
Pietari Järvi

# 3D-mallinnus verkkosivuilla

Metropolia Ammattikorkeakoulu
Insinööri (AMK)
Tieto- ja viestintätekniikka
Insinöörityö
31.1.2023

# Tiivistelmä


| | |
|---|---|
| Tekijä: | Pietari Järvi |
| Otsikko: | 3D-mallinnus verkkosivuilla |
| Sivumäärä: | 35 sivua |
| Aika: | 31.1.2023 |
| | |
| Tutkinto: | Insinööri (AMK) |
| Tutkinto-ohjelma: | Tieto- ja viestintätekniikka |
| Ammatillinen pääaine: | Ohjelmistotuotanto |
| Ohjaajat: | Lehtori Toni Spännäri |


---


Insinöörityön tarkoituksena oli tutkia 3D-mallinnusta verkkosivuilla. Tutkittavia asioita oli mallinnuksen käyttötarkoitukset, kuinka malleja luodaan ja mitä teknologioita käytetään. Tämän lisäksi tavoitteena oli luoda oma 3D-malli ja tuoda se omalle sivustolle. Projektin avulla voitaisiin arvioida valittujen menetelmien ja teknologioiden haasteellisuutta ja sopivuutta kehittäjille, joilla ei ole aikaisempaa kokemusta 3D-mallinnuksesta verkkosivuille.

Insinöörityö toteutettiin jakamalla käyttötarkoitukset tarkempiin osa-alueisiin verkkosivuilla esiintyvässä 3D-mallinnuksen käytössä. Alueiksi valittiin tuote-esittely, visuaalisuuden lisääminen ja viimeisenä taideprojektit ja pelillisyys. Menetelmät-osiossa keskityttiin ensin 3D-mallinnukseen yleisesti ja tutkittiin tarkemmin tietokonegrafiikan hahmontamista OpenGL- ja WebGL-teknologioilla. Menetelmät-osioon kuului viimeisenä myös kolmen 3D-mallinnusta verkkosivuilla helpottavan grafiikkakehyksen tutkiminen.

Tutkittujen asioiden perusteella mallinnettiin 3D-malli donitsista Blender-mallinnustyökalussa. Mallille luotiin verkkosivusto käyttämällä Three.js JavaScript -kirjastoa ja malli tuotiin sivustolle tiedostosta. Mallille toteutettiin myös yksinkertainen käyttöliittymä sen värien muokkaamiseksi. Mallinnuksessa ja sivuston luomisessa oli omat haasteensa. Geometriasolmuilla luodut strösselit eivät ensin näkyneet, kun malli tuotiin sivustolle. Myös ohjelmointiongelmat hidastivat prosessia.

Insinöörityö kuitenkin osoitti, että valitut työkalut ja teknologiat ovat hyvä valinta 3D-mallinnukseen ja sen toteuttamiseen verkkosivuille. 3D-mallinnukseen ja grafiikkakehyksiin löytyy kattavasti tietoa ja tukea, joten tyylikkään 3D-mallin luominen ja yksinkertainen 3D-mallinnusta käyttävä sivusto onnistuu myös aloittelijalta. Samalla kuitenkin todettiin, että tietyt mallintamisen tekniikat, kuten geometriasolmut, vaativat laajempaa tietämystä mallinnustyökalusta.



Avainsanat: 3D-mallinnus, kolmiulotteisuus, tietokonegrafiikka

# Abstract


| | |
|---|---|
| Author: | Pietari Järvi |
| Title: | 3D Modeling on Websites |
| Number of Pages: | 35 pages |
| Date: | 31 January 2023 |
| Degree: | Bachelor of Engineering |
| Degree Programme: | Information and Communications Technology |
| Professional Major: | Software Development |
| Supervisors: | Toni Spännäri, Senior Lecturer |


---


The purpose of this final year project was to research 3D modeling on websites. Areas to be researched were uses of 3D modeling, how 3D models are made and what technologies are used. In addition, the goal was to create a 3D model and export it to a website. The project could be used to assess the challenges and suitability of the selected methods and technologies for developers with no previous experience in 3D modeling for websites.

The project began with dividing the purposes of use of 3D modeling on websites into more specific sub-areas. The areas chosen were product presentation, increasing visuality and lastly art projects and games. In the methods, the focus was first on 3D modeling in general and the rendering of computer graphics with OpenGL and WebGL technologies was studied in more detail. Finally, the methods also included the study of three graphics frameworks used in 3D modeling on websites.

Based on the research, a 3D model of a donut was modeled in the modeling tool Blender. A website was created for the model using Three.js JavaScript library and the model was imported to the site from a file. A simple user interface for modifying its colors was also implemented. Modeling and creating the site had its own challenges. The sprinkles created with geometry nodes were not visible at first when the model was imported to the site. Programming problems also slowed down the process.

However, the study showed that the selected tools and technologies are a good choice for 3D modeling and its implementation on websites. There is comprehensive information and support for 3D modeling and graphics engines, so creating a stylish 3D model and a simple website using 3D modeling can be done even by a beginner. At the same time, however, it was noted that certain modeling techniques, such as geometry nodes require more extensive knowledge of the modeling tool.



Keywords: 3D modeling, three-dimensionality, computer graphics

# Sisällys

## Lyhenteet

CAD: *Computer-aided design*. Tietokoneavusteinen suunnittelu. Tietokoneen käyttö apuvälineenä suunnittelussa.

B-rep: *Boundary representation*. Rajaesitysmallinnus. Esineiden esittäminen tarkasti tilavuuksina.

NURBS: *Non-uniform rational basis spline*. 3D-mallinnuksen tapa, jossa käytetään polygonien sijaan käyriä.

OpenGL: *Open Graphics Library*. Ohjelmointirajapinta 2D- ja 3D-grafiikan hahmontamiseen eli renderöintiin.

WebGL: *Web Graphics Library.* JavaScript-ohjelmointirajapinta 2D- ja 3D-grafiikan hahmontamiseen eli renderöintiin verkkosivuilla.

GLSL: *OpenGL Shading Language*. Varjostinkieli, jonka avulla voidaan luoda visuaalisia tehosteita.

glTF: *Graphics Language Transmission Format*. 3D-mallien tiedostomuoto.

.glb: *glTF Binary*. 3D-mallien tiedostomuoto, jossa on samassa tiedostossa mukana kaikki tarvittavat tiedot, kuten polygoniverkko ja pintakuviointi.

# 1 Johdanto

Insinöörityössä tutkitaan kolmiulotteista mallinnusta eli 3D-mallinnusta verkkosivuilla. Kolmiulotteisella tarkoitetaan, että kuvatulla tai näytettävällä esineellä on kolme ulottuvuutta: leveys, korkeus ja syvyys. Kolmiulotteinen mallinnuksen hyödyntäminen mahdollistaa uusia keinoja olla luova web-kehityksessä, ja 3D-malleja tai kolmiulotteisuuden illuusiota käyttävät verkkosivut erottuvat joukosta.

Henkilökohtaisten tietokoneiden suorituskyvyn sekä ohjelmistojen kehittymisen ansiosta 3D-mallinnuksen mahdollisuudet ovat kasvaneet. Nykyään kuka vain, joka omistaa tietokoneen ja verkkoyhteyden, pystyy luomaan 3D-mallin ja esittämään sen omalla verkkosivullaan. 3D-mallinnusta käytetään laajasti sekä eri aloilla että yksityiseen käyttöön, ja mallit voivat olla mitä vain matkapuhelimista peleihin ja taideprojekteihin.

Työn tavoitteena on tutkia 3D-mallinnuksen käyttötarkoituksia ja menetelmiä sekä luoda oma 3D-malli mallinnustyökalussa ja esittää se verkkosivulla. Tavoitteena on myös tutkia, miten projektissa käytetyt teknologiat sopivat tehtävään sekä niiden haastavuutta ilman aiempaa kokemusta.

Työssä tutkitaan käyttötarkoituksien lisäksi 3D-mallien luontia ja tietokonegrafiikan hahmontamista tarkemmin kolmen grafiikkakehyksen sekä OpenGL- ja WebGL-ohjelmointirajapintojen kautta. Verkkosivu luodaan käyttäen HTML-, CSS- ja JavaScriptin lisäksi 3D-mallinnukseen verkossa käytettävää Three.js JavaScript -kirjastoa. Sivulla esitettävä 3D-malli mallinnetaan Blender-mallinnustyökalussa.

# 2 Käyttötarkoitukset

3D-mallinnuksessa luodaan ohjelmiston avulla kolmiulotteinen malli objektista. Ensimmäisen tietokoneavusteisen suunnittelun ohjelma eli CAD-ohjelma (engl. Computer-aided Design) kehitettiin 1960-luvun alussa. Ensimmäiset CAD-mallit

tehtiin kaksiulotteisesti piirtämällä fyysinen kappale sinikopion kaltaiseksi kuvaksi. [1, s. 4.]

Kolmiulotteinen mallinnus alkoi yleistymään vasta myöhemmin. Tässä suuressa osassa oli vuonna 1974 ensimmäisten B-rep-mallintajien kehittäminen. Rajaesitysmallinnus tai B-rep-mallinnus, jonka nimitys tulee englannin kielen vastineesta Boundary Representation, esittää esineet tarkasti tilavuuksina. Tästä syystä B-rep-malleja käytetään yleisimmin konetekniikassa ja tuotteiden suunnittelussa. [1, s. 5; 2.]

Nykyään käyttötarkoituksia 3D-malleille on kuitenkin useita. Esimerkiksi elokuva- ja TV-teollisuudessa sekä videopeleissä 3D-malleja käytetään hahmojen, esineiden ja ympäristön luomiseen. Lääketieteessä 3D-malleja käytetään esimerkiksi visualisoimaan anatomiaa [3]. Esittelen tässä luvussa hieman tarkemmin verkkosivuilla esiintyviä 3D-mallinnuksen keinoja.

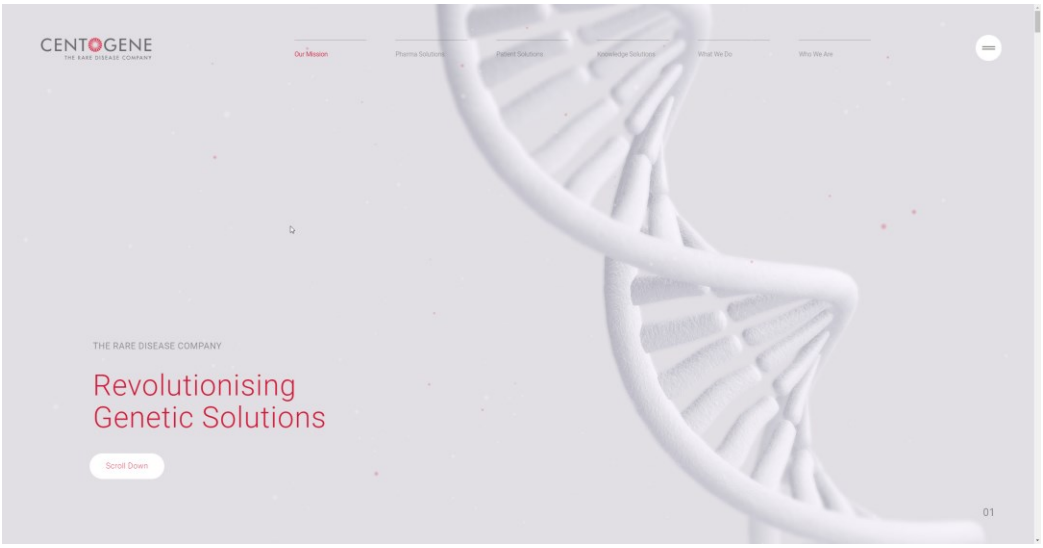
## 2.1 Tuote-esittely

3D-mallit voivat olla hyvä tapa esitellä tuotetta. Pelkän valokuvauksen sijaan tietokoneella luotua mallia voidaan manipuloida reaaliaikaisesti. Koska malli on kolmiulotteinen, verkkosivun käyttäjä pystyy kääntelemään ja tutkimaan tuotetta jokaisesta kulmasta. Malli voi olla animoitu toteuttamaan tuotteen toimintaa, tai mallin tuote voidaan purkaa näyttämään käyttäjälle tuotteen rakennetta. 3D-malleihin voidaan myös tehdä muokkauksia helposti jälkikäteen.

3D-mallinnus voi olla joissain tapauksissa halvempaa kuin kuvaustiimin palkkaaminen ja aikataulutus tiettyyn paikkaan tiettynä aikana [4]. Usein 3D-malli on kuitenkin käytössä yhdessä valokuvauksen kanssa. Tämä saattaa johtua siitä, että kuluttajat myös haluavat nähdä oikean tuotteen tietokoneella luodun mallin lisäksi. Tuote-esittelyä 3D-malleilla käytetään laajasti usealla eri alalla, ja tuotteet voivat olla mitä vain matkapuhelimista asuntonäyttelyihin.

## 2.2 Visuaalisuuden lisääminen

3D-mallinnuksella voidaan lisätä verkkosivulle visuaalisuutta ja erottua joukosta. Sivustolle saatetaan lisätä jokin aiheeseen liittyvä malli, joka ei kuitenkaan välttämättä ole mainostettava tuote. Kuvassa 1 geneettisiä sairauksia tutkiva Centogene on lisännyt verkkosivullensa DNA-juosteen.




Kuva 1. Centogenen sivustolla visuaalisuutta on lisätty lääketieteellisillä 3D-malleilla [5]

Toinen menetelmä lisätä sivustolle visuaalisuutta ja antaa sille syvyyttä on ottaa sivuston taustalla oleva elementti ja erottaa se muusta taustasta liikuttamalla elementtejä eri nopeudella. Elementti voi liikkua automaattisesti, käyttäjän selatessa sivua tai esimerkiksi hiiren kursorin mukana. Kaksiulotteinen kuva voidaan siis jakaa useaan kerrokseen ja näin luodaan kolmiulotteisuuden illuusio. Tätä kutsutaan parallaksi vieritykseksi (engl. parallax scrolling). Kuvassa 2 Moooi-huonekalu-, sisustus- ja valaistussuunnitteluyrityksen sivusto, jossa käytetään parallax-efektiä. [6.]


Kuva 2. Esimerkki parallax-efektistä suunnitteluyrityksen verkkosivuilla [7]

Sivuston etuosassa olevat kukat ja eläimet liikkuvat hitaammin kuin taustan elementit. Efektin aikaa saatetaan säätää vain hetkeksi lyhyellä animaatiolla, kun verkkosivu avataan. Efekti voi myös olla ensin heikompi ja aktivoitua täysin vasta käyttäjän toimesta. Tästä esimerkkejä ovat sivustolla sijaitsevan napin klikkaus tai hiiren rullaus.

## 2.3 Taideprojektit ja pelillisyys

3D-mallinnus soveltuu hyvin erilaisiin taideprojekteihin. Erotan taideprojektit visuaalisuuden lisäämisestä siten, että taideprojekteissa 3D-mallinnus on suuremmassa osassa sivuston ideaa ja tarkoitusta. Nämä voivat olla henkilökohtaisia projekteja, mutta eri järjestöt voivat myös ottaa kantaa yhteiskunnallisiin ongelmiin ja saada vetovoimaa ainutlaatuisuudella. Kuvassa 3 on Norjan

sademetsärahaston sivusto, jossa esitellään sademetsän luontoa, eläimiä ja asukkaita interaktiivisella kartalla.


Kuva 3. Interaktiivinen kartta Norjan sademetsärahaston sivustolla [8]

Interaktiivinen sademetsä on esimerkki siitä, että pelillisyys voi olla pienempänä osana sivustoa tuomassa interaktiivisuutta. Verkkosivustolle renderöidään 3D-malli alueesta, jota käyttäjä voi tutkia klikkaamalla eri kohteista. Pelillisyys voi myös olla koko sivuston idea. Pelejä on useita, mutta yksi esimerkki on Minecraft Classic, joka on selainversio suositusta hiekkalaatikkomaisesta pelistä.

Suurien pelien laittaminen selaimeen on vielä haastavaa. Massiivinen monen pelaajan verkkoroolipeli RuneScape on ennen toiminut selaimessa käyttäen Java-liitännäistä. Suosituimmat selaimet ovat kuitenkin lopettaneet selaimessa suoritettavien Java-ohjelmien eli Java-sovelmien (engl. Java Applet) tukemisen. Tästä syystä Javasta haluttiin eroon ja pelin uusimmalle versiolle Runescape 3 luotiin pelisovellus käyttäen HTML5-merkintäkieltä ja WebGL-ohjelmointirajapintaa. Tätä versiota ei koskaan julkaistu suoritusongelmien takia. Nykyään RuneScapea voi pelata vain työpöytäsovelluksella. [9; 10; 11.]

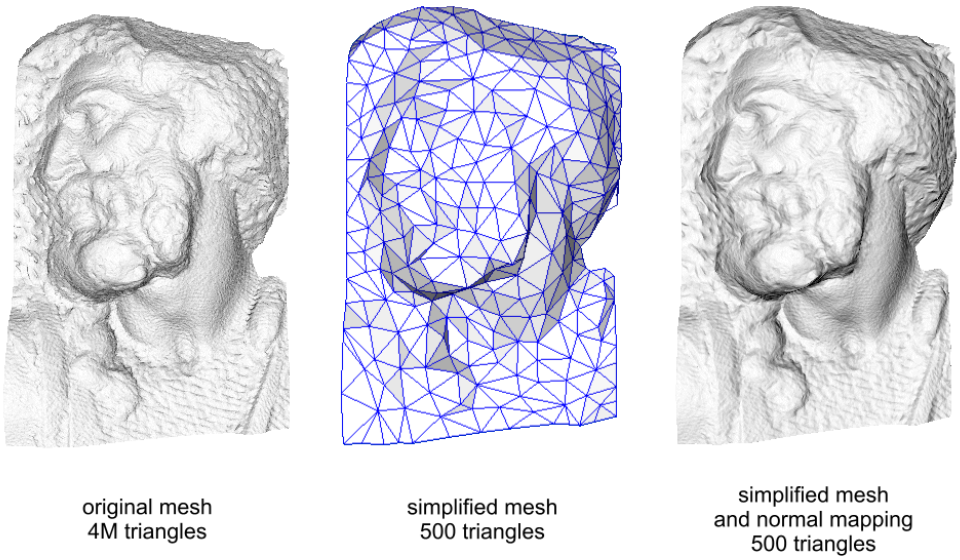
# 3 Menetelmät

## 3.1 3D-mallien luonti

3D-mallien luontiin on useita ohjelmia. Näihin kuuluvat esimerkiksi Autodeskin kehittämät Maya ja 3ds Max. Näitä 3D-mallinnuksen ohjelmia käytetään nykyään Hollywood-elokuvissa ja suosituimmissa videopeleissä, mutta ovat myös erittäin kalliita tavalliselle harrastelijalle [12; 13]. Mallien luontiin on kuitenkin myös ilmaisia vaihtoehtoja. Näistä suosituimpiin kuuluva on avoimen lähdekoodin Blender. Vaikka Blenderin ominaisuudet eivät ole yhtä laajat kuin esimerkiksi Mayan, pystyy ohjelmalla silti luomaan ammattilaistason malleja ja animaatioita. [14.]

3D-malleja voi luoda myös selaimessa. Spline on täysin selaimessa toimiva ilmainen 3D-mallinnustyökalu. Spline on helppokäyttöisempi kuin täysin varustellut 3D-sovellukset ja luodut mallit voidaan lisätä suoraan verkkosivulle ilman ulkoisia kirjastoja. Tämä tekee Splinesta hyvän vaihtoehdon yksinkertaiseen 3D-mallinnukseen verkkosivuilla. Monimutkaisempiin projekteihin Spline saattaa kuitenkin olla liian pelkistetty. [15.]

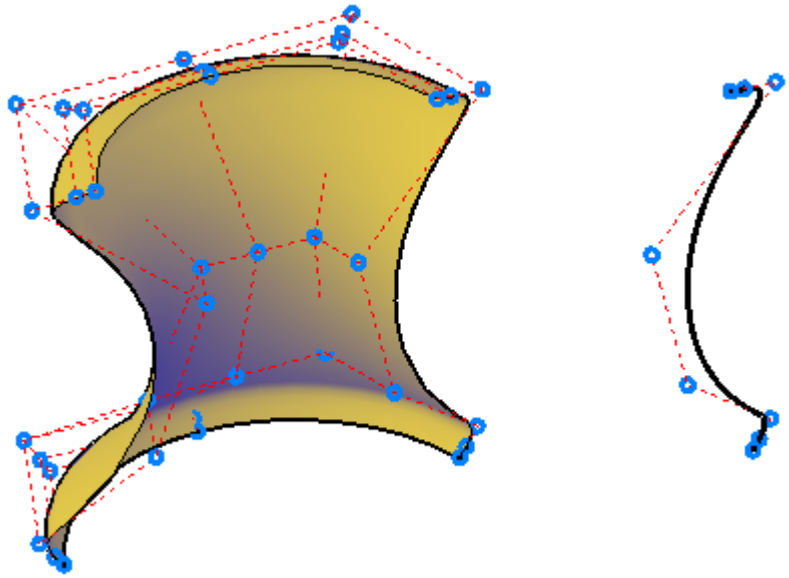
Mallinnuksessa puhutaan usein polygonimalleista. Polygonimallit muodostuvat joukosta tahoja (engl. face), tahot muodostuvat särmistä (engl. edge) ja särmät verteksipisteistä (engl. vertex, monik. vertices). Usein polygonit yksinkertaistetaan joukoksi kolmioita tai neliöitä. Mitä enemmän näitä yksittäisiä polygoneja mallissa on, sitä tarkempi malli on. Kuvassa 4 on esimerkki mallinnuksessa apuna käytetystä normaalikartoituksesta (engl. normal mapping), jolla saadaan mallille lisää yksityiskohtia ilman suurta lisäystä polygonien määrään. Normaalikartoituksessa ei tallenneta syvyyttä, vaan normaalin suunta. [16; 17.]




Kuva 4. Esimerkki normaalikartoituksesta [18]

Kuvassa vasemmalla alkuperäinen malli, johon on käytetty neljä miljoonaa polygonikolmiota. Keskellä on yksinkertaistettu mesh, jossa on vain viisisataa kolmiota ja oikealla sama yksinkertaistettu malli normaalikartoituksella. Patsaat näyttävät vasemmalla ja oikealla lähes samalta, mutta polygoneja on vain murto-osa.

Toinen tapa esittää 3D-malleja on NURBS (engl. Non-uniform rational basis spline). NURBS-mallinnuksessa käytetään polygonien sijaan käyriä. Tietokonegrafiikassa käyrä on oikeastaan vain monta pienempää viivaa. Kun viivoja on riittävän paljon, kuvio näyttää ihmissilmään käyrältä. NURBS-käyrä käyttää kontrollipisteitä, joiden välille käyrä lasketaan matemaattisesti. Kuvassa 5 on sinisellä merkattu kontrollipisteet. Oikealla kuvassa on pienemmästä määrästä kontrollipisteitä luotu käyrä ja vasemmalla useasta pisteestä luotu NURBS-pinta. [19, s. 64.]



Kuva 5. NURBS-pinta ja käyrä. Kontrollipisteet on merkitty sinisellä [20]

NURBS-mallinnus eroaa polygonimalleista adaptiivisella renderöinnillä. Tämä tarkoittaa, että mallin pikselien määrä voi muuttua, kun mallin koko muuttuu. Adaptiivisen renderöinnin ansiosta malli on tarkka minkä kokoisena tahansa. Sama tapahtuu esimerkiksi tietokoneen tekstissä. Tekstiin zoomatessa sen koko suurenee ja pysyy tarkkana. Polygonimalleilla on rajattu määrä pisteitä, särmiä ja tahoja, mutta NURBS-mallissa myös käyrien välille luotu pinta muuttuu polygoneiksi adaptiivisesti. Tätä pinnan jakamista pienempiin primitiiveihin kutsutaan tesselaatioksi. [19, s. 66, 79.]

Malleilla on lähes aina jokin materiaali. Materiaalilla tarkoitetaan geometrian näkyvän pinnan ominaisuuksia, kuten sen väriä, heijastuvuutta ja pintakuviointia. Pintakuviointi muodostuu esimerkiksi toistuvasta kuviosta ja sitä käytetään antamaan mallille syvyyttä tai vaikutelma siitä, että malli on tiettyä materiaalia. Mahdolliset ominaisuudet materiaalille määräytyvät varjostimen eli shaderin valinnassa. [19, s. 116; 21.]

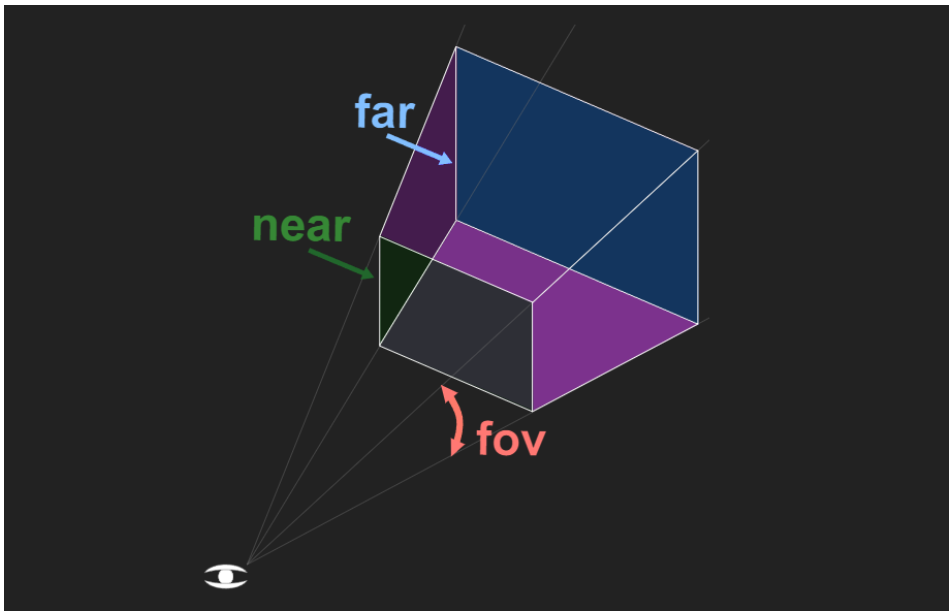
Valot luovat varjotehosteita ja vaikuttavat siihen, miten materiaalit näytetään. Mallin valotus toimii käyttämällä verteksien normaalitietoja. Normaalitiedoista

lasketaan varjostus ja hahmonnetaan valoisat ja tummat pikselit. Hahmontamisella tarkoitetaan 2D- tai 3D-geometrian muuntamista matemaattisin keinoin pikseleiksi ja niiden esittämistä näytöllä. Valaistus on tärkeä osa mallin realistisuudessa. Ilman yhtään valoa malli näyttää täysin tummalta. Huonolla valaistuksella ilman varjoja ja syvyyttä malli voi taas näyttää kaksiulotteiselta. [19, s. 111, 116.]

Toinen tarvittava elementti mallin tarkasteluun on kamera. Kamera määrittelee, mitä ruudulle hahmonnetaan. Malli on olemassa ilman kameraa matemaattisena tietona, mutta vaikka malli olisi hyvin valaistu, olisi ruutu täysin tyhjä. Kamera luo myös perspektiivin ja parallaksin. Kuten luvussa 2.2 mainittiin, parallaksia voi käyttää syvyyden illuusion luomiseksi liikuttamalla taustan elementtejä hitaammin. Kameroissa parallaksilla tarkoitetaan sitä, että kauempana olevat kappaleet näkyvät pienempänä. Toinen vaihtoehto on ortografinen perspektiivi. Ortografinen perspektiivi säilyttää kappaleiden kokosuhteen riippumatta niiden sijainnista. Kauempana Z-akselilla sijaitseva kappale näkyy samankokoisena kuin lähempänä sijaitseva kappale. [19, s. 117–118.]

Kameran polttovälillä tarkoitetaan käytännössä objektiivin pituutta. Tämä vaikuttaa siihen, kuinka laajan kuvan kameralla voi kuvata. Mitä laajempi kuva on otettava, sitä pienempi kameran polttoväli täytyy olla. Suuremmilla polttoväleillä saa tiukemman kuvan, mutta suuri polttoväli aiheuttaa myös kalansilmäefektin. Tällä tarkoitetaan sitä, että kuva vääristyy sen reunoilla. [19, s. 119.]

Kameralle on myös määriteltävä alue, jonka se hahmontaa. Myös frustumiksi kutsuttu alue määritellään leikkaustasoilla. Leikkaustasojen korkeus määräytyy näkökentän mukaan. Leikkaustasojen leveyteen vaikuttaa näkökentän lisäksi kuvasuhde. Kuvassa 6 näkyy leikkaustasojen väliin muodostuva katkaistun kartion muotoinen alue. Kamera hahmontaa vain tämän alueen sisällä olevat asiat. [19, s. 119–120; 22.]




Kuva 6. Leikkaustasojen muodostama alue, jonka sisällä olevat asiat renderöidään [22]

Leikkaustasoja on aina yksi lähempänä ja yksi kauempana. Pienten esineiden kanssa lähempänä oleva leikkaustaso tulee olla lähempänä kameraa kuin isojen esineiden kanssa. Jos leikkaustaso olisi kaukana, pientä esinettä olisi vaikea nähdä. [19, s. 119–120; 22.]

## 3.2 2D- ja 3D-grafiikan hahmontaminen

### 3.2.1 OpenGL

OpenGL on ohjelmointirajapinta eli API (engl. Application Programming Interface) tietokonegrafiikan hahmontamiseen. Hahmontamiseen on pääasiassa kaksi keinoa. Toinen tapa hahmontaa on ohjelmiston hahmontaminen, joka käyttää tietokoneen keskusyksikköä (engl. Central processing unit eli CPU). Ohjelmiston hahmontamisella pystytään ottamaan kaikki hahmonnettavat asiat, joita ovat valaistus, varjot ja pintakuviointi, ja muuttamaan ne kuvatiedostoksi. Tämä on hidas prosessi ja yhden kuvaruudun hahmontamiseen voi mennä useita tunteja. Lopputulos voi kuitenkin olla hyvin realistinen, mistä syystä tekniikkaa käytetään esimerkiksi elokuvissa. [19, s. 113–114.]

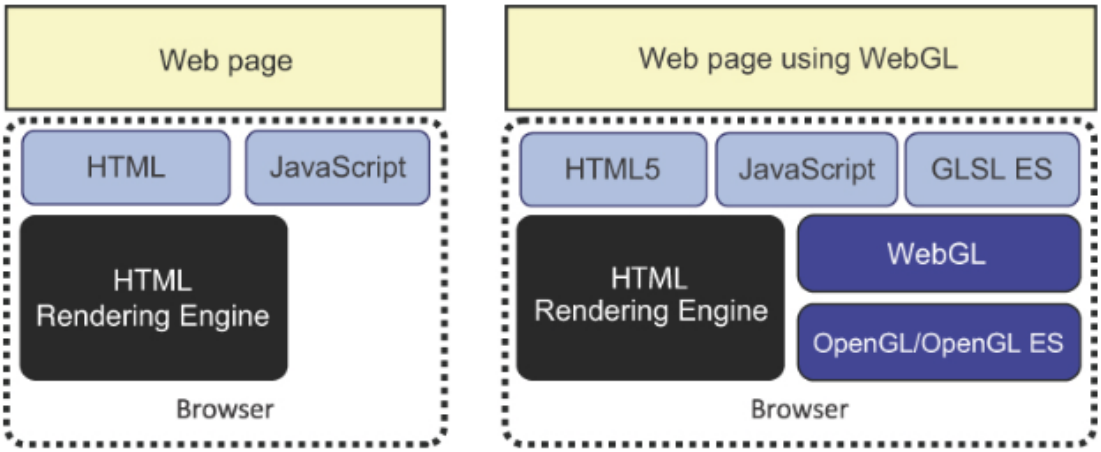
Hahmontaminen voi tapahtua myös reaaliaikaisesti. Tällöin hahmontaminen tehdään useimmiten tietokoneen grafiikkasuorittimella (engl. Graphics processing unit eli GPU). Koska reaaliaikainen hahmontaminen on vaativaa, on kehitetty OpenGL kaltaisia ohjelmointirajapintoja käsittelemään nopeasti muuttuvaa tietoa laskennallisin keinoin. [19, sivut 113–114.]

Koordinaatistossa 3D-mallinnus onnistuu lisäämällä perinteiseen kaksiulotteiseen x- ja y-akselin koordinaatistoon syvyysakseli z. 3D-tietokonegrafiikka on kuitenkin litteän tietokoneen näytön takia vain kaksiulotteisia kuvia, joissa on illuusio syvyydestä. Tästä syystä OpenGL vaatii saman illuusion saavuttamiseen matematiikkaa, kuten trigonometriaa ja matriisin manipulointia. Syvä matematiikan ymmärrys ei kuitenkaan ole välttämätön tietokonegrafiikan luomiseen. [23, luku 1.]

### 3.2.2 WebGL

WebGL (Web Graphics Library) on JavaScript ohjelmointirajapinta tietokonegrafiikan renderöimiseen verkkosivuilla. Rajapinta perustuu OpenGL ES -rajapintaan (Open Graphics Library for Embedded Systems, OpenGL sulautetuille järjestelmille) ja on siten rakenteeltaan sitä hyvin lähellä. WebGL mahdollistaa liitännäisvapaan 3D-mallinnuksen suoraan selaimeen. [24.]

WebGL on varjostinpohjainen ja koostuu täten JavaScript-koodista sekä shader-koodista. Varjostimella eli shaderilla tarkoitetaan tietokoneohjelmaa, jonka avulla ohjelmoidaan visuaalisia tehosteita. Varjostinkoodi on rajapinnassa kirjoitettu GLSL-varjostinkielellä (engl. OpenGL ES Shading Language) ja toteuttaa WebGL-piirtotoiminnon. Kuvassa 7 verrataan tavallisen verkkosivun ja WebGL:ää käyttävän verkkosivun rakennetta. Vasemmalla on tavallinen verkkosivu ilman WebGL:ää ja oikealla WebGL:ää käyttävä verkkosivu, joka tarvitsee myös shader-kieltä GLSL. [25, luku 1.]

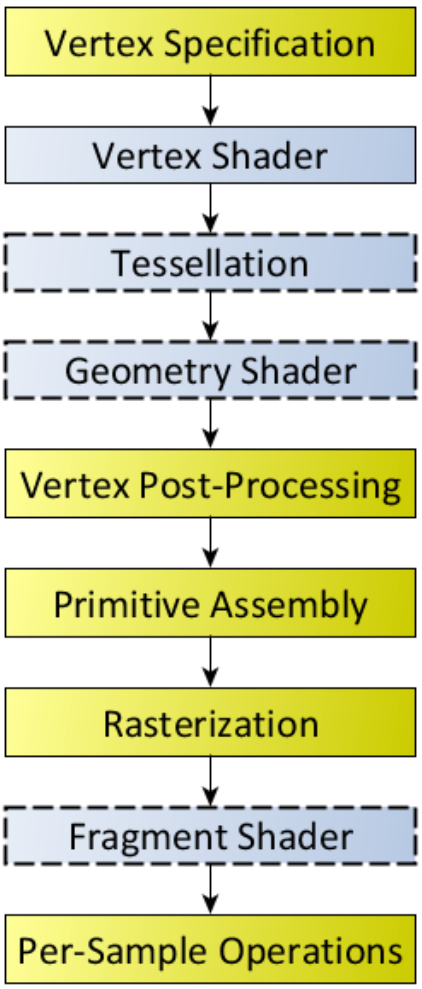



Kuva 7. Vertailu sivuston rakenteesta ilman WebGL:ää (vas.) sitä käyttävän sivuston kanssa [25, luku 1]

Varjostinkieli GLSL kirjoitetaan usein JavaScriptilla, joten kehittäjä ei välttämättä tarvitse JavaScript- ja HTML-tiedostojen lisäksi muita tiedostoja. WebGL monimutkaistaa JavaScriptiä, mutta rakenne pysyy tiedostojen kannalta samana. [25, luku 1.]

WebGL tukee verteksi- ja fragmenttivarjostimia. Jo aiemmin mainitulla verteksillä tarkoitetaan tietokonegrafiikassa pistettä, jolla on koordinaateilla määritetty sijainti. Sijainnin lisäksi vertekseille voidaan määritellä esimerkiksi väri, tekstuuri ja heijastus. Verteksivarjostimet käyttävät näitä verteksitietoja ja lisäävät matemaattisilla operaatioilla 3D-malliin eri tehosteita. [25, luku 6.]

Hahmontaminen tapahtuu grafiikkaliukuhihnan mukaan (engl. Rendering pipeline). Kuvassa 8 on diagrammi grafiikkaliukuhihnasta, jossa ensin verteksi syötetään varjostimeen ja varjostin kirjoittaa arvot lähtöverteksiksi (engl. Output vertex). Verteksitiedolle saatetaan valinnaisesti suorittaa tesselaatio ja primitiivejä voidaan myös käsitellä geometriavarjostimella. Primitiiveillä tarkoitetaan yksinkertaisia geometrisiä muotoja, kuten kolmioita ja viivoja. Geometriavarjostin pystyy käsittelemään primitiivien verteksipisteitä ja muokata primitiivejä esimerkiksi poistamalla niitä tai muuttamalla niiden tyypin kolmiosta pisteeksi. Geometriavarjostin siis käsittelee saapuvat primitiivit ja palauttaa nollan tai useamman lähtöprimitiivin. [26.]




Kuva 8. OpenGL-renderöintiputki [26]

Rasterointi tarkoittaa primitiivien jakamista fragmentteihin. Fragmentit ovat tilajoukkoja, joita käytetään pikselin lopullisen datan laskemiseen. Fragmenttivarjostin laskee jokaiselle rasteroinnin luomalle fragmentille värin ja syvyyden. Viimeisessä renderöintiputken vaiheessa Per-Sample Operations fragmentit käsitellään ja niistä saatu data kirjoitetaan eri puskureihin. [26.]

GLSL-varjostimet tarvitsevat main-funktion, josta sovellus suoritetaan. Tämä eroaa JavaScript-koodista siten, että JavaScript-sovellukset suoritetaan järjestyksessä rivi kerrallaan. Toinen ero JavaScriptiin on, että varjostimissa funktion

palautettava arvo merkataan sen tyypillä. Jos arvoa ei palauteta, liitetään funktioon avainsana void. GLSL ei tue esimerkiksi merkkijono-datatyyppiä vaan ainoat tuetut datatyypit ovat integer, float ja boolean. [25, luku 6.]

GLSL-data voi olla vektori- tai matriisimuodossa. Kuvassa 9 esimerkit vektorimuotoisesta ja matriisimuotoisesta datasta. Vasemmalla kuvassa vektori, joka järjestelee tiedon listaksi. Vektoreita käytetään esimerkiksi koordinaattien ja väridatan esittämiseen. Matriisi järjestelee tiedon taulukoksi ja niitä käytetään esimerkiksi kuvausmatriiseihin (engl. Transform matrix), jolla pystytään muokkaamaan kappaleiden kokoa, sijaintia ja orientaatiota. [25, luku 6.]

$$(3 \quad 7 \quad 1) \qquad \begin{bmatrix} 3 & 7 & 1 \\ 1 & 5 & 3 \\ 4 & 0 & 7 \end{bmatrix}$$

Kuva 9. Vektori- ja matriisidata [25, luku 6]

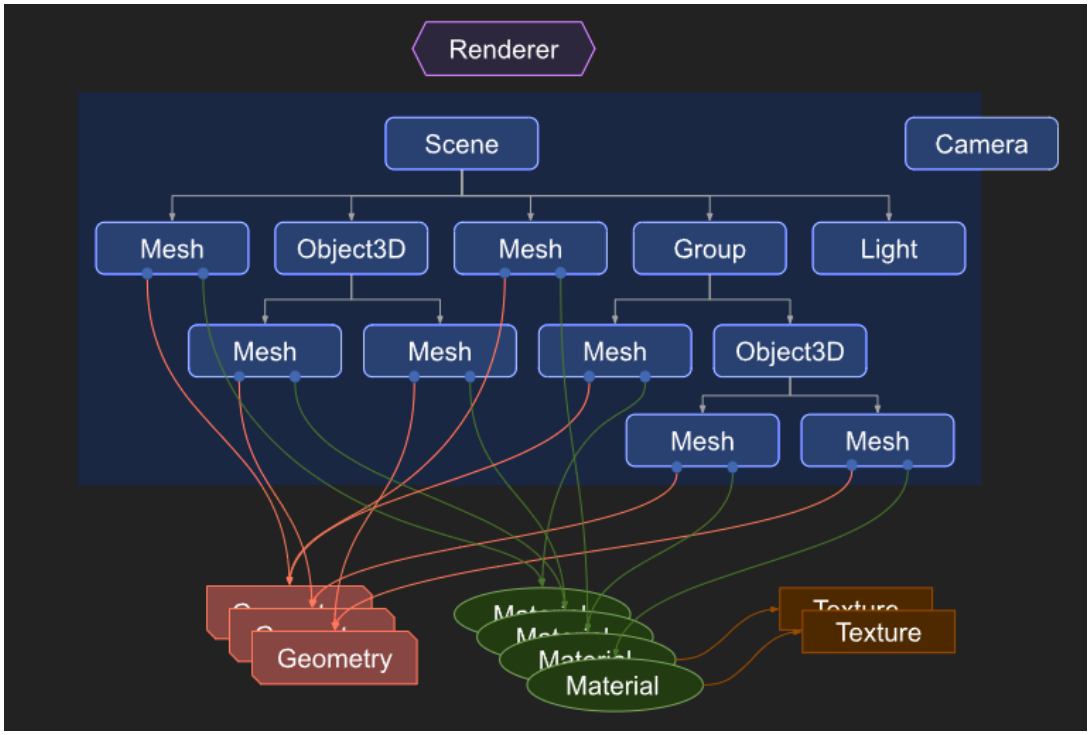
## 3.3 Grafiikkakehykset ja -moottorit selaimessa

Yksinkertainen 3D-mallinnus ei enää vaadi kehittäjältä supertietokonetta, vaan kuka vain pystyy luomaan WebGL-sovelluksen pelkällä tekstieditorilla ja selaimella. WebGL-ohjelmointi suoraan pelkällä JavaScript-koodilla on kuitenkin monimutkaista ja virhealtista [27]. Tämän takia apuna käytetään usein jotakin grafiikkakehystä. Koska kehyksiä on useita, tutkin niistä tarkemmin kolmea.

### Three.js

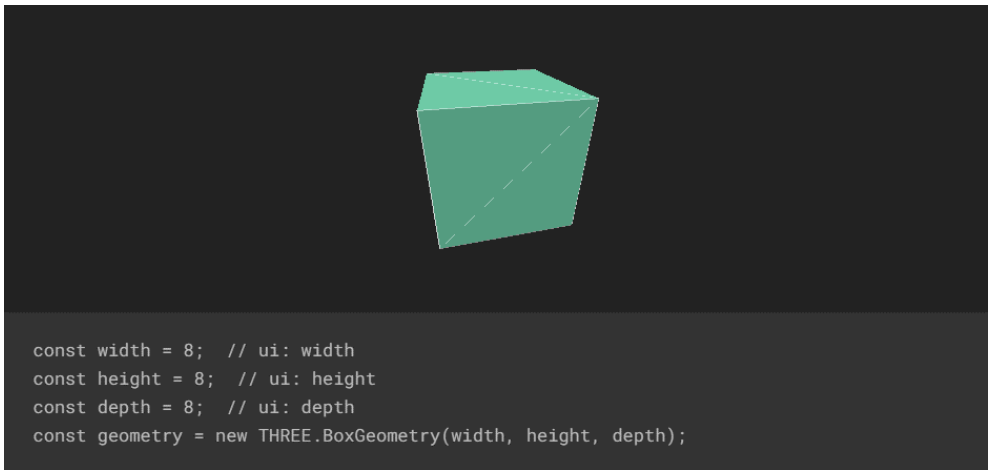
Three.js on WebGL-sisällön tuottamiseen käytettävä JavaScript-kirjasto, jotka ovat kokoelmia resursseja, kuten valmiita koodipätkiä, helpottamaan ja nopeuttamaan ohjelmointia. Three.js-sovellus rakentuu kohtaukseen (engl. scene). Kohtaus on rakenne, joka toimii konttina eri objekteille. Kohtaus on siis sovelluksen juuressa ja määrittelee, mitä asioita sovellus hahmontaa. Kuvassa 10 näkyy

pienen Three.js-sovelluksen rakenne. Sovellus koostuu mesh-objekteista, kamerasta ja valosta. [22.]




Kuva 10. Pienen Three.js-sovelluksen rakenne [22]

Mesh-objektit rakentuvat geometriasta ja materiaalista. Geometrialla tarkoitetaan hahmonnettavaa esinettä, kuten matkapuhelinta tai autoa, joka luodaan useimmiten 3D-mallinnusohjelmassa. Tämän jälkeen malli ladataan sivustolle tiedostosta. Three.js tarjoaa myös valmiita malleja yksinkertaisille esineille, kuten kuutioille ja palloille, joita pystytään muokkaamaan eri parametreillä. Kuvassa 11 on esimerkki laatikkoprimitiivin luominen Three.js-koodissa. [22.]




Kuva 11. Geometrian luonti tarjotulla laatikkoprimitiivillä. Laatikon korkeus, leveys ja syvyys asetetaan parametreilla. [28]

Myös materiaali tulee useimmiten mallinnusohjelmasta tiedoston mukana. Materiaalin pintakuviointi on usein tiedostolta ladattu kuva, joka kiedotaan geometrian ympärille. Yhtä materiaalia sekä tekstuuria voi käyttää useassa geometriassa. [22.]

Three.js tarjoaa useita eri kameravaihtoehtoja. Näihin kuuluu esimerkiksi aiemmin mainittu ortografinen kamera eli OrtographicCamera, jonka avulla kuvattu objekti pysyy samana riippumatta siitä, millä etäisyydellä kamera on. Eniten käytetty kamera on kuitenkin PerspectiveCamera, joka imitoi ihmissilmää. Kameraa luotaessa sille asetetaan eri parametrejä. Näkökentän ja kuvakulman lisäksi annetaan near- ja far-luvut, jotka määrittelevät leikkaustasojen sijainnit ja luovat alueen, jonka sisällä olevat asiat kamera hahmontaa. [22.]

3D-mallien hahmontamiseen tarvitaan hahmontaja. Hahmontaja vastaa siitä, miltä kohtaus näyttää ja miten mallit hahmonnetaan kameran perusteella. Nykyään käytössä on lähinnä vain WebGLRenderer. Ennen käytössä on ollut myös muita vaihtoehtoja, kuten CanvasRenderer, mutta näistä on puuttunut hyödyllisiä ominaisuuksia ja suorituskyky on ollut heikompi. Vuonna 2018 Three.js-kehittäjä Ricardo Cabello ilmoitti sosiaalisessa mediassa, että CanvasRenderer poistetaan kirjastosta, koska WebGL tuetaan kaikkialla. [22; 29, luku 1; 30.]

Kohtauksen hahmontamiseen tarvitaan koodissa hahmontamis- tai animaatiosilmukka. Tämä tarkoittaa, että kohtaus piirretään näkyviin ruudulle jokaisella näytön kehyspuskurin päivityksellä [31]. Päivityksien määrä riippuu näytön virkistystaajuudesta, joka on suurimmassa osassa näyttöjä 60 hertsiä eli 60 Hz. Koska virkistystaajuus vaikuttaa näytöllä näkyvien asioiden sulavuuteen, voi pelinäytöissä luku olla jopa 500 Hz.

Babylon.Js

Babylon.Js on alun perin Microsoftin kehittämä avoimen lähdekoodin grafiikkamoottori. Babylonin tunnettuja käyttökohteita ovat esimerkiksi Niken ”Nike By You” -kenkiensunnittelusovellus, jossa käyttäjä pystyy vaihtamaan kengän osien materiaaleja sekä värejä ja suunnittelemaan kenkänsä. Myös kauppaketju Target on käyttänyt Babylonia kehittämään 3D-sisustussovelluksen, jossa käyttäjä pystyy määrittelemään huoneen koon ja suunnittelemaan sen sisustuksen asettamalla tuotteita 3D-tilaan. [32; 33.]

Tutkin kahta Three.js- ja Babylon-kehyksillä luotua yksinkertaista sovellusta, jotka sisälsivät laatikon, kameran ja valon. Joitain eroja sovellusten väliltä löytyi. Molemmissa kehyksissä on mahdollista simuloida mallille valo ympäristöstä. Kyseistä Three.js valoa kutsutaan HemisphereLight-nimityksellä ja Babylonissa HemisphericLight-nimityksellä. Valotuksen toteutus kuitenkin erosi kehysten välillä. Babylonissa valoa luodessa valolle annettiin valon nimi, valon heijastumissuunta sekä kohtaus, johon valo kuuluu. Valon luonnin jälkeen valolle pystyy vielä asettamaan erikseen värityksen. Three.js-koodissa valolle voidaan antaa suoraan taivaan väri, maan väri sekä valon teho. Tämän jälkeen valo lisätään kohtaukseen. Monet erot voivat kuitenkin olla koodaajan näkökulmasta lähinnä nimityksissä. Esimerkiksi Three.js-sovelluksessa luodaan renderer-objekti ja Babylonissa tämän korvaa BABYLON.Engine.

Vaikka joitain eroja löytyy kehyksien välillä toimintojen toteutuksessa, sovellusten perusrakenne pysyy hyvin samankaltaisena. Three.js vaatii tietyissä asioissa useamman rivin koodia verrattuna Babyloniin, mutta esimerkkisovellusten

perusteella Three.js tuntui omaan käyttötarkoitukseen yksinkertaisemmalta. Monimutkaisempiin sovelluksiin Babylon saattaa kuitenkin olla parempi vaihtoehto esimerkiksi sen fysiikkamoottoreiden käytön mahdollistavan laajennusjärjestelmän ansiosta [34].

PlayCanvas

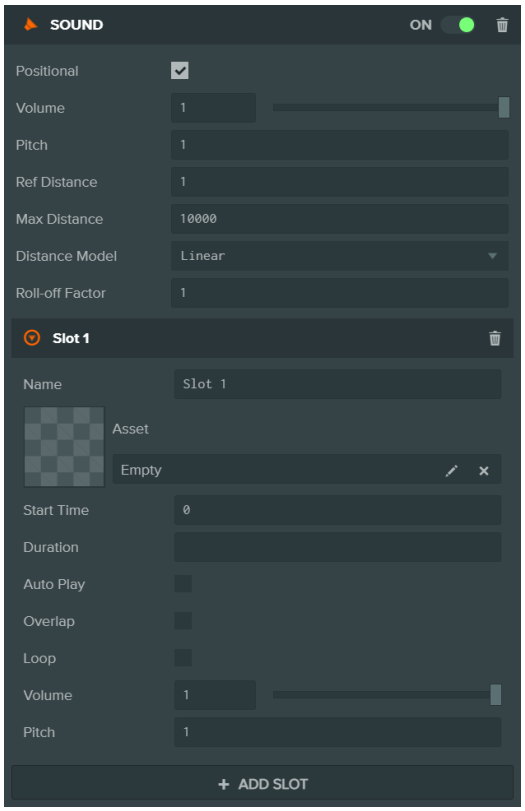
Toinen suosittu avoimen lähdekoodin 3D-moottori on 2014 julkaistu PlayCanvas, joka on suunniteltu erityisesti pelien luomiseen. PlayCanvas eroaa Babylonista ja Three.js siten, että se tarjoaa myös tehokkaan editorin. Kuvassa 12 näkyvä editori mahdollistaa uusien objektien luomisen ja muokkaamisen ilman koodin kirjoittamista. [35; 36.]


Kuva 12. PlayCanvas-editori [37]

PlayCanvasissa luoduissa sovelluksissa on samoja perusosia kuin aikaisemmin mainituissa kehyksissä. Pelissä on lähes aina kamera, valoja ja 3D-malli. PlayCanvasissa 3D-malli on osa laajempaa kokonaisuutta, joita kutsutaan nimellä Entity. Entity rakentuu komponenteista (engl. Component), joiden toiminnan määrittää komponenttijärjestelmät (engl. ComponentSystem). [38.]

PlayCanvasin omassa esimerkissä yksinkertainen Entity koostuu kolmesta komponentista: ääni-, malli- ja skriptikomponentista. Mallikomponentti on siis 3D-malli tai -primitiivi, joka renderöidään Entityn sijaintii, kun mallikomponentti aktivoidaan. Esimerkissä mallina käytetään jalkapalloa. [39.]

Äänikomponentti mahdollistaa äänitiedostojen soittamisen. Kuvassa 13 on yksinkertainen äänikomponentti yhdellä paikalla. Entitylle lisätään paikka jokaiselle tarvitulle äänitiedostolle, ja kehittäjä pystyy määrittämään editorissa esimerkiksi aloitusajan, keston ja äänitiedoston silmukan. Esimerkkisovelluksessa jalkapallolle asetetaan yksi paikka äänelle, joka toistuu, kun pallo pomppii. [39.]




Kuva 13. PlayCanvas-äänikomponentti [40]

Skriptit tarkoittavat kooditiedostoja, jotka hoitavat sovelluksen toiminnot. Vaikka PlayCanvasin editori vähentääkin koodin kirjoittamista, kaikki interaktiiviset sovellukset tarvitsevat silti yhden tai useamman skriptin. Jalkapallopelissä skriptit hoitaisivat esimerkiksi pallon pyörimisen. Esimerkkikoodissa 1 PlayCanvasin

käyttöoppaassa annettu esimerkki esineen kierittämisestä rotate-metodilla. Esinettä kieritetään kymmenen astetta joka sekunti. [39; 41.]

```
var Rotate = pc.createScript("rotate");

Rotate.prototype.update = function (dt) {
    this.entity.rotate(0, 10*dt, 0);
};
```

Esimerkkikoodi 1. Yksinkertainen animaatioskripti PlayCanvasissa

# 4 Projektisivusto

Projektisivuston tavoitteena on luoda 3D-malli ja esittää se verkkosivustolla. Samalla esittelen yleisesti käytettyjä mallinnuksen tekniikoita sekä tärkeimmät osat koodista. Luon itse 3D-mallia ensimmäistä kertaa, joten projektin on tarkoitus tutkia, miten mallin luominen ja esittäminen omalla verkkosivulla onnistuu ilman 3D-mallinnuksen kokemusta.

Teknologiat

Valitsin mallinnussovellukseksi Blenderin, koska sovellus on ilmainen, mutta pystyy kuitenkin ammattilaistason hahmontamiseen. Blender on helppokäyttöinen ja tukea sekä oppitunteja on laajasti saatavilla.

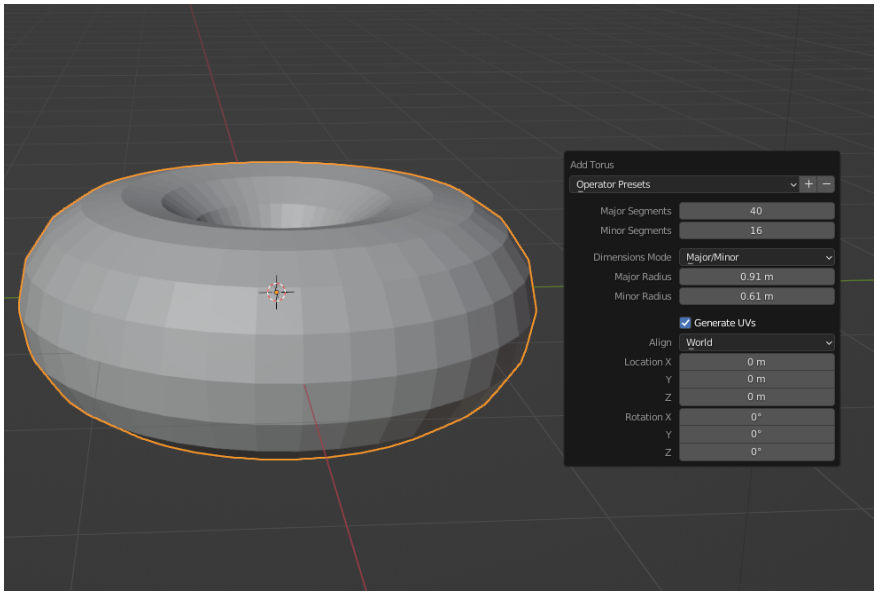
Mallinnan työssäni 3D-donitsin käytettäväksi verkkosivulla. Donitsi on simppeli muoto, mutta mahdollistaa silti tärkeiden 3D-mallinnuksen tekniikoiden ja teknologioiden käytön. Mallina donitsin luomiseen käytän BlenderGuru-kanavan YouTube-videosarjaa. Mallinnuksen tavoitteena on luoda itse donitsin muoto, donitsin kuorrute sekä kuorrutteen päällä olevat strösselit.

Koodissa grafiikkakehyksenä käytän Three.Js:tä, koska kehyksiin tutustuessa kyseinen kehys tuntui yksinkertaisimmalta ja ohjeita löytyi runsaasti. Donitsin mallintamisen lisäksi luon sivustolle yksinkertaisen käyttöliittymän. Sivuston käyttäjä pystyy muokkaamaan käyttöliittymällä mallin materiaalien väriä reaaliaikaisesti.

Projektin vaiheet

Aloitan mallinnuksen Blenderissä rinkulamaisella torus-muodolla. Torus muistuttaa jo valmiiksi hieman donitsia, vaikkakin keskusta on liian leveä ja muoto on täysin yhtenäinen ja sileä ilman pintayksityiskohtia. Asetan parametreistä muodon paksummaksi ja annan sille aluksi suhteellisen pienen määrän pintoja, jotta mallia on helpompi muokata ja tiedostokoko pysyy pienempänä. Koska muoto on aluksi usean metrin kokoinen, pienennän sen oikean donitsin mittoihin.

Kuvasta 14 voi huomata, että muoto on vielä melko kulmikas, joten sille lisätään Subdivision Surface -modifioijan, joka jakaa muodon pinnat useampaan osaan ja antaa muodolle sileämmän ulkomuodon [42]. Realistisemman paksuuden ja leveyden vaihtelun saamiseen mallille käytän Proportional Editing -tilaa, jossa pystyn muokkaamaan useamman verteksipisteen sijaintia. Yksittäisten verteksipisteiden muokkaamiseen menisi kauan ja lopputuloksesta tulisi luultavasti liian töyssyinen.




Kuva 14. Kulmikas malli ennen muotoilua

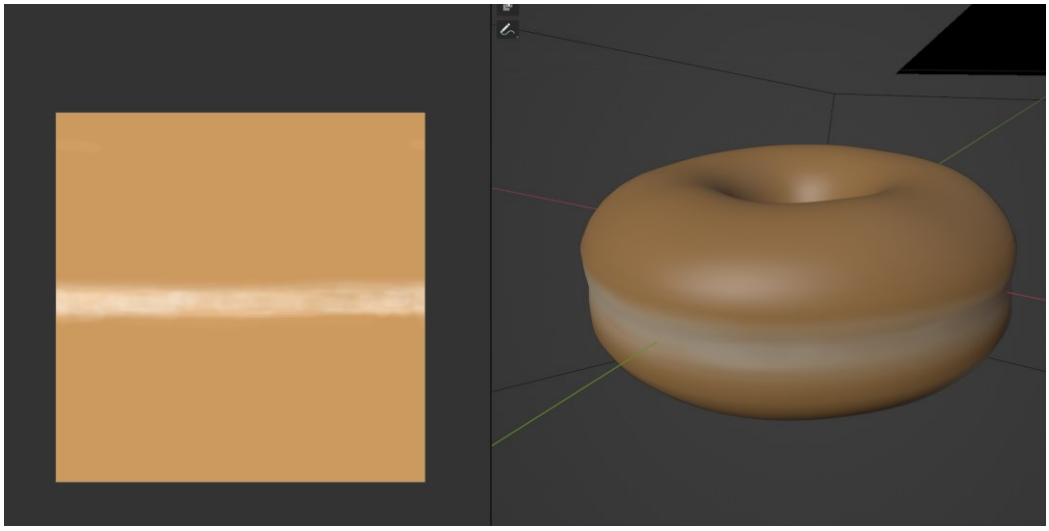
Kuorrute luodaan kopioimalla donitsin ylempi puolikas jättämällä se samaan kohtaan alkuperäisen mallin päälle. Mallia tutkitaan rautalankatilassa, joka auttaa valitsemaan koko donitsin puolikkaan tarkasti. Tällä hetkellä kopioitu puolikas on vain ohut pinta, joten puolikkaaseen lisätään Solidify-modifioija, jonka avulla säädellään kuorrutteen paksuutta. Kun kuorrutteen paksuus on valittu, kuorrutteen muotoa muokataan halutulla tavalla siirtämällä tiettyjä verteksipisteitä. Muotoilussa käytän apuna referenssikuvia oikeista donitseista, koska usein oma käsitys siitä, miltä asia näyttää ja miltä se oikeasti näyttää, eroaa.

Kuorrute näyttää vielä hyvin yhtenäiseltä. Tästä syystä kuorrutteeseen luodaan muotoja ja paksuutta tiettyihin kohtiin käyttämällä Blenderin Sculpting-tilaa. Tila tarjoaa useita työkaluja mallin veistämiseen, mutta tässä työssä käytän Inflate-, Grab- ja Smooth-työkaluja. Inflate-työkalulla täytetään ja tyhjennetään kohtia. Tällä voidaan luoda donitsin alaosaan kuorrutteen valumisesta muodostuvia pisaroita, mutta työkalu auttaa myös tekemään kuorrutteesta realistisemman paksuuden muutoksilla. Grab-työkalulla tartutaan ja muovataan haluttuja kohtia ja Smooth-työkalu tasaa ja sileyttää liian kuoppaisia kohtia.

Kuorrute on oikean muotoinen, mutta näyttää leijuvan donitsin päällä. Tämä johtuu siitä, että donitsin keskiosa on muotoiltu kapeammaksi kuin ylä- ja alaosa. Muotoilu tehtiin valitsemalla donitsin ulkopuolen keskiosassa olevat verteksit ja siirtämällä niitä donitsin keskustan suuntaan. Kyseistä muotoilua ei kuitenkaan ole tehty silloin, kun donitsin puolikas kopioitiin kuorrutetta varten. Kuorrute kääritään donitsin muotoon käyttämällä ShrinkWrap-modifioijaa, joka kutistaa objektin toisen objektin pintaan [43]**.**

Donitsi on halutun muotoinen, mutta pinta on yksivärinen ja tasainen. Blender mahdollistaa pintakuvioinnin maalaamisen Texture Paint -tilassa. Kuvassa 15 näkyy vaihtoehdot pintakuvioinnin maalaamiseen. Pintakuviointi voidaan maalata neliön muotoiselle alueelle tai suoraan 3D-malliin. Yritin ensin maalata suoraan 3D-malliin, mutta totesin, että malli ei maalautunut tarpeeksi tasaisesti.

Maalaan donitsin vaaleammaksi sen kapeammasta keskikohdasta, joka kuvastaa, että donitsi ei ole paistunut täysin tasaisesti.



Kuva 15. Donitsin maalaaminen halutun väriseksi

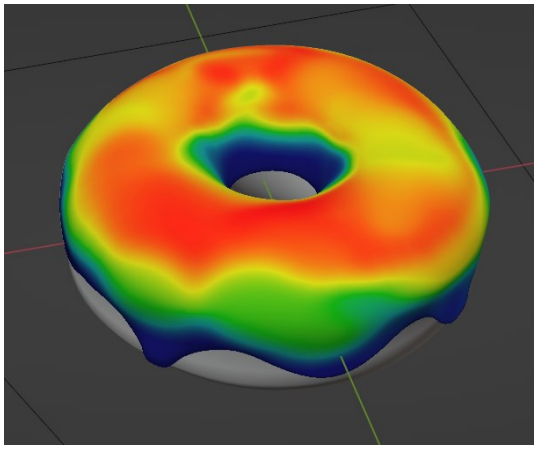
Blender käyttää varjostussolmuja eri tehosteilla, joita voidaan yhdistää toisiinsa. Donitsi käyttää Principled BSDF -varjostinta, joka perustuu Disneyn principled-malliin [44]. Vaikka donitsi on muotoiltu realistisemmaksi ja mallin paksuus vaihtelee, mallin pinta on vielä sileä. Syvyyden illuusion luomiseksi mallin pintaan yhdistän varjostimeen normaalikartoitussolmun. Koska haluan, että normaalikartoituksessa on satunnaisuutta, yhdistän tämän kohinasolmuun. Nämä solmut ovat yhteydessä myös väriä säätelevään solmuun. Yhdessä solmut tekevät pinnasta töyssyisen, ja donitsin väri vaihtelee vaaleasta tummaan satunnaisesti.

Viimeisenä donitsi tarvitsee strösselit. Koska jokaisen strösselin muotoilu yksitellen olisi hyvin työlästä, voidaan Blenderissä käyttää geometriasolmuja niiden generoimiseen muutaman strösselin perusteella. Geometriasolmuilla on tulo- ja lähtösolmu, jonka väliin lisätään solmuja eri efektien luomiseksi.

Ensin muotoilen neljä hieman toisistaan eroavaa strösseliä pienistä lieriöistä. Näistä neljästä luodaan kokoelma, jonka perusteella kaikki strösselit luodaan. Jos strösseleihin haluaisi enemmän eroavuutta, voisi kokoelmaan lisätä helposti uuden lieriön. Generointi toteutetaan solmujen kuten Distribute Points on Faces

ja Instance on Points avulla. Distribute Points on Faces -solmulla strösselit saadaan kiinnitettyä kuorrutteen pintoihin. Instance on Points -solmulla taas luodaan instansseja, jotka mahdollistavat geometrian toistamisen ilman tietojen kopioimista. Ensin strösselit olivat väärän kokoisia ja väärässä kulmassa, minkä seurauksena ne menivät kuorrutteen läpi. Kokoa ja kiertoa voidaan muokata kokoelmassa oleviin lieriöihin, ja kun muutokset otetaan käyttöön, se vaikuttaa jokaiseen luotuun strösseliin. [45; 46.]

Generoidut strösselit generoituvat koko kuorrutteeseen. Tämän korjaamiseen käytetään Weight Painting -menetelmää. Malliin väritetään kohdat, joihin strösselit voivat generoitua pensselin painon perusteella. Kuvassa 16 nähdään väritetty donitsi, joka muistuttaa lämpökarttaa. Punaisena näkyvät alueet generoivat eniten strösseleitä ja siniselle alueelle strösseleitä ei generoidu ollenkaan.



Kuva 16. Asetetaan maalaamalla alueet, mille alueille ja miten paljon strösseleitä halutaan generoida

Valmis donitsi tallennetaan .glb- eli glTF Binary -tiedostomuotoon. Tiedostomuoto mahdollistaa mallin ja sen tietojen sekä tekstuurien siirtämisen samassa tiedostossa. Nimi glTF tulee englannin kielen sanoista Graphics Language Transmission Format. Tiedostomuodon on kehittänyt myös OpenGL-kehittäjä Khronos Group. Testasin ensin mallin lataamista erilliselle glTF-tiedostojen tarkasteluun luodulle sivustolle. Malli lataantui sivustolle muuten oikein, mutta

strösseleitä ei näkynyt ollenkaan. Tähän löysin ratkaisun Realize Instances -solmusta, joka muuttaa luodut strösselit geometriaksi. [47; 48.]

Koodissa mallin lataamiseksi sivulle käytän Three.js omaa opasta .glTF-tiedostojen lataamiseen [49]. Ensin luon HTML-tiedostoon canvas-elementin Three.js-sovellukselle. Tämä toimii alueena, jolle sovellus rakentuu. JavaScript koodissa luon sovellukselle hahmontajan, kameran sekä uuden kohtauksen. Hahmontajana käytän WebGLRendereriä ja kamerana PerspectiveCameraa.

Sovelluksen hahmontamiseen tarvitaan hahmontamissilmukka. Ensin asetetaan render-metodilla, mikä kohtaus halutaan hahmontaa ja mitä kameraa käytetään. Silmukassa tarvitaan requestAnimationFrame-metodia, jonka avulla haluttu toiminto, eli tässä tapauksessa hahmontaminen, suoritetaan jokaisella näytön päivityksellä. Tarvitaan myös funktio tarkistamaan, että hahmontaja on oikean kokoinen. Hahmontajalle asetetaan funktiossa uusi koko, jos canvas on suurempi kuin näytön korkeus ja leveys. Jos uusi koko asetetaan, hahmontamissilmukka päivittää kameralle kuvasuhteen ja projektiomatriisin. Malli näyttäisi epämuodostuneelta ja matalaresoluutioiselta, jos päivityksiä ei tehtäisi.

Ensimmäisellä testillä sivusto näyttää vain valkoisen ruudun. Tulostuskomennolla pystyn kuitenkin helposti päättelemään selaimen konsolista, että JavaScript-tiedosto ei ollut yhdistetty oikein. Tämän korjasin helposti pienellä HTML-tiedoston muokkauksella. Jätän ensin kameran parametrit oletusarvoihin. Muokkaan näitä kuitenkin hieman, kun donitsi on saatu ladattua sivustolle.

Donitsimallin .glb-tiedosto ladataan GLTFLoader-lisäosalla. Esimerkkikoodissa 2 load-metodille annetaan tiedoston sijainti parametrinä ja lisätään malli kohtaukseen. Samalla luodaan laatikko, joka pitää mallin sisällään. Malli latautuu sivustolle, mutta en ole tyytyväinen kameran sijaintiin ja mallin kokoon. Muokkaan kameran sijaintia ja säädän kameran näkökenttää ja leikkaustasoja kameran parametreissa. Kun olen tyytyväinen kameraan, asetan mallille maksimi- ja minimietäisyyden kamerasta käyttämällä OrbitControls-lisäosaa. Etäisyydet

mahdollistavat zoomaamisen lähemmäksi ja kauemmaksi mallista hiiren rullaa käyttäen.

```
{
   const gltfLoader = new GLTFLoader();
   gltfLoader.load("../donut.glb", (gltf) => {
     loadedModel = gltf;
     const root = gltf.scene;

     const box = new THREE.Box3().setFromObject(root);
     const boxSize = box.getSize(new THREE.Vector3()).length();
     const boxCenter = box.getCenter(new THREE.Vector3());

     controls.maxDistance = boxSize * 5;
     controls.minDistance = boxSize * 2;
     controls.target.copy(boxCenter);
     controls.update();

     scene.add(root);
   });
 }
```

Esimerkkikoodi 2. 3D-mallin lataaminen Three.js-sovellukseen .glb-tiedostosta

Malli näkyy mustana, koska kohtauksessa ei ole valoja. Esimerkkikoodissa 3 näkyy valojen luonti ja lisääminen kohtaukseen. Valoja lisään kohtaukseen kaksi. Toinen valoista on ympäristöstä tuleva valaistus HemisphereLight, jolle asetan taivaan väriksi vaalean sinisen ja maan väriksi ruskean. Valo on ensin liian kirkas, joten lasken voimakkuuden 0,4:ään. Toinen valoista on Directional-Light, joka suunnataan suoraan donitsimalliin. Väriksi valitsen valkoisen ja voimakkuudeksi 0,8.

```
  {
    const skyColor = 0xb1e1ff;
    const groundColor = 0xb97a20;
    const intensity = 0.4;
    const light = new THREE.HemisphereLight(skyColor, groundColor, in-
tensity);
    scene.add(light);
  }

  {
    const color = 0xffffff;
    const intensity = 0.8;
    const light = new THREE.DirectionalLight(color, intensity);
    light.position.set(5, 6, 2);
    scene.add(light);
    scene.add(light.target);
  }
```

Esimerkkikoodi 3. Valojen luonti ja lisääminen kohtaukseen

Malli on tällä hetkellä liikkumaton, vaikkakin mallia voi kierrellä hiirtä käyttäen. Haluan, että malli pyörii ja kallistuu samalla hieman viistoon. Animoin mallia lisäämällä kiertoa x- ja y-akseliin render-funktiossa. Kiertoa lisätään malliin käyttämällä rotateX- ja rotateY-metodeja jokaisella näytön päivityksellä. Parametrina metodit saavat radiaaneina kulman, jonka malli kääntyy. Animointiin lisätään if-lause tarkistamaan, että rotaatiota yritetään vain silloin, kun malli on latautunut.

Haluan, että mallin värejä pystyy vaihtamaan sivustolla. Värivalitsin pohjautuu web-suunnitteluun keskittyvän Codrops-sivuston esimerkkiin [50]. Värien vaihtamisen voisi toteuttaa myös esimerkiksi Dat.GUI Three.js -kirjaston valikolla, mutta valikkoa voi olla haastava saada halutun näköiseksi.

Luon ensin värivaihtoehdoille objekteja ja laitan ne taulukkoon. Tämän jälkeen luon värejä vastaavan valikkonäkymän sivustolle. Kun käyttäjä painaa hiirellä haluamastaan väristä, väri haetaan luodusta väritaulukosta attribuutin avulla. Esimerkkikoodissa 4 mallille luodaan uusi materiaali THREE.MeshStandardMaterial, jolle annan haetun värin sekä mahdollisimman samankaltaiset ominaisuudet kuin sivustolle alun perin ladatulla mallilla. Materiaali asetetaan funktiolla setMaterial, jossa grafiikkakehyksen omalla metodilla traverse käydään malli läpi ja etsitään sen materiaali. Samassa funktiossa vaihdan sivuston värin valitun vaihtoehdon mukaisesti.

```
function selectColor(e) {
     let color = colorpalette[parseInt(e.target.dataset.key)];
     let new_material = new THREE.MeshStandardMaterial({
          color: parseInt("0x" + color.color),
          roughness: color.roughness ? color.roughness : 0,
          side: color.side ? color.side : 2,
     });

setMaterial(root, activeOption, new_material);
}

function setMaterial(parent, type, material) {
     parent.traverse((model) => {
          if (model.isMesh && model.name != null) {
               if (model.name == type) {
                    model.material = material;
                    scene.background = new THREE.Color(material.color);
               }
          }
     });
}
```

Esimerkkikoodi 4. Uuden materiaalin luominen ja vanhan materiaalin korvaaminen uudella.

Väripaletin yläpuolelle lisään kaksi painiketta, josta käyttäjä pystyy valitsemaan strösseleiden ja kuorrutteen välillä kumpaa mallin osaa halutaan muokata. Vaihtoehdoissa on klikkauksen kuuntelija, joka aktivoi funktion. Materiaalin asettava funktio ottaa materiaalin ja mallin lisäksi aktiivisen valinnan. Aktiivinen valinta asetetaan HTML-luokalla. Funktiossa poistetaan kyseinen luokka toiselta vaihtoehdolta ja lisätään sama luokka valitulle valinnalle.


Kuva 17. Donitsi ja värivaihtoehdot

Kuvassa 17 on valmis valikko väri- ja osavaihtoehdoilla. Olen asettanut oletuksena strösselit aktiiviseksi valinnaksi. Tämä tarkoittaa sitä, että jos käyttäjä avaa sivuston ja painaa suoraan jotain värivaihtoehtoa, strösseleiden väri vaihtuu valinnan mukaan.

# 5 Yhteenveto

Insinöörityössä luotiin 3D-malli donitsista Blender-mallinnustyökalulla, joka tuotiin verkkosivulle Three.js-kirjaston avulla. Yhtenä tutkimuksen tarkoituksena oli tarkastella näiden teknologioiden soveltuvuutta 3D-mallinnukseen verkkosivuilla sellaiselle, jolla ei ole aiempaa kokemusta. Mallin luominen oli odotettua yksinkertaisempaa. Mallinnustyökalulle on useita kolmannen osapuolen opetusvideoita ja oppaita. Myös Three.js soveltui hyvin aloittelijalle sen dokumentaation ja

helposti ymmärrettävyyden takia. Soveltuvuus kuitenkin riippuu lopulta siitä, millaisen mallin mallintaa ja mitä lisäominaisuuksia haluaa lisätä verkkosivulle. Käytin mallintamisessa tekniikoita, joiden käyttö olisi vaikeaa ilman apuja. Tästä esimerkkejä ovat geometria- ja varjostinsolmut.

Mallinnuksessa oli kuitenkin omat haasteensa. Ongelmia syntyi, kun toiminto oli monimutkaisempi esimerkiksi useiden varjostin- tai geometriasolmujen takia. Joissain tapauksissa toiminto vaati tiettyä tekniikkaa tai näppäinyhdistelmää. Esimerkiksi donitsin keskiosan muokkaaminen samanaikaisesti joka puolelta osoittautui hankalaksi, vaikka lopulta ratkaisu oli yksinkertainen.

Myös mallin näyttämisessä verkkosivulla oli ensin haasteita. Ensin sivusto näytti vain valkoisen tyhjän ruudun. Selaimen konsolista pystyttiin kuitenkin päättelemään, että kyse ei ollut vain siitä, että malli ei näkynyt, vaan koko Three.js-sovellus ei toiminut. Ongelma saatiin kuitenkin HTML-tiedoston muokkauksella, jossa sovellus yhdistettiin canvas-elementtiin. Myös luotujen strösseleiden saaminen sivustolle vaati tutkimustyötä. Ensimmäisellä kerralla strösseleitä ei näkynyt ollenkaan, vaikka strösselit oli generoitu mallinnustyökalussa. Tähän löydettiin kuitenkin ratkaisu Realize Instances -solmusta, joka muuttaa luodut strösselit geometriaksi.

Viimeisenä sivustolle lisättiin toiminnallisuutta luomalla käyttöliittymä mallin materiaalien värien muokkaamiseksi. Sivustoa voisi kuitenkin vielä kehittää useaan suuntaan. Yksi vaihtoehto voisi olla lisätä eri leivonnaisia, joita käyttäjä pystyy selaamaan nappia painamalla. Sivuston ulkoasua voisi myös kehittää riippuen siitä, mihin tarkoitukseen mallia halutaan käyttää. Lopputuloksena sivustosta tuli tyylikäs ja toimi toivotusti. Uskon, että muidenkin yksinkertaisten esineiden mallintaminen sekä esittäminen verkkosivulla onnistuisi suurelta osin itsenäisesti opittujen tekniikoiden avulla.

## Lähteet


1 Ball, Alex. 2013. Preserving Computer Aided Design (CAD). Verkkoaineisto. Digital Preservation Coalition. <https://www.dpconline.org/docs/technology-watch-reports/896-dpctw13-02-pdf/file>. Viitattu 10.12.2022.

2 3D Modeling. Verkkoaineisto. Siemens. <https://www.plm.automation.siemens.com/global/en/our-story/glossary/3d-modeling/17977>. Viitattu 10.12.2022.

3 “3D virtual reality models help yield better surgical outcomes: Innovative technology improves visualization of patient anatomy, study finds”. 2019. University of California – Los Angeles Health Sciences. Verkkoaineisto. ScienceDaily. <https://www.sciencedaily.com/releases/2019/09/190918131457.htm>. 18.9.2019. Viitattu 10.12.2022.

4 Price, Andrew. 2016. Photorealism Explained. Verkkoaineisto. <https://www.youtube.com/watch?v=R1-Ef54uTeU>. 25.5.2016. Viitattu 5.12.2022.

5 Ruudunkaappaus. Centogene. <https://solutions.centogene.com/>. Viitattu 12.10.2022.

6 Babich, Nick. 2020. 10 Best Parallax Website Design Examples. Verkkoaineisto. Adobe XD Ideas. <https://xd.adobe.com/ideas/principles/web-design/best-practices-for-parallax-websites/>. 1.13.2020. Viitattu 15.1.2023.

7 Ruudunkaappaus. Moooi. <https://www.moooi.com/en/a-life-extraordinary>. Viitattu 10.1.2023.

8 Ruudunkaappaus. Regnskogfondet. <https://rainforest.arkivert.no/#kart>. Viitattu 15.10.2022.

9 Costlow, Erik. 2021. The End of Applets. Verkkoaineisto. InfoQ. <https://www.infoq.com/news/2021/03/end-of-applets/>. 15.3.2021. Viitattu 11.1.2023.

10 Shankland, Stephen. 2013. It’s about time RuneScape dumps Java for HTML5. Verkkoaineisto. CNET. <https://www.cnet.com/tech/services-and-software/its-about-time-runescape-dumps-java-for-html5/>. 18.4.2013. Viitattu 10.1.2023.

11 Runefest’s big reveals. 2014. Verkkoaineisto. RuneScape. <https://secure.runescape.com/m=news/runefests-big-reveals>. 10.10.2014. Viitattu 10.1.2023.

12 Maya: Create expansive worlds, complex characters, and dazzling effects. Verkkoaineisto. Autodesk. <https://www.autodesk.com/products/maya/overview>. Viitattu 18.10.2022.

13 3ds Max: Create massive worlds and high-quality designs. Verkkoaineisto. Autodesk. <https://www.autodesk.com/products/3ds-max/overview>. Viitattu 18.10.2022.

14 Blender mallinnustyökalun kotisivut. Verkkoaineisto. Blender. <https://www.blender.org/>. Viitattu 18.10.2022.

15 Spline mallinnustyökalun kotisivut. Verkkoaineisto. Spline. <https://spline.design/>. Viitattu 18.10.2022.

16 Introduction to polygons. 2015. Verkkoaineisto. Autodesk. <https://knowledge.autodesk.com/support/maya-lt/learn-explore/caas/CloudHelp/cloudhelp/2015/ENU/MayaLT/files/Polygons-overview-Introduction-to-polygons-htm.html>. 23.1.2015. Viitattu 21.10.2022.

17 De Vries, Joey. Normal Mapping. Verkkoaineisto. LearnOpenGL. <https://learnopengl.com/Advanced-Lighting/Normal-Mapping>. Viitattu 21.10.2022.

18 Cignoni, Paolo. Kuva. <https://helpx.adobe.com/es/substance-3d-painter/using/baking.html>. Viitattu 21.10.2022.

19 Zeman, Nicholas Bernhardt. 2014. E-kirja. Essential Skills for 3D modeling, Rendering and Animation. A K Peters/CRC Press.

20 Kuva. Autodesk. <https://knowledge.autodesk.com/support/autocad/learn-explore/caas/CloudHelp/cloudhelp/2022/ENU/AutoCAD-Core/files/GUID-9EBC2243-B6BA-4A32-9300-8FAF15B2760F-htm.html>. Viitattu 4.1.2023.

21 Pintakuvioinnin määritelmä. Verkkoaineisto. TEPA-termipankki. <https://termipankki.fi/tepa/fi/haku/pintakuviointi>. Viitattu 5.1.2023.

22 Fundamentals. Verkkoaineisto. Three.js. <https://threejs.org/manual/#en/fundamentals>. Viitattu 5.11.2022.

23 Wright, Richard S.,Lipchak, Benjamin & Haemel, Nicholas. 2007. E-Kirja. OpenGL Superbible: Comprehensive Tutorial and Reference, Fourth Edition. Addison-Wesley Professional.

24 Low-level 3D Graphics API based on OpenGL ES. Khronos Group. <https://www.khronos.org/webgl/>. Viitattu 10.12.2022.

25 Matsuda, Kouichi & Lea, Rodger. 2013. E-Kirja. WebGL Programming Guide: Interactive 3D Graphics Programming with WebGL. Addison-Wesley Professional.

26 Rendering Pipeline Overview. Verkkoaineisto. Khronos Group. <https://www.khronos.org/opengl/wiki/Rendering_Pipeline_Overview>. Viitattu 15.12.2022.

27 Dirksen, Jos. 2015. E-kirja. Learning Three.js – the JavaScript 3D library for WebGL – Second Edition. Packt Publishing.

28 Ruudunkaappaus. Three.js. <https://threejs.org/manual/#en/primitives>. Viitattu 5.11.2022.

29 Dirksen, Jos. 2015. E-kirja. Three.js Cookbook. Packt Publishing.

30 @mrdoob (Three.js kehittäjä Ricardo Cabello). 2018. Twitter-päivitys. Verkkoaineisto. <https://twitter.com/mrdoob/status/1058022036038148096>. 1.11.2018 Viitattu 7.11.2022.

31 Creating a scene. Verkkoaineisto. Three.js. <https://threejs.org/docs/?q=webglrenderer#manual/en/introduction/Creating-a-scene>. Viitattu 7.11.2022.

32 Irwin, Emma. 2021. Microsoft Open Source success story—Babylon. Verkkoaineisto. Microsoft Open Source Blog. <https://cloudblogs.microsoft.com/opensource/2021/02/22/microsoft-open-source-success-story-babylon/>. 22.2.2021. Viitattu 16.12.2022.

33 Babylon.js kotisivut. Verkkoaineisto. Babylon.js <https://www.babylonjs.com/>. Viitattu 16.12.2022.

34 Using A Physics Engine. Babylon.js dokumentaatio. Babylon.js. <https://doc.babylonjs.com/features/featuresDeepDive/physics/usingPhysicsEngine>. Viitattu 17.12.2022.

35 Eastcott, Will & Nyman Robert. 2014. PlayCanvas Goes Open Source. Verkkoaineisto. <https://hacks.mozilla.org/2014/06/playcanvas-goes-open-source/>. 4.6.2014. Viitattu 19.12.2022.

36 Editor. Verkkoaineisto. PlayCanvas. <https://developer.playcanvas.com/en/user-manual/designer/>. Viitattu 4.1.2023.

37 Ruudunkaappaus. PlayCanvas. <https://playcanvas.com/features>. Viitattu 4.1.2023.

38 Entity. Verkkoaineisto. PlayCanvas. <https://developer.playcanvas.com/api/pc.Entity.html>. Viitattu 19.12.2022.

39 Making A Simple Game – Part 1. Verkkoaineisto. PlayCanvas. <https://developer.playcanvas.com/en/tutorials/keepyup-part-one/#model-component>. Viitattu 19.12.2022.

40 Ruudunkaappaus. PlayCanvas. <https://developer.playcanvas.com/en/user-manual/packs/components/sound/> Viitattu 4.1.2023.

41 Scripting. Verkkoaineisto. PlayCanvas. <https://developer.playcanvas.com/en/user-manual/scripting/> Viitattu 19.1.2023.

42 Subdivision Surface Modifier. Verkkoaineisto. Blender. <https://docs.blender.org/manual/en/latest/modeling/modifiers/generate/subdivision_surface.html>. Viitattu 10.11.2022.

43 Shrinkwrap Modifier. Verkkoaineisto. Blender. <https://docs.blender.org/manual/en/2.79/modeling/modifiers/deform/shrinkwrap.html>. Viitattu 10.11.2022.

44 Principled BSDF. Verkkoaineisto. Blender. <https://docs.blender.org/manual/en/latest/render/shader_nodes/shader/principled.html>. Viitattu 11.11.2022.

45 Distribute Points on Faces. Verkkoaineisto. Blender. <https://docs.blender.org/manual/en/latest/modeling/geometry_nodes/point/distribute_points_on_faces.html>. Viitattu 20.11.2022.

46 Instance on Points Node. Verkkoaineisto. Blender. <https://docs.blender.org/manual/en/latest/modeling/geometry_nodes/instances/instance_on_points.html>. Viitattu 20.11.2022.

47 glTF 2.0. Verkkoaineisto. Blender. <https://docs.blender.org/manual/en/2.80/addons/io_scene_gltf2.html>. Viitattu 23.11.2022.

48 Realize Instances Node. Verkkoaineisto. Blender. <https://docs.blender.org/manual/en/latest/modeling/geometry_nodes/instances/realize_instances.html>. Viitattu 5.1.2023.

49 Loading a .GLTF File. Verkkoaineisto. Three.js. <https://threejs.org/manual/#en/load-gltf> Viitattu 5.11.2022.

50 Wetton, Kyle. 2019. How To Build A Color Customizer App for a 3D Model with Three.js. Verkkoaineisto. Codrops. <https://tympanus.net/codrops/2019/09/17/how-to-build-a-color-customizer-app-for-a-3d-model-with-three-js/>. 17.9.2019. Viitattu 12.1.2023.