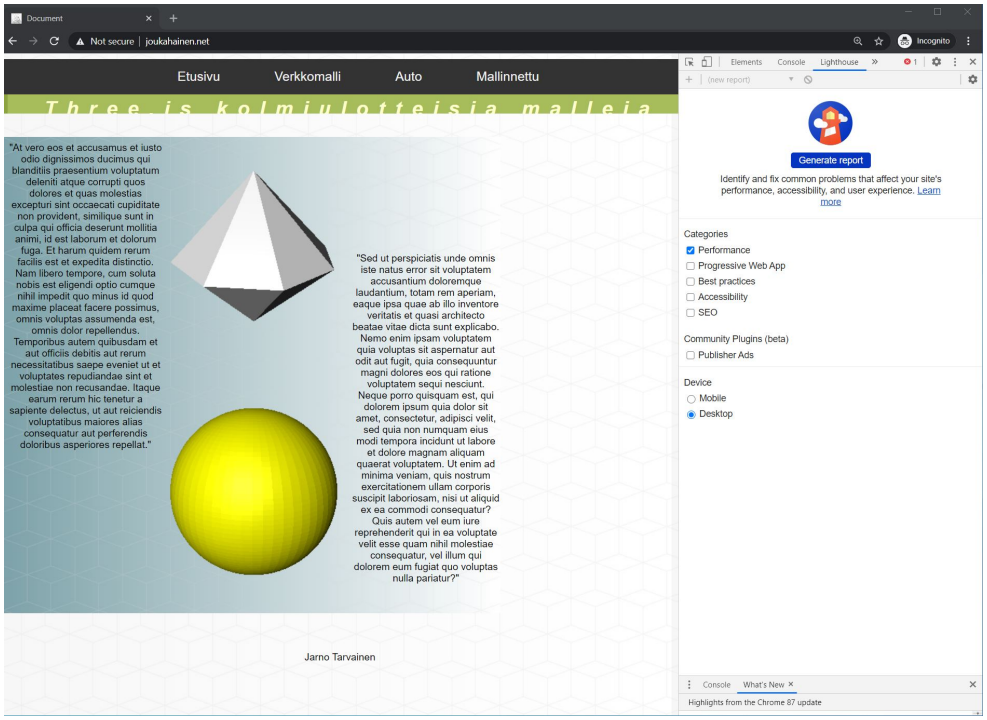


Kuva 24. Kolmiulotteinen malli tuotu verkkosivulle gbl-tiedostomuodon avulla

4.5 Kahden 3D-mallin esimerkki verkkosivuilla Three.js-kirjastolla

Aiempien 3D-esitysten jälkeen ohjelmoin kaksi kolmiulotteista kappaletta sisältävän verkkosivun, jonka tarkoituksena on testata Three.js-kirjaston avulla ohjelmoitujen kolmiulotteisten kappaleiden piirtymisen nopeutta verkkoselaimessa. Tästä syystä sivun tarkoitus on olla muulta rakenteeltaan yksinkertainen. Toteutuksessa kiinnitin huomiota kappaleiden asetteluun. Three.js-kirjaston avulla JavaScript-koodissa kirjoitetaan kappaleiden visuaalinen ilme. Ensin määritellään, mitkä kohdat koodissa ovat niitä, jotka saadaan kiinni HTML-dokumentissa. HTML-dokumentissa määritellään, mihin kohtiin kappaleet halutaan sijoittaa ja

millä tavalla. Lisäksi käytetään CSS-muotoiluja tehostamaan sijoittelua. Tässä tapauksessa 3D-elementit on sijoitettu HTML-dokumenttiin CSS Grid Layout tekniikalla (Mozilla 2021b). Tekniikka on yksi tapa sijoitella palikoita eri kohtiin verkkosivuilla. Tämä mahdollistaa sen, että kappaleet voidaan asetella myös visuaalisesti näyttävämmille verkkosivuille. Lisäksi toteutuksessa kiinnitin huomiota kappaleessa 3.3 esittelemääni tekniikkaan, jolloin ohjelma ei syö turhaan laitteistoresursseja, kun useampaa kolmiulotteista kappaletta esitetään verkkosivulla. Kuva 25. esittää kahta kolmiulotteista kappaletta, jotka on aseteltu verkkosivulle edellä mainittuja tekniikoita hyödyntäen. Lisäksi Kuvassa 25. näkyy oikealla Google Chrome selaimen Lighthouse-työkalu.



Kuva 25. Kaksi kolmiulotteista kappaletta samalla verkkosivulla

4.6 Mitä asioita pitää huomioida kolmiulotteisia malleja sisältävän verkkosivunsuunnittelussa

Kolmiulotteisia malleja sisältävän verkkosivun tehokkuus siis paranee nopeampina latausaikoina, kun verkkoselaimessa latautuvat tiedostot saadaan mahdollisimman pieneen kokoon ja ilman mitään ylimääräisiä toiminnallisuuksia. Lisäksi pitää huomioida aika, jolloin kolmiulotteinen malli latautuu taustalla, vaikka muu

verkkosivun ohjelmakoodi olisi jo latautunut. Grafiikan tarkkuus voidaan asettaa joko ennalta keskimääräiseksi, odotetun loppukäyttäjäkunnan mukaiseksi tai taustalla olevaa koodia voidaan kehittää mahdollistamaan grafiikan heikkeneminen ja tarkentuminen. Tällöin sisällöstä pitää olla erilaatuisia versioita, joka toisaalta vie enemmän tallennustilaa.

Tehokkuuden näkökulmasta voidaan kiinnittää huomiota mallien laatuun ja mahdolliseen haluttuun käyttäjäkuntaan. Näyttävyyden kannalta on tärkeää, että grafiikasta saadaan niin tarkkaa kuin verkon ja käytetyn päätelaitteen nopeus mahdollistaa. Kuitenkin on tärkeää, että sivuston yleinen toiminta ei vaikuta hitaalta. Tehokkuus ja verkkosivuston rakenne vaikuttavat näkyvyyteen hakukonetuloksissa. Mikäli sivustolla on tarvetta esittää enemmän kiintolevytilaa vievää yksityiskohtaisempaa grafiikkaa, tulee miettiä latautumisen kestoa ja siitä ilmoittamista verkkosivun käyttäjälle. Yksi mahdollisuus on rakentaa ilmoituspalkki, josta käyttäjä näkee latautumiseen kuluvan ajan. Tällöin voidaan ehkäistä turhautuneen käyttäjän poistuminen verkkosivustolta. Jatkossa esittelemäni 3D-mallinnusohjelmalla toteutetun mallin latautumisajat verkon yli havainnollistavat tarvetta latautumisen ilmoittamisesta käyttäjälle.

5 Tulokset

5.1 Tuloksista yleisesti

Tuloksissa käyn läpi verkkosivujen siirto- ja latausaikoja. Siirtoaika on se aika, joka kuluu tiedon siirtoon verkon yli. Latausaika (Onload Time) taas tarkoittaa tässä tapauksessa aikaa, kun lähes kaikki verkkosivun komponentit ovat latautuneet näkyväksi ruudulle (Mozilla 2021c). Latausaika koostuu suoritus- ja verkkosiirtoajasta. Tuloksia on mitattu Google Chrome -selaimen Lighthouse- ja performance-työkaluilla. Tulosten mittauksessa on käytetty kahta eri pöytämallista PC-tietokonetta Windows 10 -käyttöjärjestelmällä. Tuloksia on mitattu lähiverkossa ja verkkopalvelimen kautta. Testattavat verkkosivut on viety omalle palvelimelle verkossa ja tuloksia on testattu hitaammilla ja nopeammilla verkkoyhteyksillä.

Verkkoyhteyksinä internetiin olen käyttänyt langatonta 100 M:n 4G-verkkoyhteyttä ja 50 M:n langallista valokuituyhteyttä. 4G-yhteydellä mitatut tulokset on mitattu haja-asutusalueella.

Lisäksi verkkoyhteyden hidastamiseksi mittauksissa on käytetty Suomessa toimivaa virtuaaliverkkoa (VPN). VPN-verkko on salattu yhteys internetverkon välityksellä toiseen maantieteellisesti toisessa sijainnissa sijaitsevaan laiteeseen ja siitä internetiin. Selaimen työkalu muuttaa mittalaitteen virtuaaliseen mobiili- tai työpöytätietokonemalliin. Tämän avulla tulosvertailu on saatu toteutettua tavallisen tietokoneen ja mobiililaitteen välille.

Tuloksissa vertaillaan yhteenvetotiedoston sisältäviä (Bundle) verkkosivujen mittaustuloksia, jotka sijaitsevat kokonaan samalla palvelimella, CDN-toteutuksella tehtyihin, joissa osa ohjelmasta haetaan sisällönjakeluverkosta (Content Delivery Network). CDN-verkko on eri maantieteellisiin sijainteihin hajautettu verkko, joka muodostuu välityspalvelimina toimivista tietokoneista (CDNetworks 2021). Yhden verkkosivun mittauksessa on otettu kaksi otantaa, joista on kirjattu ylös lataus- ja siirtonopeudet. Ensimmäinen otanta on tehty työpöytä- ja toinen mobiililaitteella. Tuloksista voidaan erottaa lataus- ja siirtoaikojen keskiarvot Bundle ja CDN-toteutusten välillä. Bundle ja CDN-toteutuksien ero millisekunneissa lasketaan kaavalla:

$$e = \left| \frac{1}{n} \sum_{i=1}^{n} x_i = (x_1 + \cdots x_n) - \frac{1}{n} \sum_{i=1}^{n} y_i = (y_1 + \cdots y_n) \right|$$

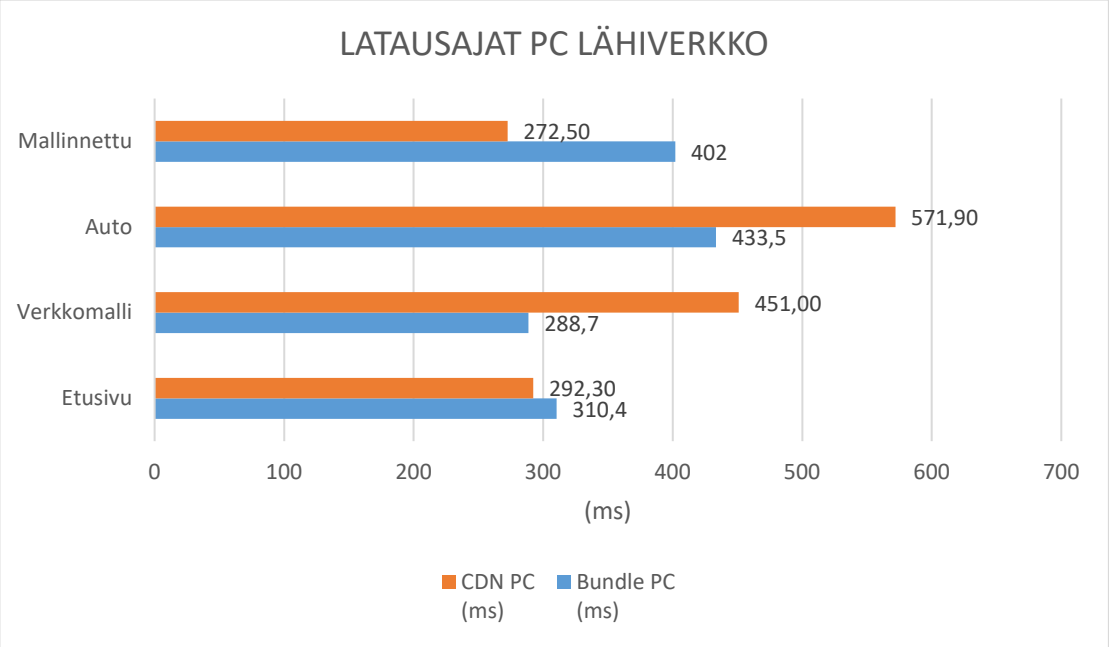
Kaavassa tulos e on keskimääräinen ero mobiili- ja työpöytäkonemittausten sekä Bundle- ja CDN-toteutusten välillä. Muuttuja x on Bundle-toteutuksen latausaika ja muuttuja y on CDN-toteutuksen latausaika. Molempien aritmeettisten keskiarvojen erotuksesta otetaan itseisarvo.

Siirtoajat verkon yli voidaan laskea samalla kaavalla. Lisäksi siirtoajat ovat summia eri verkkosivun komponenttien siirtoajoista. Tämän jälkeen voidaan laskea erotuksien aritmeettiset keskiarvot. Latausajoissa keskiarvoksi saadaan 367 millisekuntia ja siirtoaikojen keskiarvoksi saadaan 364 millisekuntia.

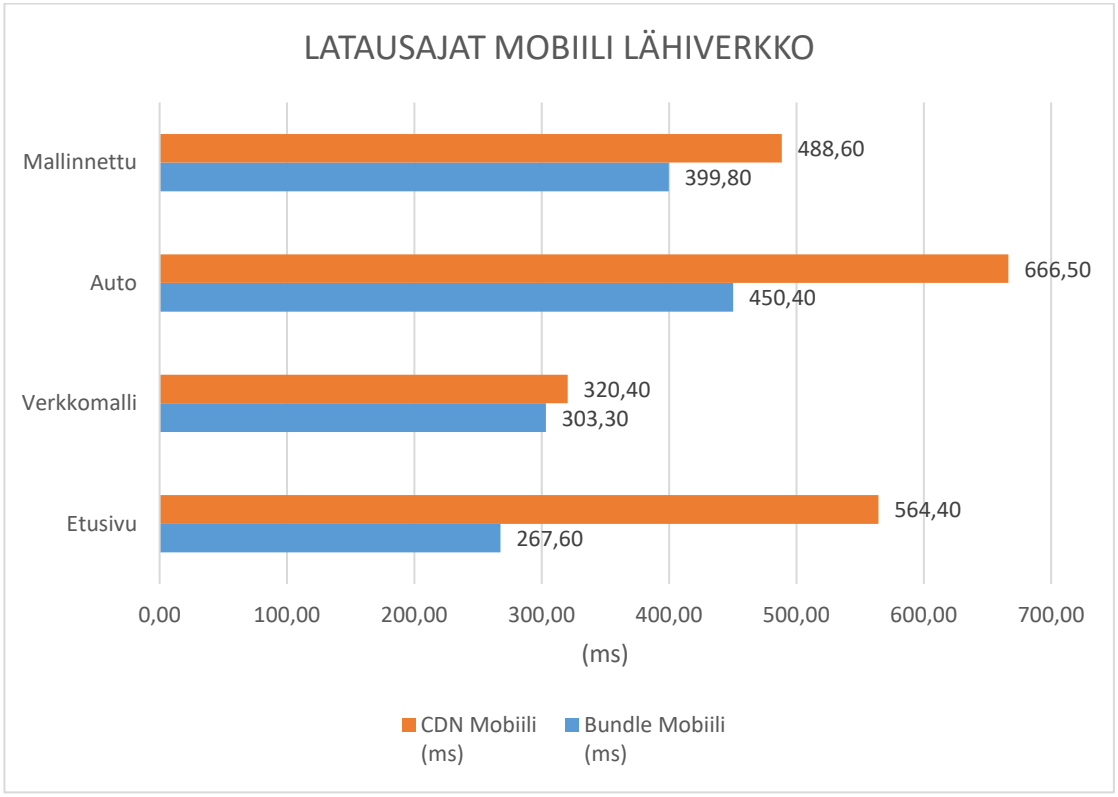
5.2 Mittaustulokset lähiverkossa

Lähiverkon mittaustulokset osoittavat, että tiivistettyyn muotoon (Bundle) tehdyt verkkosivut latautuvat nopeammin samalta palvelimelta, kun maantieteellinen sijainti on suunnilleen yhtä lähellä CDN-verkkopalvelimiin verrattuna.

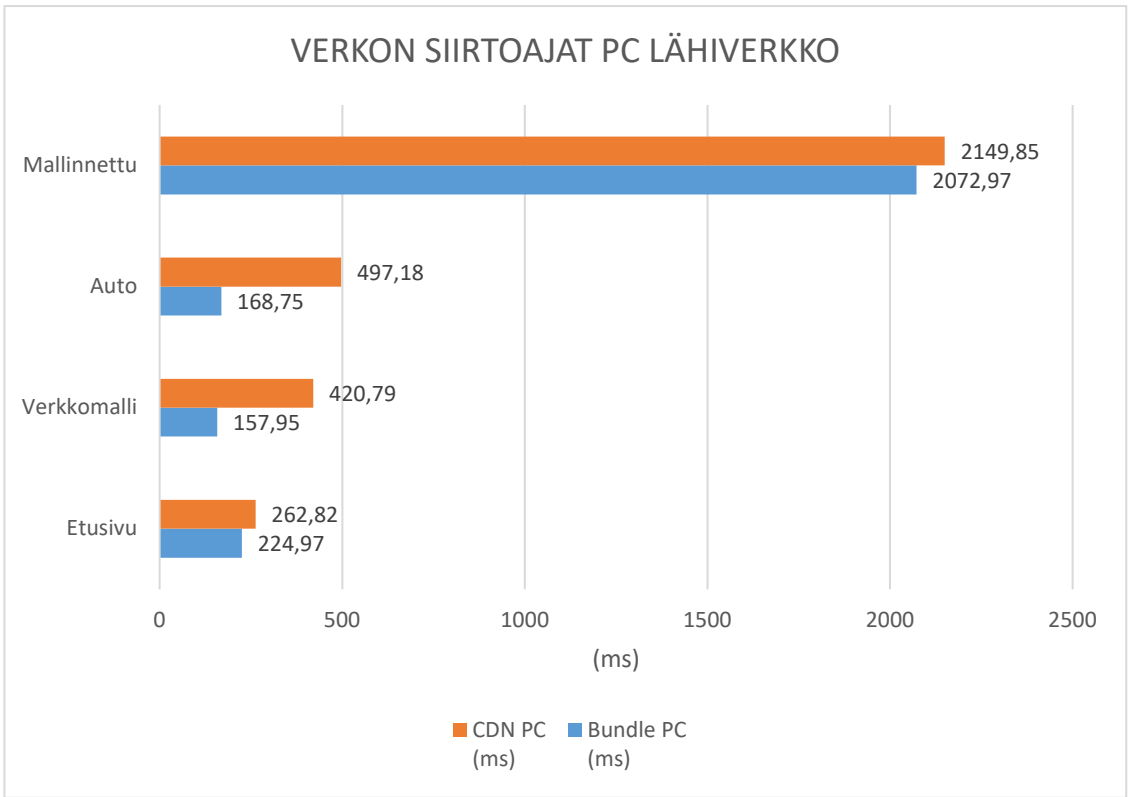
Tulokset näkyvät poikkeuksetta nopeampina siirto- ja latausaikoina työpöytä- ja mobiililaitteella. Kuviot 1. ja 2. havainnollistavat latausaikoja ja Kuviot 3. ja 4. siirtoaikoja verkon yli.



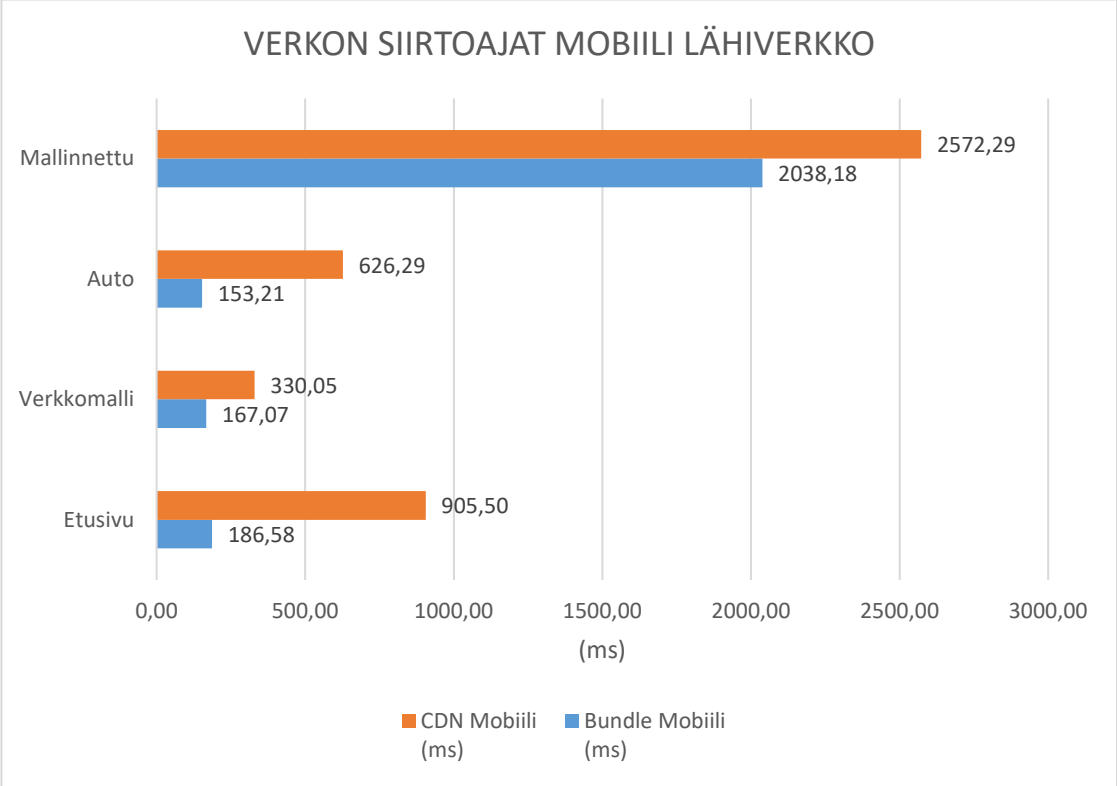
Kuvio 1. Lähiverkon latausajat työpöytätietokoneella



Kuvio 2. Lähiverkon latausajat puhelimella



Kuvio 3. Lähiverkon siirtoajat työpöytätietokoneella

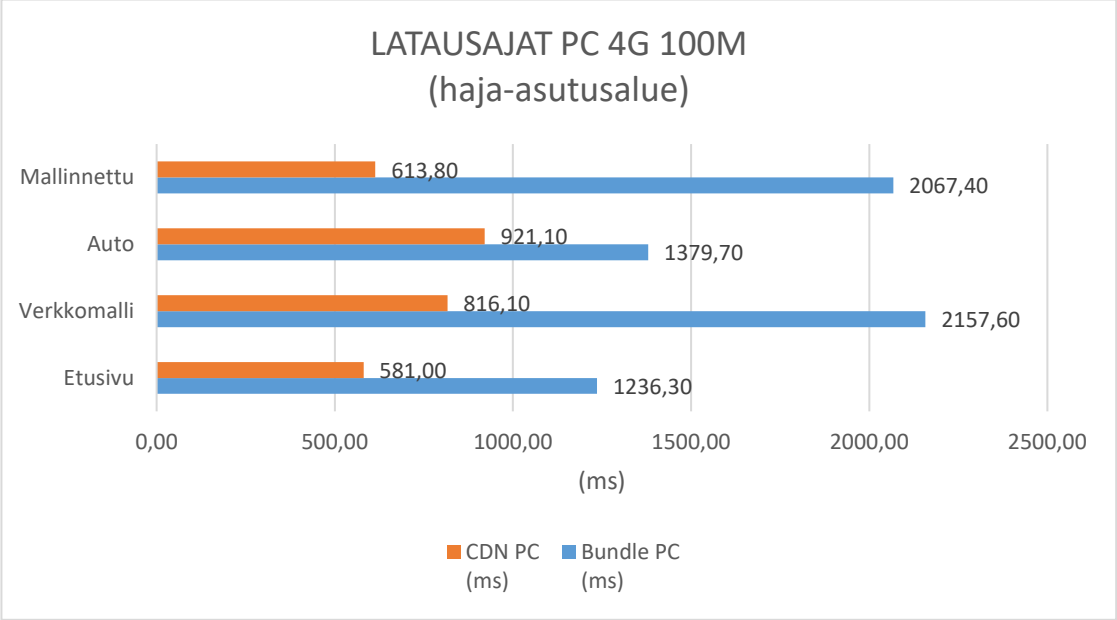


Kuvio 4. Lähiverkon siirtoajat puhelimella

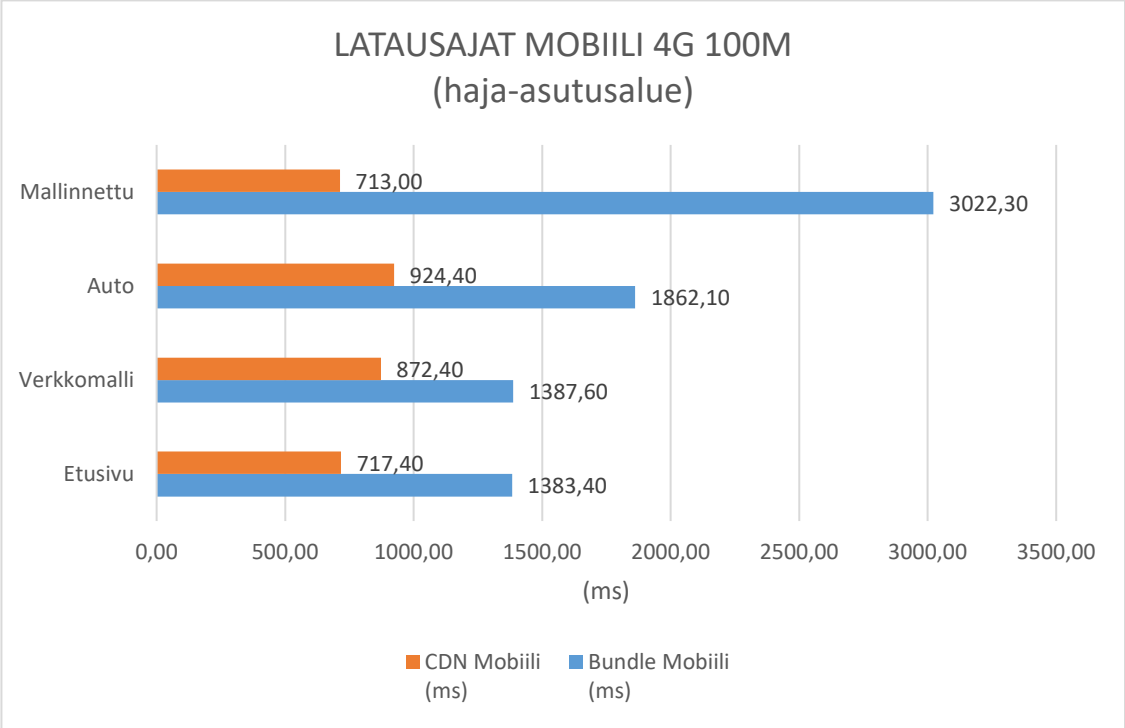
5.3 Latausaikojen mittaustulokset palvelinkoneelta 100M 4G internet-yhteydellä haja-asutusalueella

Kuvioissa 5, 6, 7 ja 8 näkyy langattomalla 4G-yhteydellä mitatut latausajat. Kasi viimeistä Kuviota esittävät mittaustulokset hidastetulla VPN-yhteydellä. Latausajoista voidaan huomata maantieteellisen sijainnin etu CDN-verkkoja käytettäessä.

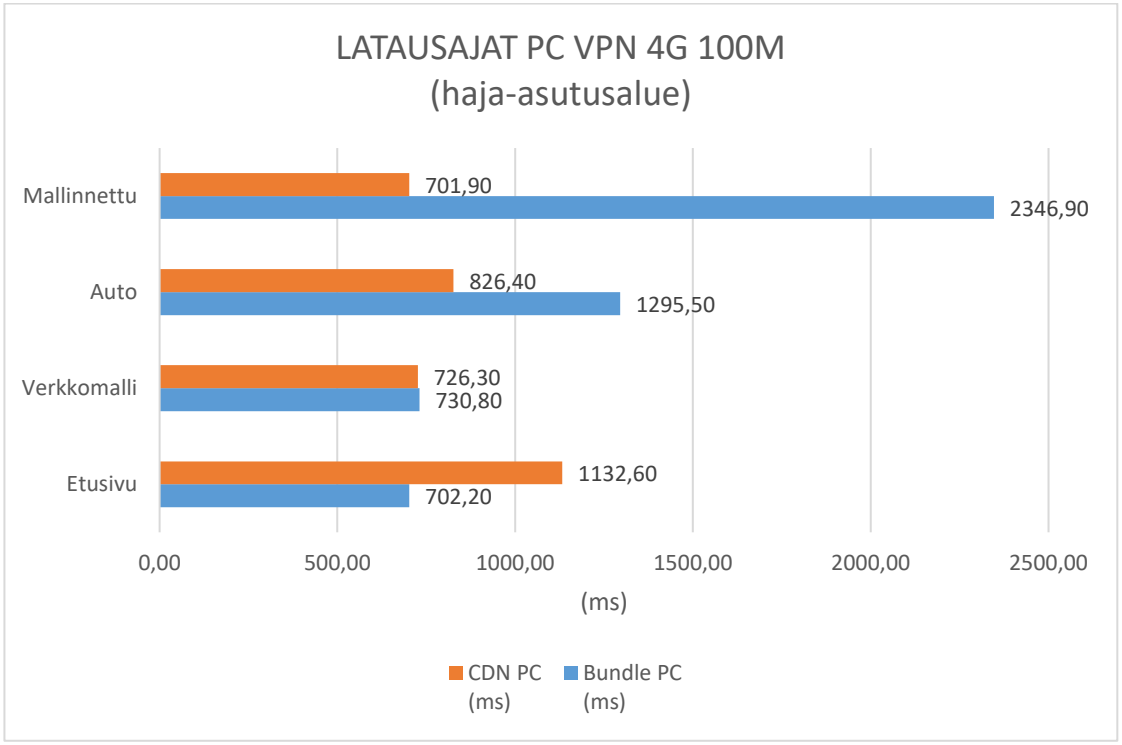
Verrattuna Bundle-toteutukseen, jossa verkkosivukomponenttien siirron maantieteellinen sijainti pysyy samana. Kuviossa 7. voidaan huomata kuitenkin tasoittumista latausajoissa verkkosivujen eri toteutusten välillä.



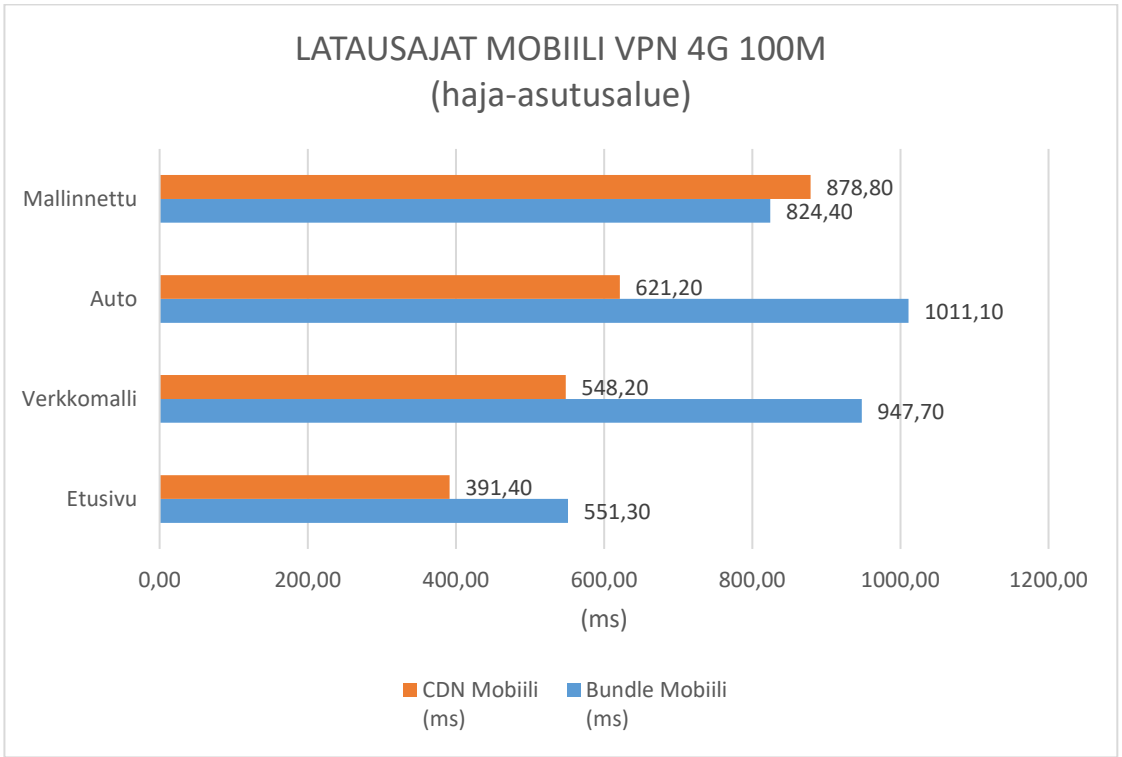
Kuvio 5. Verkon latausajat palvelimelta työpöytätietokoneelle 100M 4G-yhteydellä



Kuvio 6. Verkon latausajat palvelimelta puhelimelle 100M 4G-yhteydellä



Kuvio 7. Verkon latausajat virtuaaliverkon kautta palvelimelta työpöytätietokoneelle 100M 4G-yhteydellä

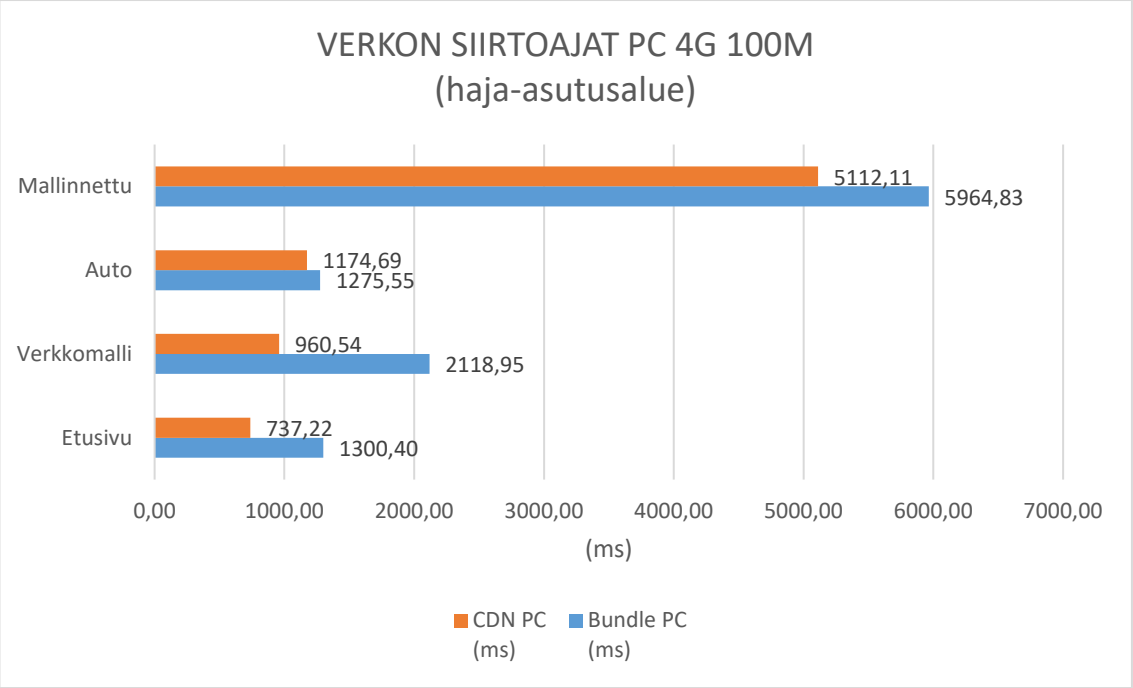


Kuvio 8. Verkon latausajat virtuaaliverkon kautta palvelimelta puhelimeen 100M 4G-yhteydellä

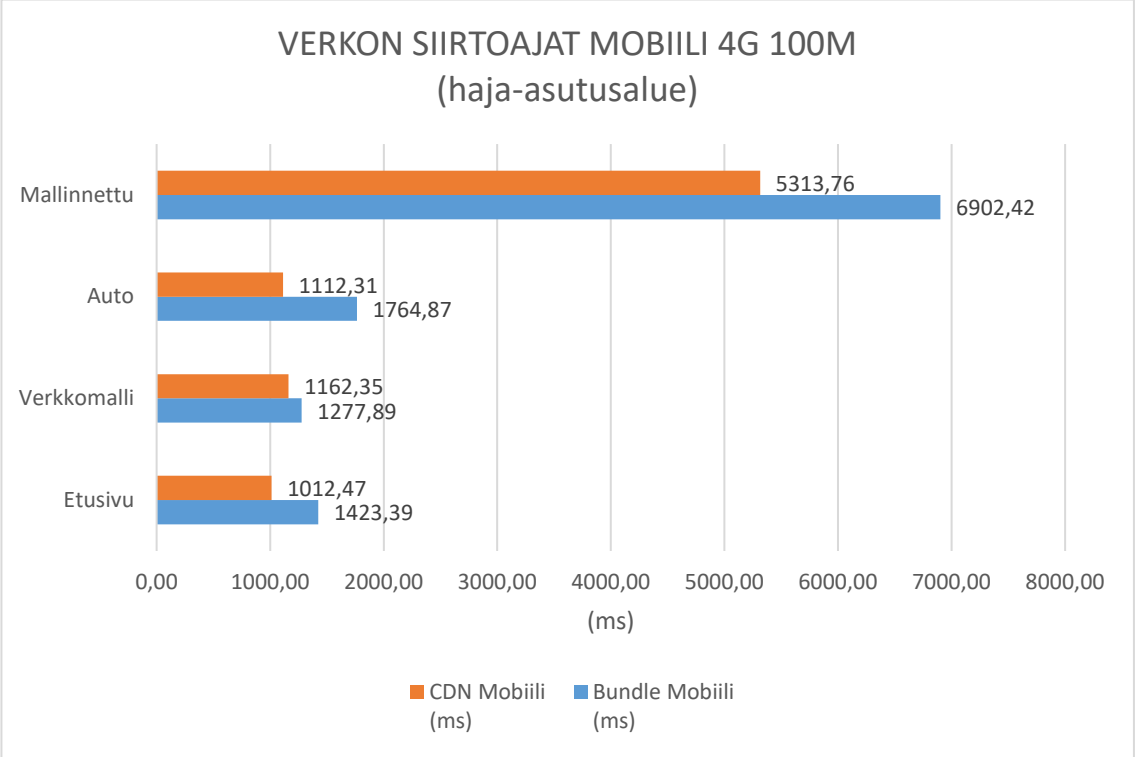
5.4 Siirtoaikojen mittaustulokset palvelinkoneelta 100M 4G internet-yhteydellä haja-asutusalueella

Kuvioissa 9, 10, 11, ja 12 esitetyissä 4G-yhteydellä mitatuissa siirtoajoissa voidaan edelleen havaita CDN-verkon tuomaa etua maantieteellisesti. Kuitenkin siirtoaikojen mittaustuloksista huomataan eron tasoittuvan enemmän.

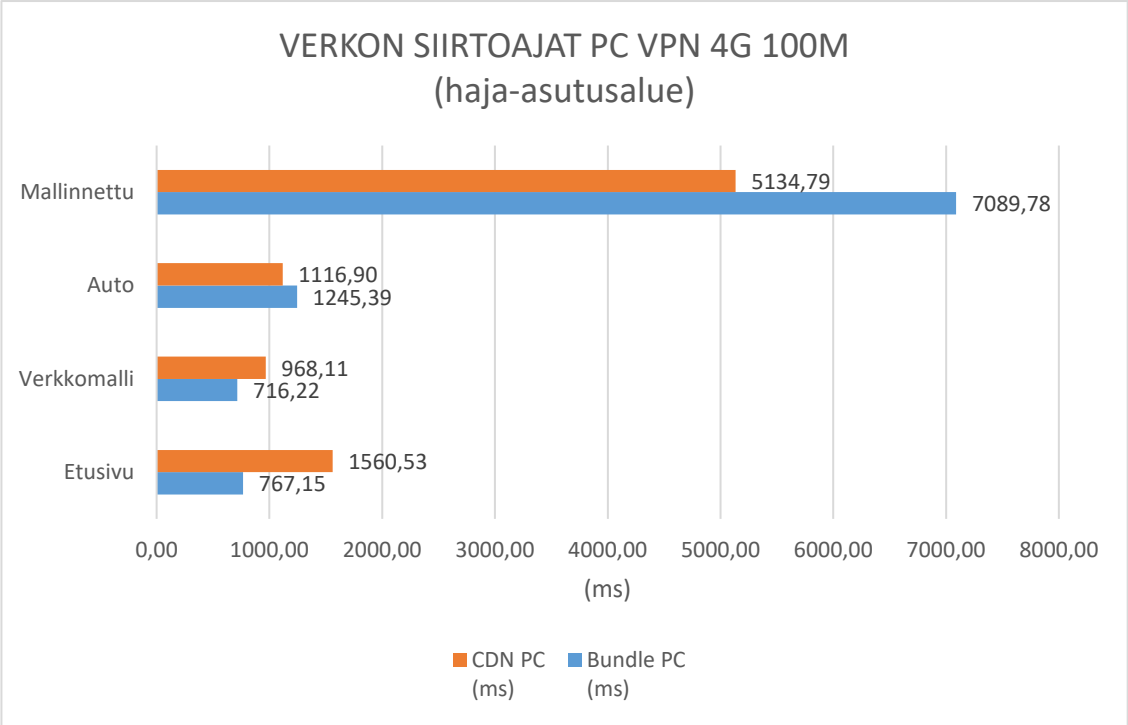
Kuvioissa 11. ja 12. huomataan samalla palvelimella olevan tiivistetyn (Bundle) toteutuksen olevan välillä nopeampi CDN-toteutukseen verrattuna. CDN-toteutus vaikuttaisi olevan parempi ratkaisu 4G-yhteydellä, kun verkkosivuston palvelinkone ei sijaitse riittävän lähellä maantieteellisesti.



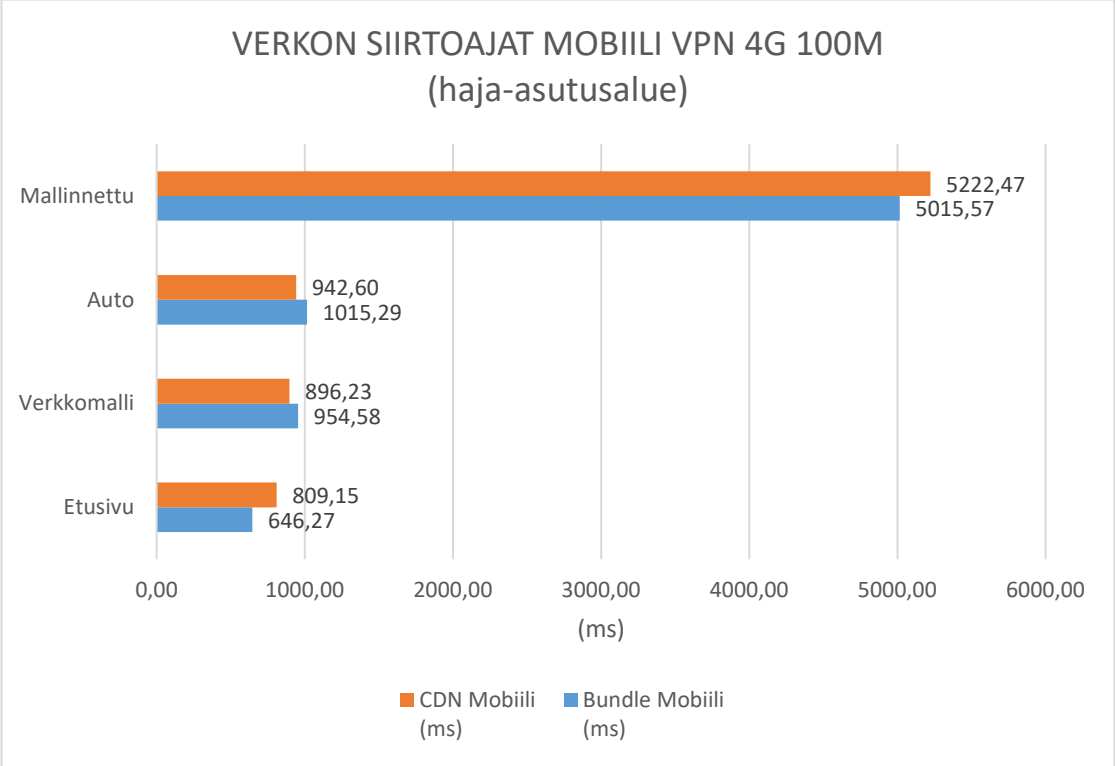
Kuvio 9. Verkon siirtoajat palvelimelta työpöytätietokoneelle 100M 4G-yhteydellä



Kuvio 10. Verkon siirtoajat palvelimelta puhelimeen 100M 4G-yhteydellä



Kuvio 11. Verkon siirtoajat virtuaaliverkon kautta palvelimelta työpöytätietokoneelle 100M 4G-yhteydellä

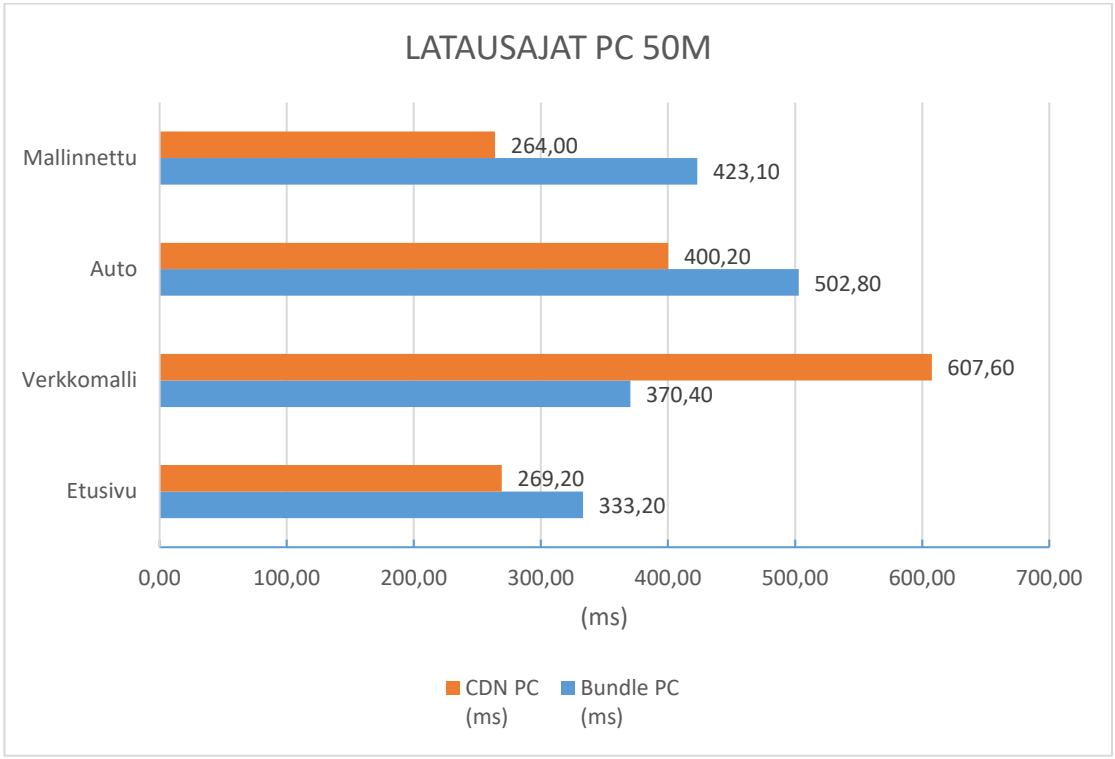


Kuvio 12. Verkon siirtoajat virtuaaliverkon kautta palvelimelta puhelimeen 100M 4G-yhteydellä

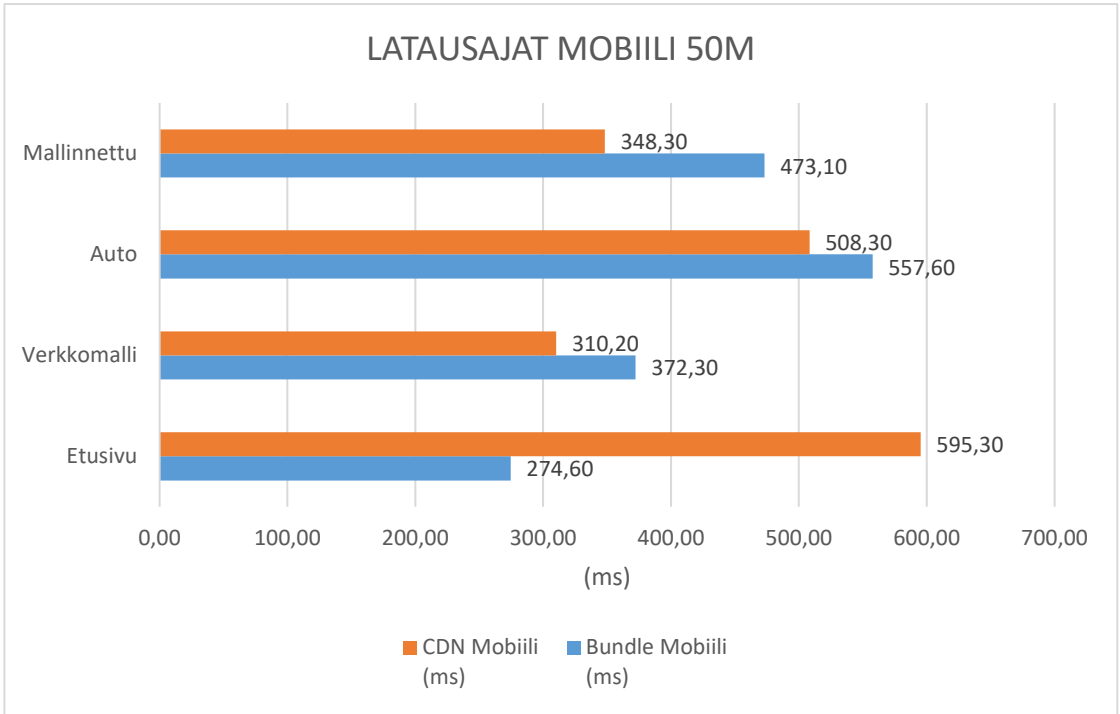
5.5 Latausaikojen mittaustulokset palvelinkoneelta 50M internet-yhteydellä

Kuvioissa 13, 14, 15 ja 16 on esitelty verkon yli mitattuja latausaikoja 50M valokuituyhteydellä. Kaksi viimeistä kuviota esittävät mittaustulokset VPN-yhteydellä hidastettuna.

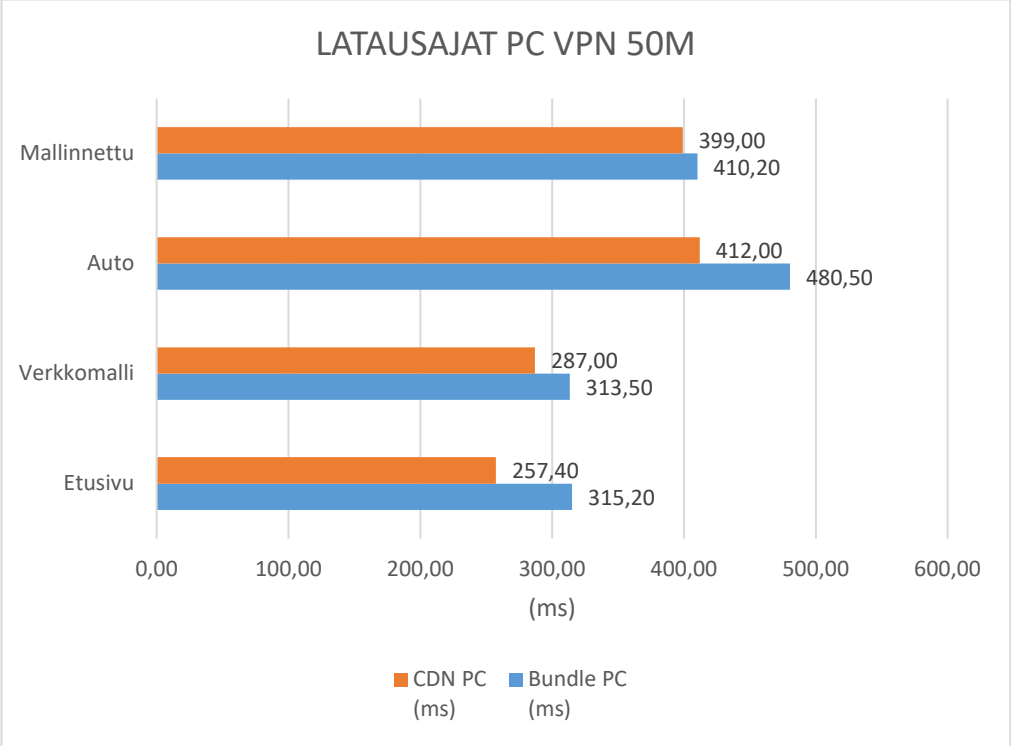
Tuloksista voidaan havaita jonkin verran CDN-verkon tuomaa etua maantieteellisen sijainnin kannalta. Kuitenkin Kuvioissa 13., 14., ja 16. yksi tulos neljästä on Bundle-toteutuksella nopeampi kuin CDN-verkkoa käyttävällä.



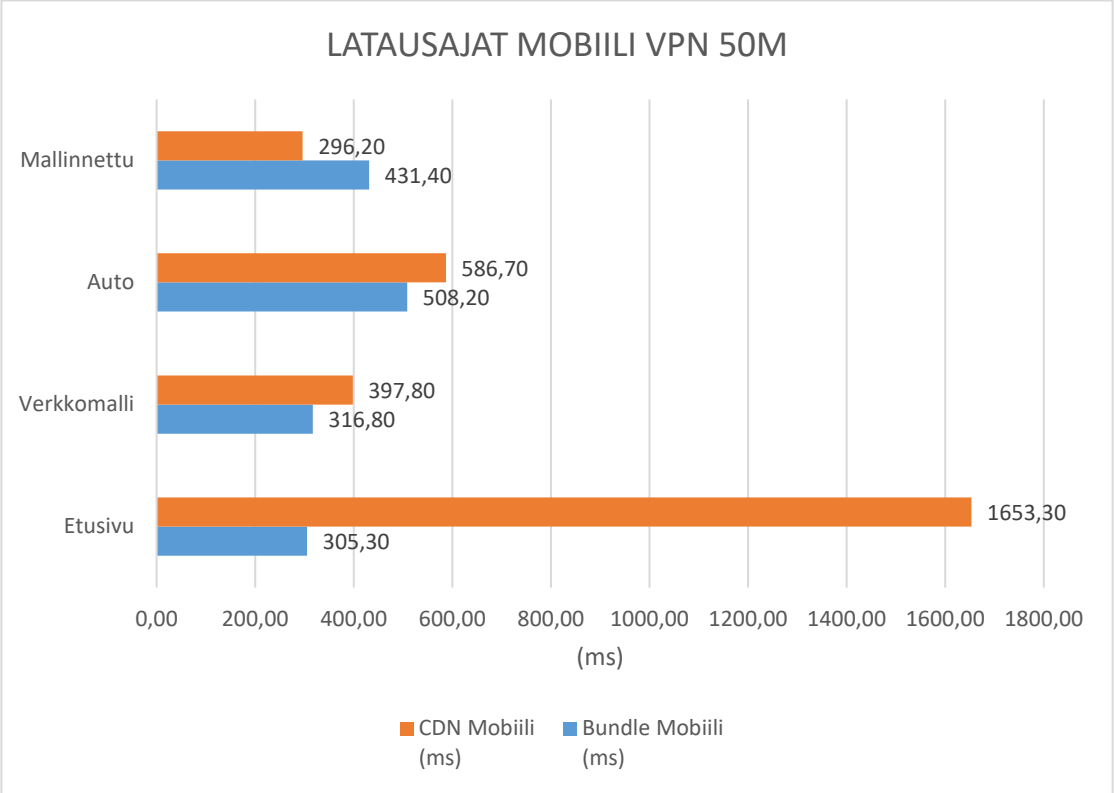
Kuvio 13. Verkon latausajat palvelimelta työpöytätietokoneelle 50M-yhteydellä



Kuvio 14. Verkon latausajat palvelimelta puhelimeen 50M-yhteydellä



Kuvio 15. Verkon latausajat virtuaaliverkon kautta palvelimelta työpöytätietokoneelle 50M-yhteydellä

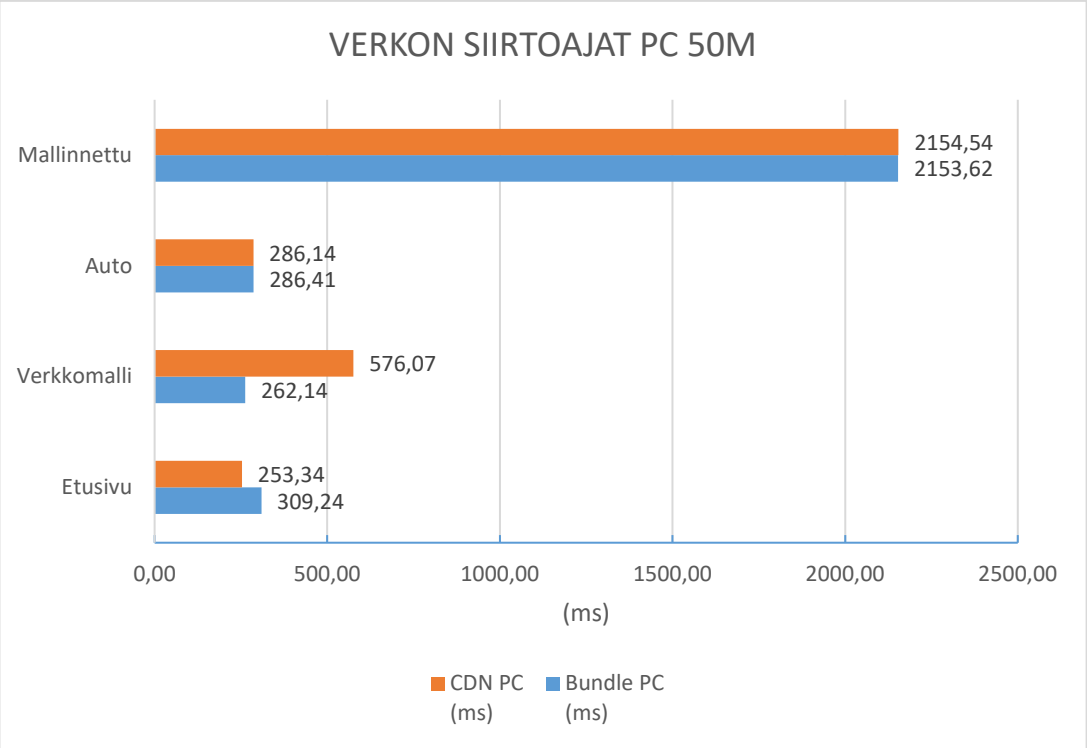


Kuvio 16. Verkon latausajat virtuaaliverkon kautta palvelimelta puhelimeen 50M-yhteydellä

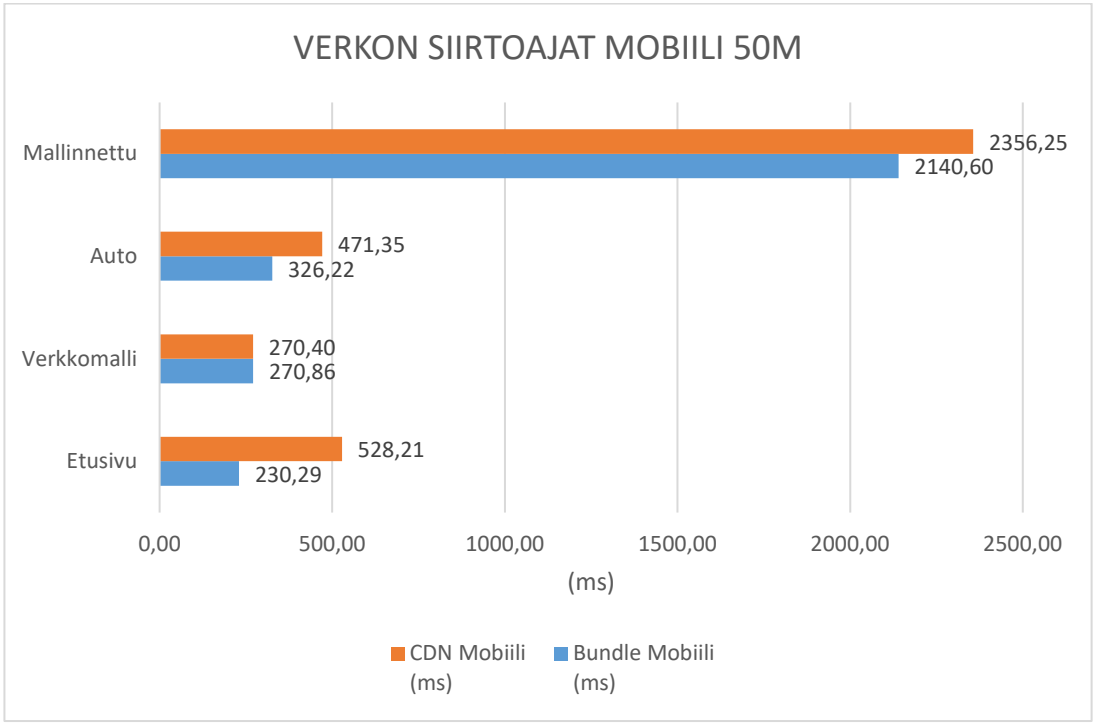
5.6 Siirtoaikojen mittaustulokset palvelinkoneelta 50M internet-yhteydellä

Kuivioissa 17, 18, 19 ja 20 esitettyjen 50M valokuituyhteydellä mitattujen verkon siirtoaikojen tuloksista voidaan huomata Bundle-toteutuksen olevan nopeampi CDN-verkkoihin nähden. Tällöin voidaan todeta stabiilin nopeamman verkkoyhteyden edun siihen, että pienet maantieteelliset sijaintierot eivät hidasta siirtoaikoja CDN-verkkoihin verrattuna.

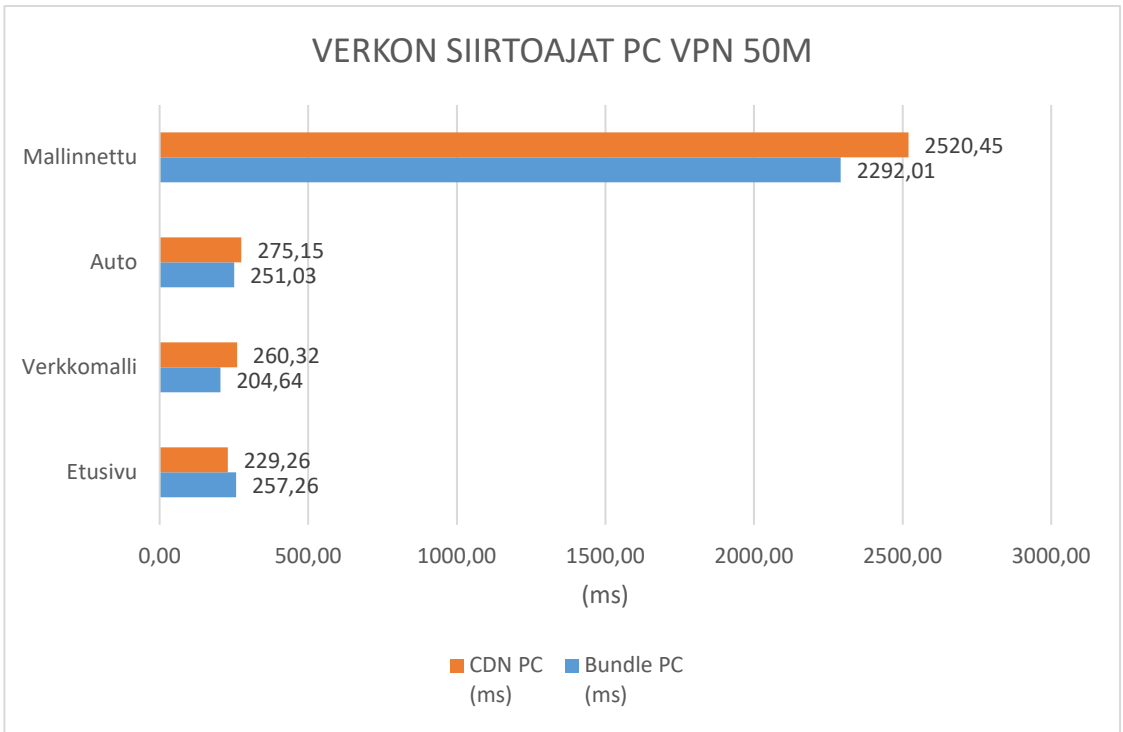
Verkon siirtonopeuksista voidaan havaita, että 3D-ohjelmalla mallinnettu (Mallinnettu) verkkosivutoteutus vaatii vielä varsin suuria latausaikoja. Verkkosivukehittäjän tulisi harkita latautumispalkin sijoittamista verkkosivulle, joka kertoo käyttäjälle kuinka pitkään kolmiulotteisten mallien lataus kestää. Ainoastaan 50M valokuituyhteys antaa jo kohtuullisen nopean siirtoajan, jolloin latautumispalkin pois jättämistä voisi harkita.



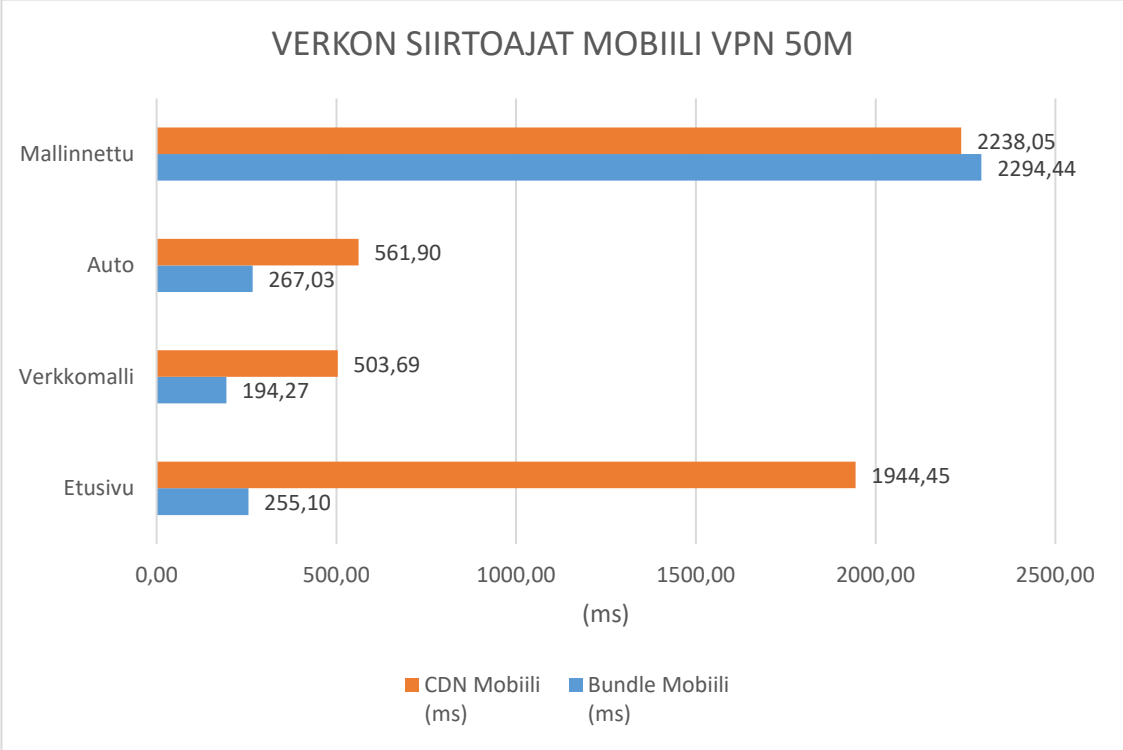
Kuvio 17. Verkon siirtoajat palvelimelta työpöytätietokoneelle 50M-yhteydellä



Kuvio 18. Verkon siirtoajat palvelimelta puhelimeen 50M-yhteydellä



Kuvio 19. Verkon siirtoajat virtuaaliverkon kautta palvelimelta työpöytätietokoneelle 50M-yhteydellä



Kuvio 20. Verkon siirtoajat virtuaaliverkon kautta palvelimelta puhelimeen 50M-yhteydellä

6 Pohdinta

Esittelemäni mittaustulokset osoittavat, että on parasta valita keskimääräisen tarkka grafiikka, joka toimii riittävän nopeasti useammilla markkinoilla olevilla laitteilla ja verkkoyhteyksillä. Mikäli grafiikka muuttuu yksityiskohtaisemmaksi ja se alkaa viemään useista megatavuista gigatavuihin tilaa, verkkosivujen käyttäjälle on parasta ilmoittaa latautumisen kestosta. Verkkosivutoteutuksessa voi olla kuitenkin mukana nopeammin ja hitaammin latautuvia kolmiulotteisia komponentteja mielenkiinnon säilyttämiseksi. Lopuksi mittaustulosten perusteella voidaan todeta vielä, että varsin raskaan Three.js-kirjaston tai vastaavan käyttöä tulisi harkita silloin, kun kolmiulotteisen grafiikan esittämiseen tarvitaan niissä jo valmiiksi kehitettyä koodia. Muutoin hyvin yksinkertaista kolmiulotteisuutta voi rakentaa JavaScriptillä suoraan. Tällöin voidaan saada vielä lisäetua siirto- ja latautumisnopeuksiin. Kolmiulotteisuuden lisääminen verkkosivuille voi siis sopivissa määrin parantaa verkkosivujen visuaalista ilmettä ilman, että sivujen latautumisnopeus

kärsii liikaa. Nopea latautuminen ja optimaalinen toteutus parantaa taas näkyvyyttä hakukonetuloksissa, joka omalta osaltaan vaikuttaa kävijöiden määrän kasvuun. Parempi visuaalinen ilme voi tuoda enemmän käyttäjiä ja niitä, jotka palaavat sivustolle, koska niitä on miellyttävä käyttää.

Mielestäni opinnäytetyön aihe oli itselleni haastava, koska kolmiulotteisten mallien lisäämisestä verkkosivuille ei ollut aiemmin kokemusta. Aihe oli mielenkiintoinen ja siksi kannusti perehtymään asiaan laajemmin. Työn edetessä aihealue laajeni hieman ja asiasisältö muuttui rikkaammaksi. Vaikka muut työtehtävät venyttivät aikataulua, kokonaisuudessaan työ onnistui mielestäni hyvin.

Verkkoyhteydet ja käytettävät laitteet nopeutuvat koko ajan. Kun 50 M:n valokuituyhteyksiä nopeammat internet-yhteydet ovat yleistyneet kaikkialla. Yhä yksityiskohtaisempaa kolmiulotteista grafiikkaa voidaan lisätä verkkosivuille parantamaan kävijän visuaalista kokemusta.

Mittaustuloksia voidaan hakea niitä mittaavista verkkopalveluista, joista tutkimukseen voidaan ottaa mukaan myös muita verkkosivuntoimintaan liittyviä asioita. Kuten FLT-arvo (Full Load Time), joka saadaan, kun on varmistettu, että sivun kaikki verkon yli siirtyvät komponentit ovat latautuneet. Mittaustuloksia esittelevistä verkkopalveluista esimerkkinä on GTmetrix (GTmetrix 2021). Verkkosivujen toteutusta voidaan edelleen parantaa ja yksi esimerkki tästä on välimuistien hyödyntäminen nopeampien siirto- ja latausaikojen saavuttamiseksi. Lisää aiheesta on Akamai-yhtiön artikkelissa, joka esittelee välimuisteja verkossa ja niiden käyttöä (Akamai 2021).

Lähteet

A-Frame. 2020. https://aframe.io/docs/1.0.0/introduction. 14.11.2020.

Adobe. 2020. Comp CC. https://www.adobe.com/products/comp.html. 18.10.2020.

Akamai. 2021. What in Web Caching? https://www.akamai.com/uk/en/resources/web-caching.jsp. 15.2.2021.

Ambiera. 2020. CopperLicht. https://www.ambiera.com/copperlicht. 14.11.2020.

Artec 3D. 2019. Portable 3D Scanners. https://www.artec3d.com/portable-3d-scanners. 5.4.2019.

AutoDesk. 2020a. AutoDesk Online Gallery. https://gallery.autodesk.com. 20.10.2020.

AutoDesk. 2020b. AutoCAD. https://www.autodesk.com/products/autocad/overview. 20.10.2020.

AutoDesk. 2020c. 3Ds Max. https://www.autodesk.com/products/3ds-max/overview. 20.10.2020.

Away Studios. 2020. Away3D. http://away3d.com. 14.11.2020.

GTmetrix. 2021. https://gtmetrix.com. 8.2.2021.

Babel. 2020. What is Babel?. https://babeljs.io/docs/en. 14.11.2020.

Babylon.js. 2020. https://www.babylonjs.com. 14.11.2020.

Blender. 2019. Sculpting. https://www.blender.org/features/modeling. 30.4.2019.

Browserify. 2020. http://browserify.org. 31.10.2020.

Cocos. 2020. https://www.cocos.com/en/about. 14.11.2020.

Cormen, T.H., Leiserson C.E. & Stein, C. 2009. Introduction to Algorithms 3nd Ed. Cambridge, Massachusetts, London, England. The MIT Press.

CDNetworks. 2021. What is CDN. https://www.cdnetworks.com/what-is-a-cdn/. 23.01.2021.

Cabello, R. 2015. NVScene 2015 Session: Reinventing The Wheel - One Last Time (Ricardo Cabello). https://www.youtube.com/embed/LXWYOF4VibE. 20.10.2020.

Creative Commons. 2020. Creative Commons licenses. https://creativecommons.org/licenses/. 18.10.2020.

Discoverthreejs.com. 2020. Your First three.js Scene: Hello, Cube! 1.2 The Components of a Real-Time 3D App. https://discover-threejs.com/book/first-steps/first-scene. 20.10.2020.

Ecma International. 2020.Presentation on ECMAScript. https://www.ecma-international.org/activities/Golden%20jubilee/cc-2011-014.ppt. 14.11.2020.

Feldman, S. I. 1978. Make – A Program for Maintaining Computer Programs.

GNU Operating System. 2020. What is Copyleft?. https://www.gnu.org/licenses/copyleft.html. 12.10.2020.

Git. 2020. https://git-scm.com. 14.10.2020.

GitHub. 2020a. First public version. Still a lot to do. https://github.com/mrdoob/three.js/commit/a90c4e107ff6e3b148458c96965e876f9441b147. 14.10.2020.

GitHub. 2020b. White Paper for Three.js?. https://github.com/mrdoob/three.js/issues/1960. 14.10.2020.

GitHub. 2020c. mrdoob/three.js. https://github.com/mrdoob/three.js. 14.10.2020.

GitHub. 2020d. BabylonJS/Babylon.js. https://github.com/BabylonJS/Babylon.js. 14.10.2020.
GitHub. 2020e. LayaAir. https://github.com/layabox/LayaAir. 14.11.2020.
GitHub. 2020f. mrdoob/three.js. Damaged Helmet. https://github.com/mrdoob/three.js/tree/dev/examples/models/gltf/DamagedHelmet. 14.10.2020.
Google. 2020. Lighthouse. https://developers.google.com/web/tools/lighthouse/. 14.10.2020.
IBM. 2020. Application Programming Interface. https://www.ibm.com/cloud/learn/api. 31.10.2020.
InkScape. 2020. https://inkscape.org. 18.10.2020.
Itch corp. 2020. Top Games made with Three.js. https://itch.io/games/made-with-threejs. 14.10.2020.
Shabiralyani, G., Hasan, K.S., Hamad, N. & Iqbal N. 2015. Impact of Visual Aids in Enhancing the Learning Process Case Research: District Dera Ghazi Khan. Journal of Education and Practice. 6 (19), 228-233.
JSON.ORG. 2021. Introducing JSON. https://www.json.org/json-en.html. 23.1.2021.
Khronos Group. 2020a. WebGL Overview. https://www.khronos.org/webgl. 19.10.2020.
Khronos Group. 2020b. gITF Runtime 3D asset delivery. https://www.khronos.org/gitf. 14.11.2020.
Maxon. 2020. Cinema 4D. https://www.maxon.net/en/products/cinema-4d/overview. 20.10.2020.
Mozilla. 2021a. MDN Web Docs. Canvas API. https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API. 26.4.2021.
Mozilla. 2021b. MDN Web Docs. CSS Grid Layout. https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Grid_Layout. 23.1.2021.
Mozilla. 2021c. MDN Web Docs. Window: load event. https://developer.mozilla.org/en-US/docs/Web/API/Window/load_event. 23.1.2021.
Naker. 2021. Why 3D is the future of the internet?. https: //www.naker.io/posts/why-3d-is-the-future-of-the-internet. 18.04.2021.
Node.js. 2020. About Node.js. https://nodejs.org/en/about. 31.10.2020.
NPM. 2020. https://www.npmjs.com/about. 20.10.2020.
OpenSceneGraph. 2020. http://www.openscenegraph.org. 14.11.2020.
PixiJS. 2020. The HTML5 Creation Engine. https://www.pixijs.com. 14.11.2020.
Scotch. 2020. JavaScript Transpilers: What They Are & Why We Need Them. https://scotch.io/tutorials/javascript-transpilers-what-they-are-why-we-need-them. 14.11.2020.
Smithsonian. 2020. Smithsonian X 3D. http://legacy.3d.si.edu/explorer. 20.10.2020.
Stallman, R. M. 2015. Free Software, Free Society, Selected Essays of Richard M. Stallman, 3rd ed. Boston MA, USA: Free Software Foundation.
The Linux Information Project. 2020. Markup Language Definition. http://www.linfo.org/markup_language.html. 14.11.2020.
The XFree86 Project. 2020. X Consortium. https://xfree86.org/3.3.6/COPYRIGHT2.html#3. 18.10.2020.

Three.js. 2020a. https://threejs.org. 18.10.2020.
Three.js. 2020b. https://threejs.org/docs/#api/en/core/BufferGeometry. 18.10.2020.
Three.js Fundamentals. 2020. https://threejsfundamentals.org/threejs/lessons/threejs-multiple-scenes.html. 20.10.2020.
Trimble. 2020a. 3D Warehouse. https://3dwarehouse.sketchup.com. 18.10.2020.
Trimble. 2020b. Sketchup. https://www.sketchup.com/. 18.10.2020.
TurboSquid. 2020. 3D Models for Professionals. https://www.turbosquid.com. 18.10.2020.
Quality Nonsense Ltd. 2020. SEO Success Guide: Understand Why Google Is King & How You Can Master SEO Fast. https://html.com/seo. 19.10.2020.
Wings 3D. 2020. A Polygon Modeler. http://www.wings3d.com. 18.10.2020.
Unity Technologies. 2020. Unity Asset Store. https://assetstore.unity.com. 18.10.2020.
W3Schools. 2020. HTML Tags Ordered Alphabetically. https://www.w3schools.com/TAGs/. 30.10.2020.
Webpack. 2020a. https://webpack.js.org. 20.10.2020.
Webpack. 2020b. HtmlWebpackPlugin. https://webpack.js.org/plugins/html-webpack-plugin. 14.11.2020.
WebPlatform.org. 2020. Code minification. https://webplatform.github.io/docs/concepts/programming/javascript/minification. 31.10.2020.
Wikipedia. 2019a. Game physics. https://en.wikipedia.org/wiki/Game_physics. 30.4.2019.
Wikipedia. 2019b. Wire-frame model. https://en.wikipedia.org/wiki/Wire-frame_model. 6.5.2019.
Wikipedia. 2019c. Rendering (computer graphics). https://en.wikipedia.org/wiki/Rendering_(computer_graphics). 31.10.2019.
Wikipedia. 2019d. Polygonimalli. https://fi.wikipedia.org/wiki/Polygonimalli. 20.10.2019.
Wikipedia. 2019e. 3D computer graphics. https://en.wikipedia.org/wiki/3D_computer_graphics. 5.4.2019.
Wikipedia. 2019f. Vapaa ohjelmisto. https://fi.wikipedia.org/wiki/Vapaa_ohjelmisto. 6.5.2019.
Wikipedia. 2019g. WWW-sisällönhallinta. https://fi.wikipedia.org/wiki/WWW-sisällönhallinta. 20.10.2019.
Wikipedia. 2019h. Package manager. https://en.wikipedia.org/wiki/Package_manager. 31.10.2019.
WordPress. 2020. https://wordpress.org. 31.10.2020.