**Matti Suorsa**

# GRAFIIKAN ESITTÄMINEN HTML5:N AVULLA

## Esimerkkinä Three.js

**Opinnäytetyö**
**CENTRIA AMMATTIKORKEAKOULU**
**Mediatekniikan koulutusohjelma**
**Toukokuu 2015**

**TIIVISTELMÄ OPINNÄYTETYÖSTÄ**

| **Yksikkö**<br>Ylivieska | **Aika**<br>Toukokuu 2015 | **Tekijä**<br>Matti Suorsa |
|---|---|---|
| **Koulutusohjelma**<br>Mediatekniikan koulutusohjelma | | |
| **Työn nimi**<br>GRAFIIKAN ESITTÄMINEN HTML5:N AVULLA<br>Esimerkkinä Three.js | | |
| **Työn ohjaaja**<br>Hannu Puomio | | **Sivumäärä**<br>33 + 4 |
| **Työelämäohjaaja** | | |


Opinnäytetyön aiheena oli tutkia HTML5:n mahdollisuuksia ja vaatimuksia selaimessa esitettävän grafiikan osalta.

Tavoitteena oli löytää ja valita sopivat menetelmät interaktiivisen 3D-grafiikan luomiseen ja esittämiseen selaimessa. Valittujen menetelmien avulla toteutettiin useampi toimiva web-sovellus eri tarkoituksiin.

Käytettyihin menetelmiin kuuluivat Three.js JavaScript-kirjasto ohjelmistokehyksenä ja WebGL API rendaajana. Opinnäytetyössä käydään läpi vaiheet käytettyjen menetelmien osalta interaktiivisten sovellusten ja niiden grafiikan luomisesta.



**Asiasanat**
3D, Canvas, Grafiikka, HTML5, JavaScript, Selain, Three.js, WebGL

Centria
UNIVERSITY OF APPLIED SCIENCES

## ABSTRACT


| Unit | Date | Author |
|---|---|---|
| Ylivieska | May 2015 | Matti Suorsa |


| |
|---|
| **Degree programme**<br>Media Technology |
| **Name of thesis**<br>PRODUCING GRAPHICS WITH HTML5<br>Three.js as an example |

| Instructor | Pages |
|---|---|
| Hannu Puomio | 33 + 4 |

**Supervisor**


The subject of this thesis was to research the possibilities and requirements of HTML5 in terms of displaying interactive graphics in a browser.

The aim was to find and choose the most convenient methods for showing and producing 3D-graphics in a browser. The chosen methods were used to create multiple working web-applications for different purposes.

Methods used in producing these applications were Three.js JavaScript-library as a framework and WebGL API as a renderer. The process of creating an interactive application and its graphics is explained in the thesis regarding the chosen methods.



**Key words**
3D, Browser, Canvas, Graphics, HTML5, JavaScript, Three.js, WebGL

## KÄSITTEIDEN MÄÄRITTELY

| | |
|---|---|
| API | Application Programming Interface = Ohjelmointirajapinta. helpottaa sovellusten kehittämisessä |
| DOM | Document Object Model. Ohjelmointirajapinta, joka mahdollistaa verkkosivun elementtien muokkaamisen |
| export | Ulosvienti. Tiedosto tallennetaan eri muotoon ohjelmasta toiseen vietäväksi |
| HTML5 | Hypertext Markup Language. Selaimessa käytettävän sivun kuvauskielen viimeisin versio |
| JavaScript | Web-ympäristössä käytettävä komentosarjakieli |
| JavaScript-kirjasto | Esi-kirjoitettu JavaScript-dokumentti. Helpottaa JavaScript -pohjaisten sovellusten kehittämistä |
| mesh | 3D-mallinnuksessa vertexeistä muodostuva monitahkoinen 3D-objekti |
| modifier | 3D-mallinnuksessa mallille lisättävä modifikaattori |
| objekti | Esine 3D-avaruudessa |
| renderöinti | Tietokoneessa matemaattisen 3D-grafiikan piirtäminen 2D-kuvaksi ruudulle |
| Three.js | JavaScript-kirjasto 3D-applikaatioiden luomiseen |
| webGL | API 3D-grafiikan esittämiseen selaimessa |

**TAULUKOT**

# 1 JOHDANTO

Opinnäytetyössäni tutkimuksen kohteena on verkkoselaimessa HTML5:n avulla näytettävän grafiikan esittämistavat ja luominen yleisellä tasolla. Perehdyn tarkemmin HTML5:n mahdollisuuksiin kolmiulotteisen grafiikan osalta web-sovellusten luomisessa.

Ensimmäinen luku käsittelee tutkimustyötä tavoista ja tekniikoista soveltaa interaktiivista grafiikkaa selaimessa. Toisessa luvussa siirryn käsittelemään 3D-grafiikkaa, ja sen nykymahdollisuuksia selainkäytössä ohjelmointirajapintoja hyödyntämällä. Käyn myös läpi, kuinka valmiita JavaScript-kirjastoja voidaan käyttää hyväksi 3D-sovellusten luomisessa. Kolmannessa luvussa käyn läpi selaimeen tuotettavan 3D-grafiikan kehitysvaiheet. Esittelen myös menetelmiä interaktiivisen sisällön luomiseen 3D-sovellukselle sekä tapoja parantaa sen visuaalisuuksia. Lopuksi käyn läpi tulokset tekemieni sovellusten osalta ja pohdin myös käyttämieni tekniikoiden käyttömahdollisuuksia sekä vahvuuksia.

Sain opinnäytetyölleni aiheen kesällä 2014 työstäessäni kesäprojektia Centria-ammattikorkeakoululla. Projektiin kuului web-ohjelmistoprototyypin kehittäminen paikalliselle yritykselle. Ohjelmiston kehittämisessä käyttämäni tekniikat saivat minut kiinnostumaan aiheesta ja tartuinkin siihen tämän opinnäytetyön muodossa.

Opinnäytetyöni aihe vaati taitoja sekä 2D- että 3D-grafiikan luomisen osalta, www-ohjelmointitaitoja sekä ymmärrystä ohjelmointirajapintojen soveltamisesta. Tärkeää oli myös osata ottaa selvää vaihtoehtoisista tekniikoista sekä vertailla niitä ja niiden käyttötarkoituksia.

Päälähteinä työssäni käytin www-dokumentteja, Three.js:n kotisivuilta löytyvää dokumentaatiota ja esimerkkejä. Käytössäni oli myös Jos Dirksen Learning Three.js: The JavaScript 3D Library for WebGL -kirja.

# 2 GRAFIIKAN ESITTÄMISTAVAT SELAIMESSA

Tässä luvussa käsittelen suosituimmat tavat esittää interaktiivista grafiikkaa selaimessa: HTML5:n ja laajennukset, eli pluginit. Vaikka opinnäytetyöni käsittelee HTML5:sta grafiikan esittämisessä, oli hyvä tarkastella muitakin grafiikan esittämiseen soveltuvia tekniikoita.

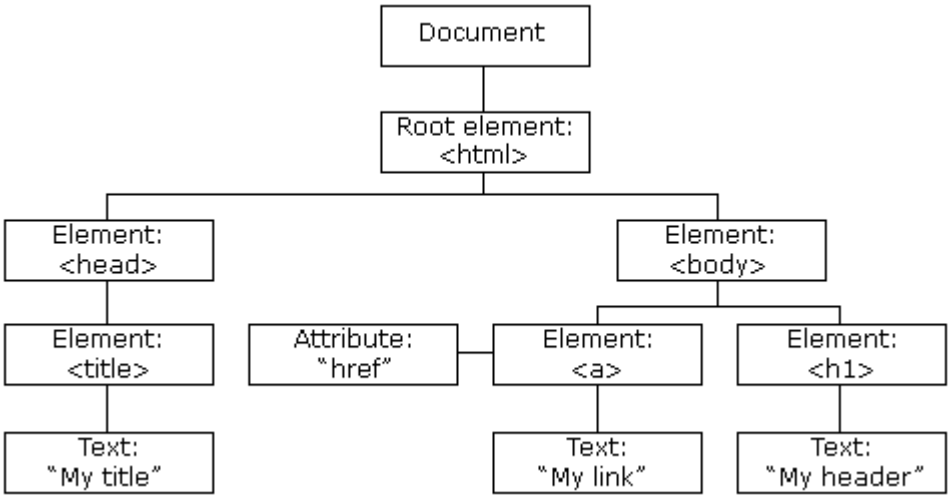
## 2.1 HTML

Interaktiivista grafiikkaa suunniteltaessa alkuun oli hyvä saada ymmärrys HTML-dokumentin sekä muiden, siihen vaikuttavien dokumenttien rakenteesta ja toiminnoista.

HTML on standardi kuvauskieli verkkosivujen kehittämiseen ja esittämiseen. HTML-kieli kuvastaa web-dokumentin rakennetta elementein (KUVIO1). Elementit voivat sisältää esimerkiksi mediaa, kuten kuvia tai tekstiä. Jokaisella elementillä on oma ”tagi”, joka kertoo elementin tyypin. (Mozilla Foundation. 2015)

tagi esimerkki kuvalle: `<img src=″/kuvat/kuva.jpg″ />`

Käyttäjän näkemä verkkosivu rakentuu useammista eri teknologioista. Cascading Style Sheets (CSS), on tyylidokumentti, jolla voidaan määrätä kuinka jokin tietty elementti piirretään näytölle. Esimerkkinä, tyylidokumentin kautta voidaan määrittää elementin mitat, sijainti tai sen taustan väri. Javascript on nykyään tärkeässä roolissa verkkosivujen luomiseen käytettävistä teknologioista. Se on dynaaminen, oliopohjainen komentosarjakieli, jolla voidaan luoda interaktiivisuutta HTML-dokumentissa oleville elementeille. JavaScriptin ansiosta selaimelle voidaan luoda myös monenlaisia verkkosovelluksia, jopa pelejä. (Mozilla Foundation. 2015)




KUVIO 1. DOM rakenne HTML-dokumentissa (W3schools 2015.)

DOM, eli Document Object Model, on HTML ja XML dokumentille kehitetty ohjelmointirajapinta. DOM luo dokumentille puumaisen rakenteen (KUVIO1), joka mahdollistaa dokumentin eri elementtien sisällön ja tyylin käsittelemisen, esimerkiksi JavaScriptin tai tyylidokumentin kautta. Se on täysin olio-pohjainen esitys web-sivusta, joka toimii välikappaleena web-sivun ja ohjelmointi- tai komentosarjakielen välillä. (Mozilla Foundation 2015.)

Esimerkki kuinka tiettyyn elementtiin päästään käsiksi DOMia hyödyntäen:

HTML: `<div id=”ThreeJS”></div>`
CSS: `#ThreeJS { position: absolute; }`
JavaScript: `var container = document.getElementById('ThreeJS');`

HTML:n tai JavaScriptin kirjoittamisessa ei tarvittu erityisiä ohjelmia; tekstityökalut, kuten Windows-käyttöjärjestelmästä löytyvä NotePad, tai MAC-käyttöjärjestelmästä löytyvä TextEdit soveltuvat www-ohjelmointiin. Tavallinen tekstieditori siis riitti, mutta ohjelmointiin suunnitellun editorin käyttäminen oli kuitenkin suositeltavaa. Esimerkiksi rivinumerot, oikeinkirjoituksen tarkistus ja välilehdet tekevät koodin kirjoittamisesta paljon helpompaa. Hyviä vaihtoehtoja olivat Notepad++, joka on ilmainen, sekä Adobe Dreamweaver, joka taas on maksullinen.

HTML5 on HTML:n uusin versio, joka on tuonut mukanaan tagit videon ja äänen esittämiselle, sekä paremman tuen interaktiivisen grafiikan luomiselle ja näyttämiselle. HTML5 on ollut käytössä jo vuodesta 2011, mutta virallinen versio julkaistiin vasta lokakuussa 2014.

HTML5:n mukana tullut <canvas> elementti mahdollistaa 2d-grafiikan ja animaation rendaamisen selaimessa reaaliaikaisesti. Koska canvas on integroituna HTML5:een, voidaan sen avulla upottaa grafiikkaa ja animaatiota suoraan verkkosivuille interaktiivisina elementteinä. Canvas itsessään on siis vain kehys esitettävälle grafiikalle; itse grafiikan piirtämiseen vaaditaan JavaScript komentoja. (Pilgrim 2015.)

Canvas elementtiin pääsee käsiksi DOM:issa samalla tavoin kuin muihinkin elementteihin. Itse grafiikan luominen tapahtuu piirtämiskomennoin: esimerkiksi viivaa piirrettäessä sille annetaan aloitus ja lopetus sijainnit x ja y-koordinaateilla. Kuvia käytettäessä voidaan hyödyntää valmista image -elementtiä, tai voidaan luoda kuvaobjekti suoraan JavaScriptista. Myös kuva sijoitetaan canvasiin koordinaatein, ja sille annetaan mittasuhteet. Canvasin objekteille voidaan luoda interaktiivisuutta esimerkiksi ”mouseover” ja ”onclick” tapahtumilla (event). Canvasiin voidaan myös sijoittaa esimerkiksi tekstiä ja liukuvärejä. (Pilgrim 2015.)

## 2.2 Laajennukset grafiikan esittämisessä

Ennen HTML5:n yleistymistä laajennuksilla eli plugineilla on ollut varsin suuri rooli selaimessa esitettävän grafiikan esittämisessä, sillä aiempien HTML-versioiden mahdollisuudet olivat hyvin rajoittuneet. Laajennus on erilliseen ohjelmistoon asennettava lisä/liitännäinen, joka mahdollistaa sillä lisäominaisuuksien käyttämisen. HTML:ssä plugini lisätään dokumenttiin <object> -elementillä.

### 2.2.1 Flash -laajennus

Flash on Macromedian kehittämä ohjelma, jolla voidaan tuottaa vektorigrafiikka-pohjaista animaatiota ja kuvaa. Se on tunnettu tiedostoformaatistaan, joka mahdollistaa interaktiivisuuden ja skaalaantuvuuden, sekä pienet tiedoston koot. Flash-animaation katsominen selaimessa vaatii Flash Player –laajennuksen asentamisen. (Rouse 2005.)

Flash Player HTML-dokumentin elementissä:

```
<object width="800" height="500" data="animation.swf"></object>
```

Adobe Flash -kehitysympäristö tarjoaa graafisen käyttöliittymän animaation luomiseen. Ohjelmaan sisältyy monenlaisia ominaisuuksia ja työkaluja, joista yksi tärkeimpiä on sen ominainen aikajana, jonka avulla on helppo luoda animaatiota. Käytössä on myös Flashin oma komentosarjakieli ”ActionScript”. JavaScriptin tapaan ActionScript on oliopohjainen kieli, joka on suunnattu web-sovellusten kehittämistä varten.

Flashin uusin kehitysversio kykenee julkaisemaan grafiikkaa ja animaatiota myös HTML5:n Canvas elementissä WebGL:n avulla, sekä joissakin mukautetuissa käyttöympäristöissä. Pluginin käyttäminen ei siis ole enään välttämätöntä flashilla tuotetun grafiikan esittämisessä. Flashin julkaiseminen canvasissa kuitenkin vaatii tiettyjen ominaisuuksien, kuten filtterien karsimista (Labrecque 2015.). Adobe on lopettanut Flashin tuen mobiililaitteille, ja sen käyttö verkkosivuilla on myös vähentynyt lähivuosina huomattavasti. Mahdollisuus julkaista Flashia Canvas-elementissä on siis positiivinen uutinen sen myös ollessa uusi tapa julkaista Flash-materiaalia mobiililaitteille.

### 2.2.2 Unity –laajennus

Unity Technologiesin kehittämä Unity3D-pelimoottori ei ole ehkä laajennuksena tutuimpia verkossa tavatuista vaihtoehdoista. Päätin kuitenkin tutkia sitä sen merkityksellisyyden johdosta pelinkehityksessä. Unityn uusin versio tarjoaa myös uusia mahdollisuuksista selaimessa.

Unity on joustava kehitysympäristö 2D- ja 3D -peleille sekä interaktiivisille kokeiluille. Unityn tuki eri alustoille on laaja. Tuettuihin alustoihin kuuluvat mm. Android, Windows, iOS, PS4 ja Xbox One. Uusimpina alustoina ovat tulleet WebGL sekä Oculus Rift. Unity on lähivuosina noussut suosituimmaksi pelimoottoriksi pelialalla. Myös suurin osa mobiilipeleistä kehitetään nykyään sen avulla. (Unity. 2015)

Koodin kirjoittamiseen Unity tarjoaa C#, UnityScript ja Boo –kielet. C# on oliopohjaisena ohjelmointikielenä tuttu, ja yleensä mieluisin vaihtoehto pelinkehittäjille. UnityScript taas on Unityn oma komentosarja kieli, joka muistuttaa hieman JavaScriptia.

Uusimman version, eli Unity 5.0 mukana on tullut tuki WebGL:lle, eli tapa julkaista 3D-grafiikkaa selaimessa ilman pluginia. Kyseessä ei kuitenkaan ole valmis versio, vaan ”early-access add-on”. WebGL-tuki on siis vasta kokeiluvaiheessa Unityssa, ja valmis versio tuleekin olemaan todennäköisesti maksullinen. Tuettuja selaimia ovat Firefox ja Chrome.

## 2.3 Vertailua

Vertailin vaihtoehtoja tavoista luoda interaktiivista grafiikkaa selaimeen (TAULUKKO1). Vertailtaviksi otin HTML5:n lisäksi Adobe Flashin ja Unityn kehitysympäristöt niiden tarjoamien selain-laajennusten osalta.

TAULUKKO 1. Vertailua: HTML5, Flash ja Unity

| | **HTML5** | **ADOBE FLASH** | **UNITY** |
|---|---|---|---|
| **Avaaminen selaimessa** | Ei vaatimuksia | Flash Player –plugin | Unity Player -plugin |
| **Sovelluksen kehittäminen** | Ei vaatimuksia, API:t 3D-grafiikalle | Oma kehitysympäristö | Oma kehitysympäristö |
| **Scriptikieli** | JavaScript | ActionScript, Pixel Bender | c#, UnityScript, Boo |
| **Sovelluksen hinta** | | Virallinen maksaa, myös ilmaisia versioita | Ilmainen ja maksullinen versio |
| **Tuki mobiiliselaimelle** | Hyvä | Tuki lopetettu, vain WebGL | Ei vielä |
| **WebGL -tuki** | Hyvä | Pieniä puutteita | Kokeiluvaiheessa |

Unity tarjoaa ilmaisen ympäristön pelien ja sovellusten kehittämiselle. Sen mahdollisuudet tuottaa 2D- ja 3D -grafiikkaa ovat hyvät. Unity on suositeltava vaihtoehto, jos tarkoituksena on luoda monimutkaisempi sovellus. Sen sisältämän WebGL:n ollessa kokeiluvaiheessa, on plugini edelleen varmempi vaihtoehto selaimessa julkaisemiseen.

Flash on edelleen helppo tapa tuottaa 2D-animaatiota selainkäyttöön. Sen tuki WebGL:lle tarjoaa uusia mahdollisuuksia. Jos Flash on tuttu, omistaa sovelluksen ja ActionScript sujuu, on se edelleen varteenotettava vaihtoehto.

Valintani, eli HTML5 tarjoaa helpon tavan tuottaa grafiikkaa työpöytä- ja mobiiliselaimeen, mutta ei tarjoa sen luomiseen graafista käyttöliittymää. 3D -sovelluksen luominen vaatisi ohjelmointirajapintojen hyödyntämistä sekä riittävät JavaScript-taidot. Unity olisi myös hyvä vaihtoehto WebGL -tuen kanssa, mutta en uskonut sen olevan paras vaihtoehto käyttötarkoitukseeni.

# 3 3D-GRAFIIKKA

Kolmiulotteista grafiikkaa sisältävää sovellusta kehitettäessä oli hyvä ymmärtää mitä 3D todella tarkoittaa. 3D-grafiikan luominen ja sijoittaminen kolmiulotteisessa avaruudessa oli helpompaa, sekä valmista materiaalia tutkittaessa oli helpompi ymmärtää mitä sovelluksessa todella tapahtui.

3D-grafiikka on grafiikkaa, joka käyttää kolmiulotteista esittämistapaa tietokoneelle tallennetusta geometrisesta tiedosta. 3D-grafiikan piirtämistä voi verrata esimerkiksi canvas-elementin xy-koordinaateilla piirtämiseen, mutta 3D-grafiikassa käytetään myös z-koordinaattia, eli syvyyttä. (Parisi 2014.)

3D-grafiikan piirtämiseen on useampia tapoja, mutta suosituin tapa on käyttää *mesh*iä. Mesh on objekti, joka muodostuu yhdestä tai useammasta polygon muodosta, jotka taas muodostuvat *vertice*istä omilla xyz-koordinaateillaan. Käytettävät polygonit muodostuvat yleensä kolmesta tai neljästä verticestä. kolmiulotteista meshiä kutsutaan usein myös 3D-malliksi. (Parisi 2014.)

## 3.1 Renderöinti

3D-tuotannossa rendereröinti on teknisesti monimutkaisin vaihe. Esimerkkinä 3D-grafiikan renderöimistä voidaan verrata filmikuvaukseen: valokuvaaja joutuu kehittämään ja printtaamaan ottamansa valokuvat ennenkuin hän voi tarkastella lopullista kuvaa. 3D-mallia tai sceneä luodessaan kehittäjä näkee ruudulla vain matemaattisen esityksen mallin vertexeistä ja polygoneista kolmiulotteisessa tilassa. Renderöinti on siis toimenpide, jossa rendaaja muuttaa matemaattiset arvot viimeistellyksi 2D-kuvaksi. Renderöinnin aikana renderöijä laskee ja yhdistää scenen informaation valoisuudesta, tekstuureista ja väreistä jokaista pixeliä kohden lopullista kuvaa varten. (Slick 2015.)

Renderöinnin voi jakaa esi- ja reaaliaikaiseen renderöintiin. Esirenderöintiä käytetään, kun rendausaika ei ole ongelmana. Kaikki visuaalisesti

monimutkaisimmat tuotteet, kuten animaatiot ja erikoisefektit, esirenderöidään, sillä niissä käytetään enemmän polygoneja ja tekstuurien resoluutiot ovat suurempia. Esirenderöinti suoritetaan yleensä prosessorilla. Reaaliaikaisessa renderöinnissä rendausaika on hyvin lyhyt. Sitä käytetään interaktiivisen grafiikan esittämisessä, kuten peleissä, joissa 3D-informaatio täytyy muuttaa kuviksi hyvin nopeasti. Vähintään 18-20 kuvaa sekunnissa on suositeltava määrä sovelluksen käytettävyyden säilyttämiseksi. Reaaliaikainen renderöinti suoritetaan yleensä näytönohjaimella. Näytönohjaimen rasitusta vähennetään esikoostetulla informaatiolla, kuten tekstuuritiedostoiksi tallennetulla ympäristön valaistus kartoilla. (Slick 2015.)

### 3.2 3D HTML5:n avulla

Interaktiivisen 3D-grafiikan esittämiseen selaimessa pystyvät ainakin HTML5 ohjelmointirajapintoja hyödyntäen, CSS3 sekä pluginit, kuten Unity Player. HTML5 vaatii ohjelmointirajapintojen ja JavaScript kirjastojen hyödyntämistä 3D-grafiikan esittämiseen. Päätin ottaa selvää kuinka ne toimivat, ja kuinka niitä käytetään.

Kolmiulotteisen grafiikan ja animaation hyödyntäminen selaimessa Internet-käyttöä varten on ollut aiemmin vähäistä. Hitaat Internet-siirtonopeudet ja tietokoneen tietojenkäsittelynopeudet ovat olleet ongelmana 3D-grafiikkaa sisältävien sovellusten runsaiden datan käyttö vaatimusten johdosta. Viime vuosina siirtonopeudet ovat moninkertaistuneet, ja tavalliset peruskäyttöön tarkoitetut tietokoneet riittävät 3D-grafiikan pyörittämiseen riittävän hyvin. Tämän johdosta suosituimmat selaimet ovat alkaneet hyödyntämään WebGL ja CSS3 teknologioita, joiden avulla 3D-grafiikan pyörittäminen selaimessa onnistuu ilman plugineita. Uusimmat selaimet kykenevät myös laitteistokiihdytys –teknologiaan, jonka avulla selaimessa grafiikan esittämiseen voidaan hyödyntää tietokoneen näytönohjainta. (Google 2010.)

### 3.2.1 Ohjelmointirajapinnat

Ohjelmointirajapinta (API = Application Program interface) on sarja funktioita, protokollia ja työkaluja sovelluksen luomiseen. Ohjelmointirajapinta määrittelee, kuinka ohjelman komponentit ovat vuorovaikutuksessa toisiinsa. Niitä käytetään, kun kehitetään graafinen käyttöliittymä sovellukselle. Ohjelmointirajapinnan tulisi koostua selkeistä ”rakennuspalikoista”, joita ohjelmoija voi käyttää oman ohjelmansa rakentamiseen. (Beal 2015.)

Itse HTML5:n mahdollisuudet antoisan grafiikan luomiseen ovat hyvät, mutta todelliseen potentiaaliin päästään vasta ohjelmointirajapintoja hyödyntämällä. On kuitenkin hyvä muistaa, että vasta HTML5:n nykyisen version myötä tulleet animaation päivittämisen tekniikka, ja kehittyneet suorituskyvyt, ovat olleet välttämättömyys näiden ohjelmointirajapintojen luomiselle. (Parisi 2014.)

### 3.2.2 WebGL

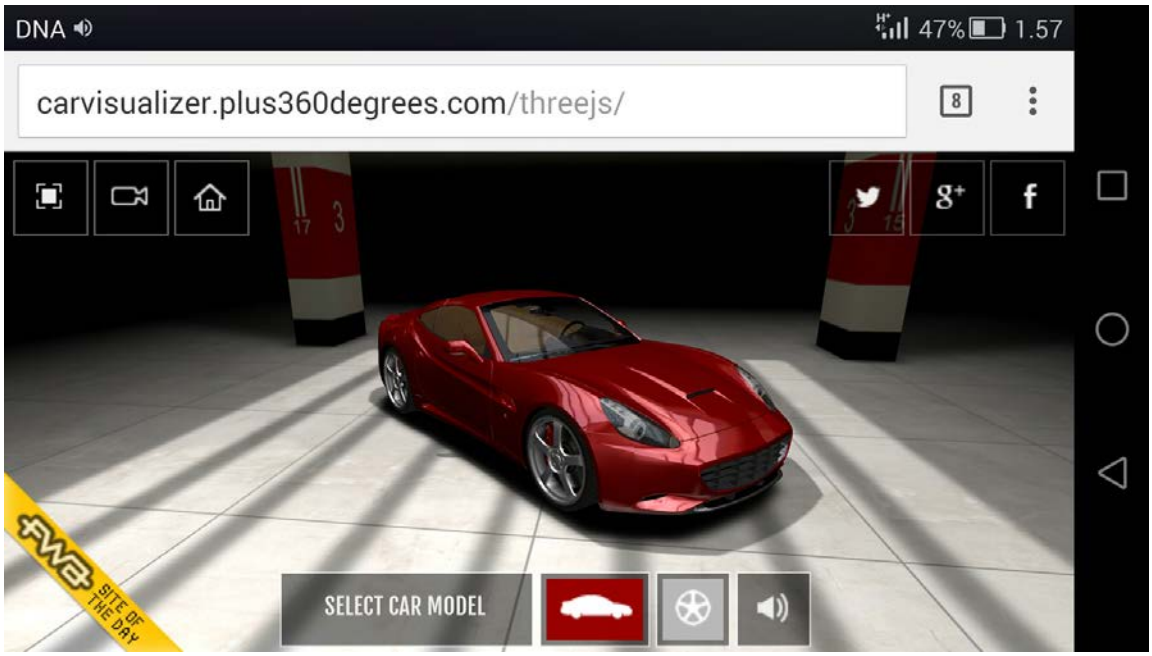
Graafisissa testeissäni hyödynsin WebGL-ohjelmointirajapintaa 3D-grafiikan esittämiseen. WebGL:n todellinen merkitys projektissani olisi voinut jäädä helposti pienemmälle huomiolle, sillä minun ei tarvinnut kehittää mitään sen osalta, eikä sen kannalta ilmennyt ongelmiakaan. Päätin kuitenkin ottaa selvää sen toimintaperiaatteista.

WebGL on tekijänoikeusvapaa ohjelmointirajapinta 3D-grafiikan esittämiseen selaimessa. Se käyttää HTML5:n canvas -elementtiä ja on täysin integroitu kaikkiin DOM-rajapintoihin. WebGL:n ollessa DOM-ohjelmointirajapinta, sitä voidaan käyttää kaikissa DOM-yhteensopivissa ohjelmointikielissä, kuten JavaScriptissä ja Javassa. WebGL perustuu yleisesti käytettyyn 3D-grafiikka standardiin, OpenGL:ään, joten se on yhteensopiva kaikissa suosituimmissa selaimissa. WebGL on myös hyvin integroitu HTML-sisältöön, kuten tasojen rakenteeseen, muiden HTML-elemettien kanssa vuorovaikuttamiseen ja HTML:n standardiin tapahtumien hallintamekanismin (eng. event handling mechanism) käyttämiseen. WebGL tukee laitteistokiihdytystä 3D-grafiikan esittämisessä.

Koodausympäristönä WebGL on hyvin haastava: WebGL lasketaan ”low-level” API:ksi, joten yksinkertaisenkin asian koodaaminen voi viedä runsaasti aikaa. (Khronos 2010.)

WebGL:ään ohjelmiston kehittäjä pääsee käsiksi vain tiettyjen JavaScript -ohjelmointirajapintojen avulla; WebGL ei käytä tageja, kuten HTML. (Parisi, T. 2014).

Suosituimmat työpöytä- ja mobiiliselaimet ovat saaneet WebGL -tuen (KUVIO 3). Tarvittaessa WebGL:n tuen laitteelle tai selaimelle pystyi tarkistamaan sille suunnitellulta verkkosivulta: *https://get.webgl.org/*




KUVIO3. WebGL-sovellus mobiiliselaimessa

### 3.3 JavaScript-kirjastot

JavaScript-kirjastot, eli valmiiksi kirjoitetut JavaScript-koodit helpottavat ja nopeuttavat skriptausprosessia, joka normaalisti veisi runsaasti aikaa. Tietyt WebGL:ään perustuvat kirjastot mahdollistavat 3D-sovellusten kehittämisen, vaikka kokemus ja tietämys 3D-maailmasta olisikin vähäistä. Myös kokeneemmat

kehittäjät voivat hyödyntää niitä säästääkseen aikaansa skriptauksen osalta. (Parisi 2014.)

### 3.3.1 Vaihtoehdot

JavaScript-kirjastoja oli kehitetty jo useita hyödyntämään WebGL:ää. Vertailin joitakin suosituimpia kirjastoja löytääkseni sopivimman tarkoitukseeni. Vertailun kohteiksi valitsin Three.js, Babylon.js ja PlayCanvas –kirjastot.

TAULUKKO 2. JavaScript -kirjastojen vertailu

| | **Three.js** | **babylon.js** | **PlayCanvas** |
|---|---|---|---|
| **Rendaaja** | WebGL,Canvas,SVG | WebGL,Canvas,SVG | WebGL |
| **tuki 3D-mallien formaateille** | .obj. FBX, Blender | .obj, .FBX, .MXB, Blender | ainakin FBX, COLLADA |
| **Lisenssi** | Open Source | Open Source | ilmainen ja maksullinen versio |
| **Fysiikat** | ei | löytyy | löytyy |
| **Julkaisu** | 2009 | 2013 | 2014 |

Babylon.js on pelinkehitykseen soveltuva kirjasto. Yleisten ominaisuuksien lisäksi se myös sisältää tekniikat törmäyksen havaitsemiselle ja antialisoinnille. (Hewitson 2013). Babylon.js vaikuttaa olevan hyvin tehokas kirjasto, joka soveltuu erityisesti isompien projektien tekemiseen.

Three.js pyrkii hyödyntämään kaikki selaimessa käytettävien rendaajien edut. Se tarjoaa laajat mahdollisuudet grafiikan ja animaation luomiselle. Sen ideana ei ole erikoistua mihinkään tiettyyn sovellustyyppiin. (Hewitson, J. 2013)

Playcanvas vaikutti sisältävän suurimman määrän toimintoja. Playcanvas kirjastolla kehittämien vaatii palveluun kirjautumisen, jossa myös sovelluksen kehittäminen tapahtui. Playcanvas vaikutti olevan myös pelinkehitykseen keskittyvä kirjasto.

Valitsin käyttööni Three.js –kirjaston. Se oli helppokäyttöinen, omiin pieniin projekteihin ja testeihin soveltuva kirjasto. Sitä voidaan käyttää kaikenlaisten interaktiivisten sovellusten kehittämiseen. Omalle verkkosivuille integrointi oli myös hyvin helppoa. Three.js sopi aiheesta kiinnostuneelle ensikertalaiselle: avuksi löytyi paljon esimerkkejä.

### 3.3.2 Three.js

Three.js on avoimeen lähdekoodiin perustuva JavaScript-kirjasto, joka on luotu helpottamaan 3D-grafiikan tuomista selaimeen. Sen käyttäminen vaatii tuntemusta 3D:hen ja JavaScriptin kirjoittamiseen. Three.js:n kehittäjä, Ricardo Cabello, joka on tunnettu myös nimimerkillä ”mrdoob”, kirjoittaa:

*“The aim of the project is to create a lightweight 3D library with a very low level of complexity — in other words, for dummies.”* (Cabello 2015.)

Three.js:n käyttö oli merkittävässä roolissa opinnäytetyöni käytännön osuudessa. Mukaan kuului paljon JavaScriptin kirjoittamista. Aikaa kului runsaasti myös tiedon hakemiseen: ”takaisinmallinnus”, eli valmiiden esimerkkien toiminnallisuuden selvittäminen oli usein välttämätöntä riittävän ymmärryksen saamiseen tietyistä toiminnoista.

Three.js JavaScript-kirjastoa otettaessa käyttöön, tulee kehittäjän ladata lähdekoodi tietokoneelleen. Lähdekoodin uusimman version löytää Three.js:n kotisivuilta, tai GitHub –palvelusta:
*https://github.com/mrdoob/three.js/*

Three.js:n sisällöstä oleellisin päätiedosto löytyi ’build’ -kansiosta. Valittavissa oli versio, joka sisälsi kommentit, ja minimaalinen versio, joka oli kutistettu mahdollisimman tilaa säästäväksi. Jos haluttiin tehdä muutoksia itse lähdekoodiin, kannatti käyttää kommentillista versiota. Tiedosto lisättiin HTML-dokumenttiin <script> tagilla. Näin kaikki kirjaston sisältö oli käytettävissä DOM:in avulla omasta JavaScript-dokumentista käsin. Muut valmiit komponentit, kuten kontrollit tai shaderit, eivät sisältyneet päätiedostoon, joten ne jouduttiin lisäämään erikseen. Kirjaston avulla käytettävät valmiit komennot löytyivät Three.js:n omasta dokumentoinnista, Three.js:n verkkosivuilta, tai ladatusta Three.js:n kansiosta. Dokumentaatio oli osittain vielä keskeneräinen, joten tietoa täytyi etsiä muistakin lähteistä. Esitetyn tiedon julkaisemisen päivämäärä oli hyvä tarkistaa, sillä vanhempien versioiden komennoissa ilmeni eroavaisuuksia nykyiseen versioon.

Tavalliseen HTML-dokumenttiin luotiin viittaus käytettävistä three.js:n komponenteista:

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

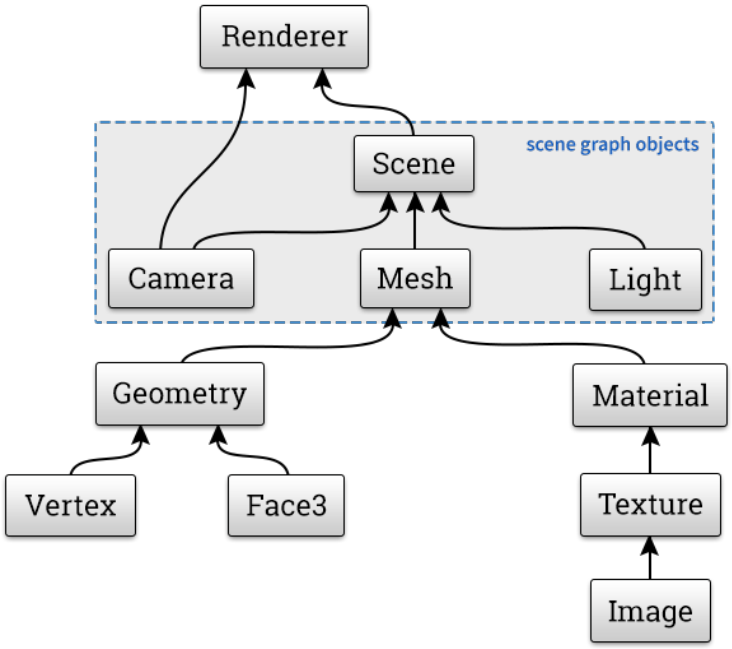
Tämän jälkeen luotiin elementti, jossa grafiikka piirretään. Elementille sai antaa mieleisensä sijainnin ja mitat. ID:n, eli tunnisteen antaminen elementille oli tärkeää, jotta siihen päästiin myös käsiksi JavaScriptista:

HTML: `<div id=”ThreeJS”></div>`
JavaScript: `container = document.getElementById('ThreeJS');`

HTML dokumenttiin tehtiin myös viittaus omasta JavaScript-dokumentista, jossa sovelluksen toiminnallisuuden luominen tapahtuisi.

Three.js:llä kehitetty sovellus rakentui useista osista, joista tärkeimmät olivat rendaaja, scene ja kamera. Osat lisättiin omaan JavaScript tiedostoon tietyin komennoin sekä viittauksin Three.js lähdekoodiin.

Scene on eräänlainen säiliö, joka varastoi kaikki sisältämänsä objektit rendausta varten (KUVIO 4). Kaikki objektit lisätään siis sceneen halutuille paikoilleen. Kamera taas on omalla sijainnillaan oleva objekti scenessä, joka välittää näkemänsä kuvan rendaajalle. Rendaaja laskelmoi ja piirtää kuvan selaimelle kameran välittämän kuvan mukaisesti. Lopulliseen rendaukseen vaikuttavat useammat tekijät, kuten valo-objektit, shaderit ja scenessä sijaitsevat mallit sekä niiden materiaalit ja tekstuurit. (Dirksen 2013, 19-23.)

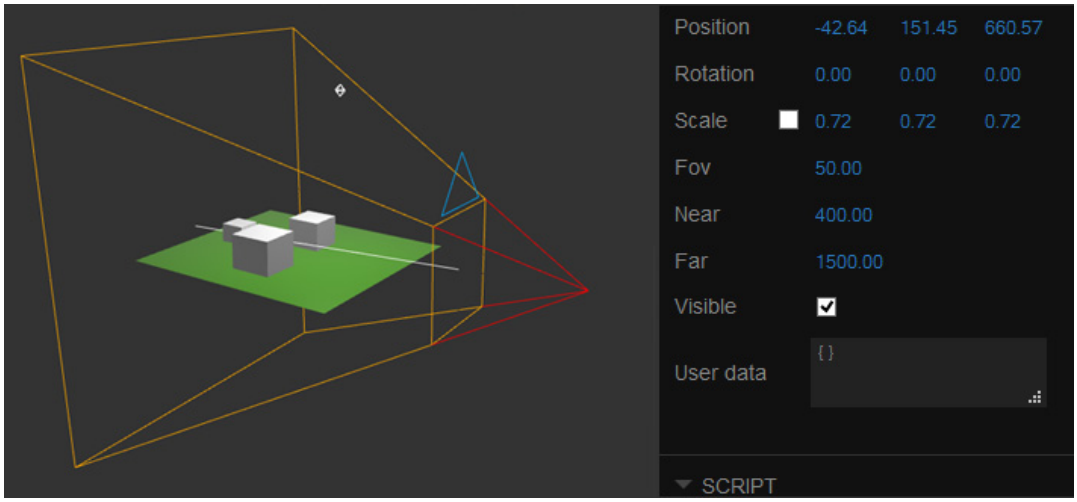



KUVIO 4. Scenen rakenne (Lyons 2015.)

Jokaisella sovelluksen osalla on omat ominaisuutensa, joita voidaan muuttaa tarpeen mukaan. Esimerkiksi kameralle voidaan määritellä Field of View (näkökenttä), lähimmäinen ja kauimmainen piirtämisen piste, kameran sijainti sekä suunta. Piirtopisteiden ulkona olevaa aluetta ei rendata. Kameran tyypiksi voidaan määritellä joko ortograafinen kamera, joka esittää kaukaisen ja lähellä olevan esineen saman kokoisina, tai perspektiivi kamera, joka esittää realistisen kuvan rendaajalle. (Dirksen 2013, 60-63.)

Esimerkki kameran ominaisuuksista (KUVIO 5):

```
var camera = new THREE.PerspectiveCamera(50, width/height, 400,
1500);
```




KUVIO 5. Kamera ja sen ominaisuudet editorissa

Rendaajalle voi myös valita rendaajan tyypin, esimerkiksi WebGL renderer 3D-grafiikalle, tai Canvas renderer 2D-grafiikalle. Rendaajan ominaisuuksiin voi lisätä antialisoinnin, eli reunojen pehmennyksen, ja taustan läpinäkyvyyden (alpha). Jos läpinäkyvyyttä ei haluta käyttää, voidaan taustalle määrätä väri. (Dirksen 2013, 22-23.)

```
var renderer = new THREE.WebGLRenderer( {antialias:true} );
```

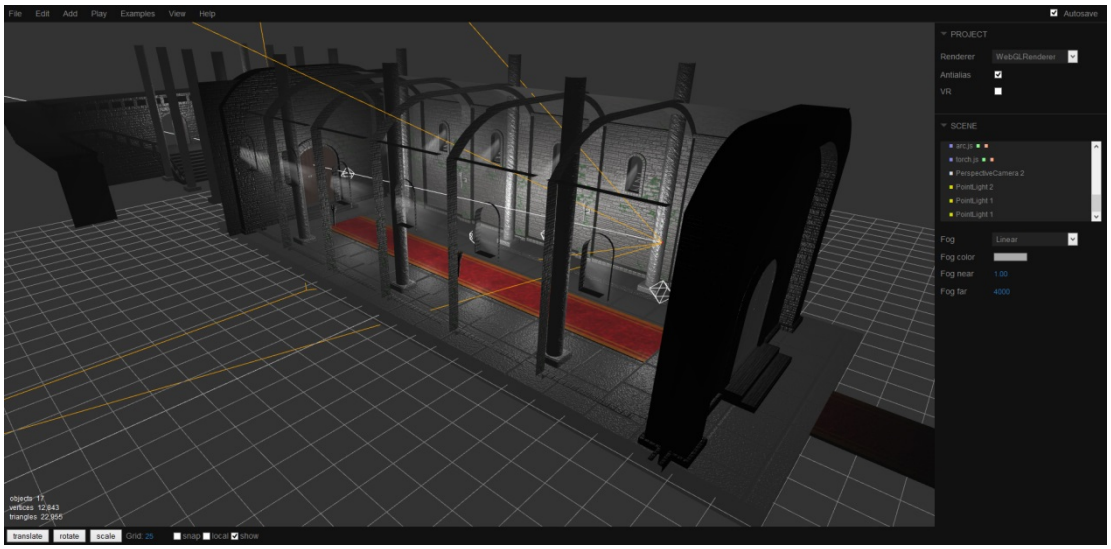
Rendaajalle valittiin HTML-dokumentista elementti, jossa scene piirrettäisiin. Sovellukselle pystyi luomaan oman taustakuvan käyttämällä alpha channelia rendaajalla ja lisäämällä taustakuva HTML-dokumentissa sijaitsevaan erilliseen elementtiin. Muita scenessä sijaitsevia objekteja olivat esimerkiksi valonlähteet ja mallit. Yksinkertaiset mallit, kuten laatikot, voitiin helposti luoda koodin kautta. Monimutkaiset mallit kannatti kuitenkin luoda ulkopuolisella, siihen kehitetyllä 3D-mallinnusohjelmalla. 3D-mallien ominaisuuksiin Three.js:ssä kuuluivat: tekstuurikartat, materiaalit, läpinäkyvyys, väri, shading, kiilto ja geometria. Malli voi myös omistaa animaation. Valojen ominaisuuksiin kuului: tyyppi, kirkkaus, valaisu etäisyys ja varjojen piirtäminen.

Objektin sijainti scenessä voitiin määrittää sen ominaisuuksista: sijoittaminen tapahtui antamalla x,y ja z akselien koordinaatit.

Esimerkkinä komennot, joilla kamera saatiin lisättyä sceneen, ja sijoitettua halutuille koordinaateille:

```
camera.position.set(0,200,0);
scene.add( camera );
```

Objektien kohdistaminen scenessä oli hankalaa koodista käsin ja sitä varten käytettävissä olikin scene-editori (KUVIO 6). Editori oli selaimessa toimiva graafista käyttöliittymää hyödyntävä sovellus, joka mahdollisti mm. objektien asettelemisen scenessä, ja niiden ominaisuuksien muuttamisen. Käyttöliittymä oli verrattavissa esimerkiksi 3D-mallinnusohjelman tai pelin kehitysympäristön käyttöliittymään, jossa scenen rakentamien ”käsin” on helppoa. Editori tarjosi myös muuta oleellista tietoa, kuten verticeiden määrän. Valmis scene voitiin tallentaa omaan projektiin vietäväksi.




KUVIO 6. Three.js scene editor

Kameralle oli hyvä kirjoittaa skripti, jolla sen välittämä kuva saatiin skaalautumaan sijaitsemassaan elementissä. Esimerkiksi, jos elementin mitat oli toteutettu prosenteilla, skaalautui se selaimen ikkunan koon muuttuessa.

JavaScript tiedoston loppuun täytyi vielä lisätä funktio ylläpitämään scenen rendaamista:

```
function render () {
   requestAnimationFrame( render );
   renderer.render(scene, camera);
}
render();
```

Toimivan scenen luominen ei vielä vaatinut paljoa omaa koodia, mutta interaktiivisuuden luominen sisältäisi omat haasteensa. Ensiksi sceneen täytyi kuitenkin saada omia 3D-malleja kokeiltaviksi.

# 4 3D-GRAFIIKAN LUOMINEN SELAINKÄYTTÖÖN

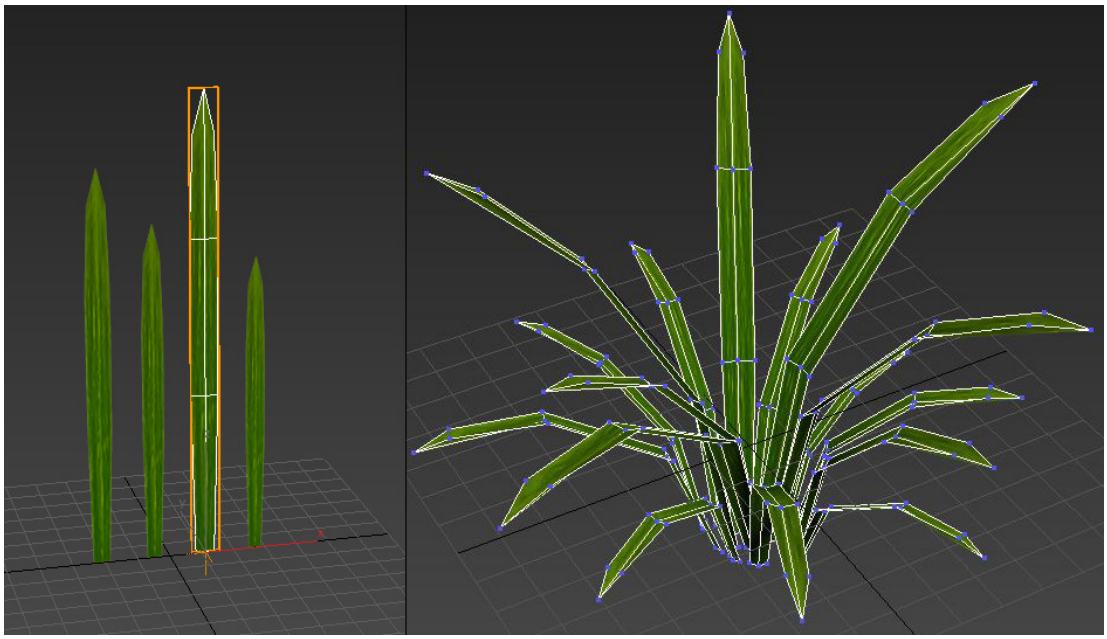
Tässä luvussa käyn läpi selaimeen luotavan 3D-grafiikan mallintamisen ja teksturoinnin osalta tärkeät vaiheet. Käyn myös läpi, kuinka Three.js scenen objekteille voidaan lisätä interaktiivisuutta, ja kuinka scenen visuaalisuuksia saadaan parannettua.

## 4.1 Mallintaminen

Omia 3D-malleja luotaessa verkkokäyttöön käytettiin perinteisiä 3D-mallinnus ohjelmia, kuten Autodesk 3ds Max ja Blender. 3ds Max on normaalisti maksullinen ohjelma, mutta käytössäni oli ilmainen opiskelijaversio. Blender taas on kaikille ilmainen, mutta päädyin kuitenkin käyttämään 3ds Maxia myös sen oletettavasti paremman tuen three.js exportereille. 3D-grafiikan luominen verkkosovelluksille tapahtuu samoin perustein kuin pelikäyttöön.

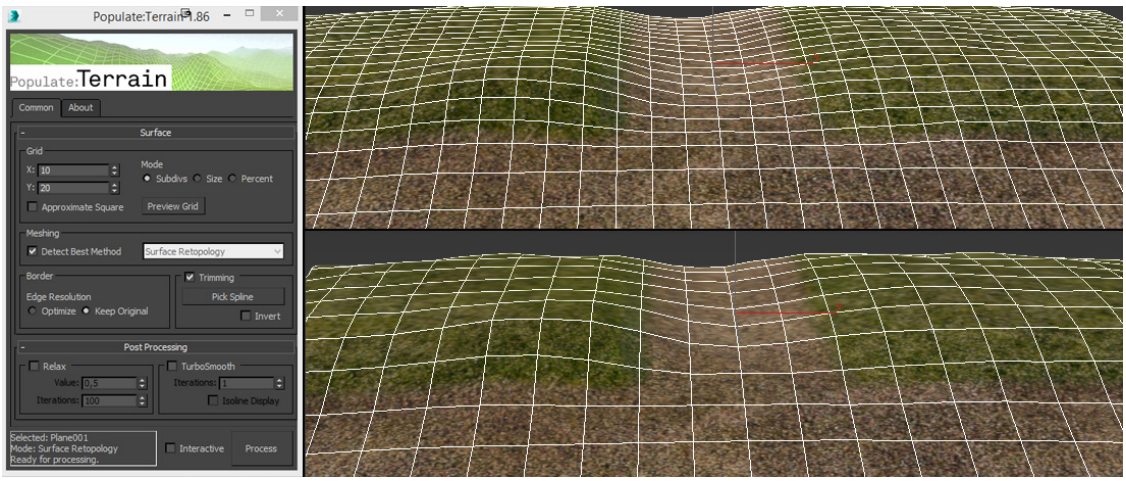
Pelikäyttöön kehitettävien mallien polygonimäärä pyritään pitämään matalana, sillä niiden määrä vaikuttaa rendauksen kestoon. On myös tärkeää tehdä malli muodostumaan joko kolmen tai neljän vertexin polygoneista. Sovellusten käyttäessä reaaliaikaista renderöintiä, tulee hyödyntää kaikki keinot rendausajan lyhyenä pitämiseen. Siihen vaikuttavat kuitenkin monet muutkin tekijät, joten pelkällä polygonien määrällä ei välttämättä aina saada riittäviä tuloksia aikaiseksi. (Silverman 2013.)

Sovelluksia varten tarvittaisiin monenlaisia malleja. Yksi scenessä tarvitsemani malli oli ruoho, joka saatiin toteutettua suhteellisen pienellä polygonimäärällä (KUVIO 7). Ensimmäiseksi tein yhen ruohon korren, jolle lisättiin UV map. UV mapin jälkeen mallille lisättiin bend modifier, jolla se saatiin taivutettua. Vasta tämän jälkeen korsi kopioitiin useammaksi malliksi, jotka sitten muotoiltiin erinnäköisiksi. Mallin pinnat luotiin yksipuolisina, joten Three.js:n puolelta materiaalille täytyi lisätä kaksipuolisuus, jotta tekstuuri saatiin näkymään korressa molemmin puolin.



KUVIO 7. Ruohon mallintamisen vaiheet

Erääseen sovellukseeni täytyi luoda maasto. Maaston olisi voinut luoda myös JavaScriptin kautta, mutta omaan tarkoitukseeni maasto oli helpompi luoda 3ds Maxin kautta: maastossa täytyi olla syvennös polulle. Mallin luominen 3ds Maxin kautta ei ollut ongelma, sillä sen ei tarvinnut olla kovin laaja. Maaston rakentamisen aloitin Plane Object-tyypillä. Aloitin runsaalla määrällä segmenttejä, jotta sain haluamani muodot mallille. Maaston muodon tein paint deformation työkalulla, jolla pinnan muodot sai itse maalata. Lisäsin pintaan myös realistista epätasaisuutta noise modifierilla. Lopuksi optimoin maaston Populate:Terrain-pluginin avulla (KUVIO 8).




KUVIO 8. Mallin optimointi pluginin avulla

Joissakin tapauksissa mallin geometria oli hankala pitää yksinkertaisena. Se ei välttämättä haitannut jos mallin haluttiin vain näyttävän tyylikkäältä, eikä sivulla käytetty muutakaan raskasta sisältöä. Yhtenä esimerkkinä päätin tehdä Centria-logon verkkokäyttöön (KUVIO 9). Esimerkiksi yritys voisi toteuttaa sivuillaan esiintyvän logon 3D-mallilla. Käytin Adobe Illustrator, versio 8 -muodossa olevaa vektorikuvaa logosta, jonka avaaminen onnistui 3ds Max -ohjelmassa. 3ds Maxissa logo avaantui spline -muodossa, eli käyrinä. Muutin mallin editable meshiksi, jotta sain lisättyä sille syvyyttä. Tein logolle myös viistetyt reunat saadakseni niihin kiiltoa scenessä. Viistetyt reunat tehtiin chamfer-modifierilla.


KUVIO 9. Logo illustratorissa, 3ds Maxissa ja selaimessa

## 4.2 Teksturointi

Tekstuureita tehtäessä tuli jälleen muistaa verkkosoveltuvuus: tekstuurien resoluutiot pidettiin pienempinä, ja tarvittaessa hyödynnettiin samaa tekstuuritiedostoa useammalle mallille. Tekstuureiden tekemiseen käytin Adobe Photoshop CS6 –ohjelmaa. Ennen 3D-mallin teksturointia mallille tuli tehdä toimenpide nimeltä UV mapping, eli UV-kartoitus.

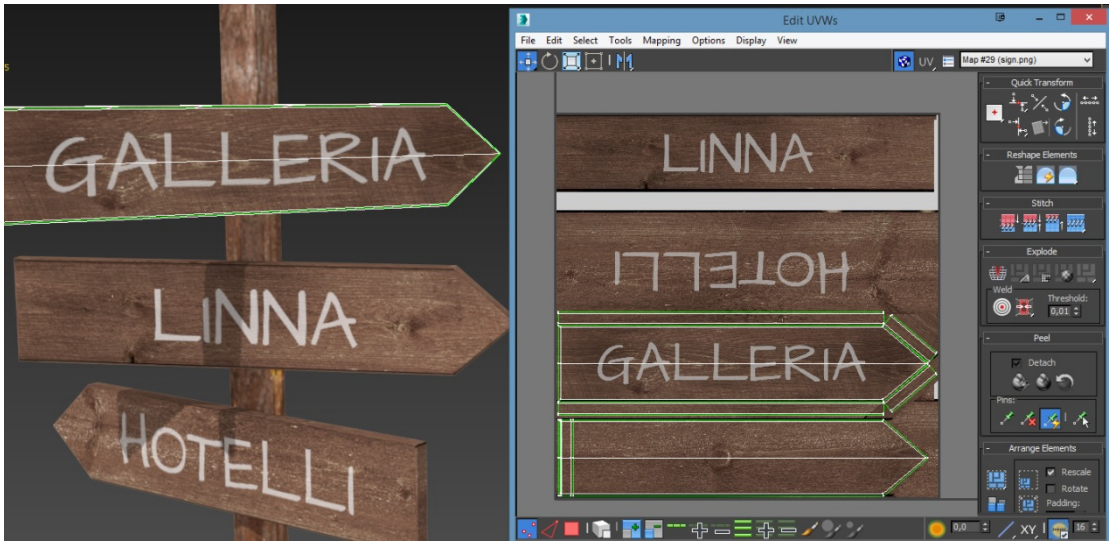
UV mapping on toimenpide, jossa 3D-mallin pinnoista luodaan UV-map, eli kaksiulotteinen koordinaatti kartta. Näille pinnoille voidaan lisätä tekstuurit. U ja V kuvaavat kartan eri koordinaatteja. Vaihe, jossa 3D-mallin pinnoille luodaan ja lisätään tekstuurit, on nimeltään UV-teksturointi. (World Origin 2012.)

UV mapping tehtiin 3ds Max:ssa UVW unwrap -modifierin avulla. Valmis UV map voitiin tallentaa kuvatiedostoksi. Tekstuurikarttaa Photoshopissa luotaessa tekstuurit piirrettiin omalle kuvatasolleen, UV unwrap toiminnolla luodun kartan avustuksella. Tekstuurit tein yleensä joko itse piirtämällä tai valokuvien avulla pyrkiessäni fotorealistisuuteen.

Three.js tukee ainakin color, bump ja specular tekstuurikarttoja (eng. texture map) (KUVIO 10). Color map on yleisimmin käytetty tekstuurikartta, joka kertoo pinnan värin ilman valon tai shaderien vaikutusta. Color map:ia voi käyttää pohjana muiden karttojen tekemiseen. Esimerkiksi bump map, eli korkeuskartta, voidaan tehdä muuttamalla Color map harmaaväriskaalaiseksi. Bump map kertoo mallin pinnan syvyyseroista muuttamatta itse geometriaa. Sen tekstuurikartassa vaalea sävy kertoo korkeimmista pinnoista ja tumma syvemmistä. Specular map voidaan myös tehdä värikartan pohjalta. Specular map määrää mallin pintojen kiiltävyyden. Specular map toteutetaan myös harmaaväriskaalassa, samalla periaatteella kuin bump map. (Digital-Tutors Team 2015.)

KUVIO 10. Color + bump + specular mapit 3D-mallissa

Toteutin malleilleni kaikki 3 erilaista tekstuurikarttaa päästäkseni haluamaani realistisuuteen. Pystyin hyödyntämään bump mappia myös specular mappina useammissa malleissa, sillä niiden eroavaisuudet eivät välttämättä olleet suuret. Joihinkin tekstuuritiedostoihin pystyttiin myös luomaan useamman mallin tekstuurit, vaikka ne liitettiinkin eri objekteina sceneen. (KUVIO 11).




KUVIO 11. Samaa tekstuuritiedostoa voitiin käyttää eri kylteissä.

Valmiit tekstuurit tallensin Photoshopin save for web toiminnolla: png-8 oli yleensä riittävä muoto kuvien tallentamiseen. Näin saatiin tekstuurien tiedostokokoa karsittua ja latausaikoja selaimessa lyhennettyä. Läpinäkyvyyttä sisältäville tekstuureille oli suositeltavaa käyttä png-24 muotoa.

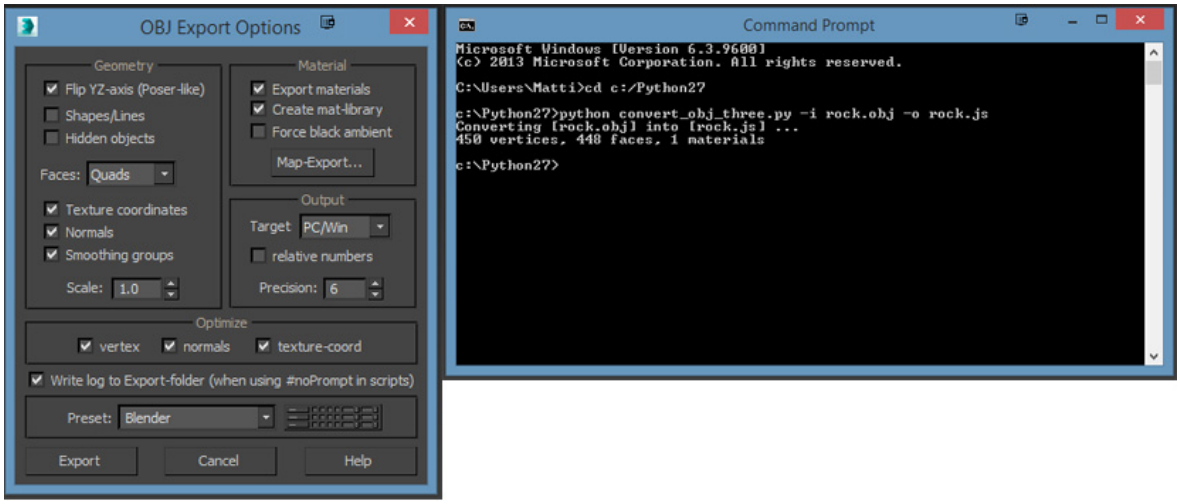
### 4.3 Ulosvienti

Ennen kuin mallit ulosvietiin (export) Three.js:n käyttöä varten, mallit kannatti keskittää mallinnusohjelman näkymässä. Näin malleja voitaisiin kiertää JavaScriptin avulla ongelmitta. Jotta mallit saatiin vietyä selainkäyttöön, vaadittiin mallin muuttaminen .js tai .JSON muotoon. Mallin muuttaminen tapahtui siihen kehitetyllä convert-työkalulla.

Mallin ulosviemiseen 3ds Maxista oli kaksi vaihtoehtoa: ThreeJSExporter ja Python converter. Molemman löytyvät Three.js:n kansiosta: Python converter löytyy ”utils/converters/obj” -kansiosta nimellä convert_obj_three ja threejsExporter

-kansiosta “utils/converters/fbx” nimellä convert_to_threejs. ThreeJS exportetin kanssa oli ongelmia mallin tasoittamisen kanssa (eng. smoothing groups), joten Python converter oli parempi vaihtoehto. Python converterin käyttäminen vaati Python 2.7 -version asentamisen tietokoneelle. Ennen convertointia 3D-malli tuli ulosviedä .obj muotoon 3ds Maxista, ja sitten muuttaa .js muotoon komentorivistä käsin tietyllä komennolla (Despoulain 2012.):

`python convert_obj_three.py -i mesh.obj -o mesh.js` (KUVIO 10):




KUVIO 10. Export .obj muotoon, sekä Python converterilla .js muotoon.

Jos malli sisälsi morph -animaation, täytyi jokainen morphaus ulosviedä erillisinä .obj tiedostoina. Tässä tapauksessa jouduttiin myös käyttämään erilaista python komentoa:

```
python convert_obj_three.py -i unmorphedmesh.obj -m morphmesh*.obj
-o comppiledTargets.js
```

Skeleton animationin ulosvientiin käytettiin Three.js:n kansiosta löytyvää ThreeJSAnimationExporter:ia, joka löytyi kansiosta ”utils/exporters/max”.

JavaScriptissä mallit ladattiin Three.js:n JSONLoaderilla sceneen:

```
jsonLoader = new THREE.JSONLoader();
jsonLoader.load( "model.js", addModelToScene );
```

## 4.4 Interaktiivisuus

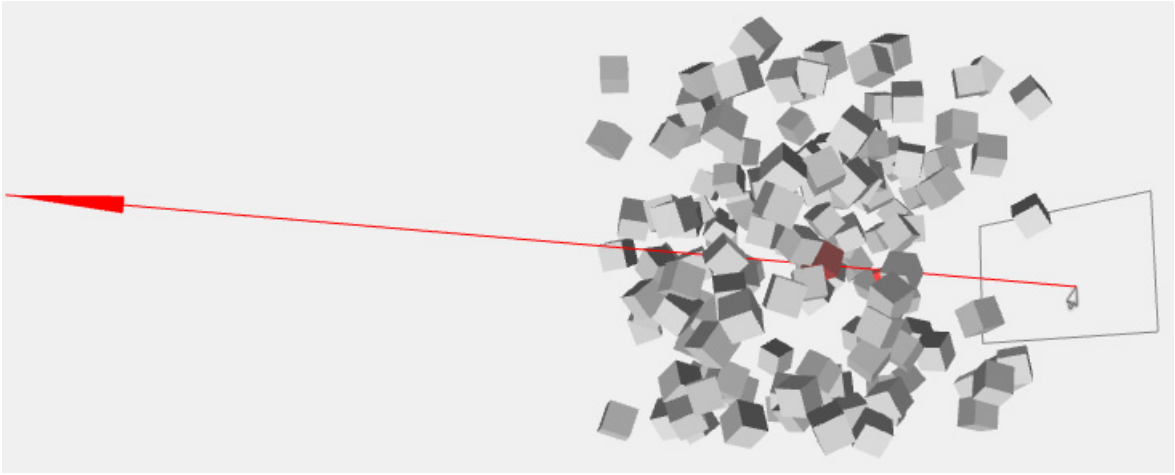
Interaktiivista grafiikkaa kehitettäessä Three.js:n avulla, oli hyvä osata ohjelmoinnin perusteita. Interaktiivisuuden lisääminen oli pääosin ohjelmoijan omissa käsissä. Interkatiivisuutta sovellukseen pystyi luomaan esimerkiksi kontrolleilla, hiiren osoitus- ja klikkaus tapahtumilla, animaatioilla sekä fysiikoilla.

Mahdollisuus kontrolloida kameraa oli tärkeä osa sovellusta. Three.js:n kansiosta löytyi lukuisia valmiita kontrolleja, jotka sovelluksessa liitettiin kameraan. Kokeilin muun muassa orbitControls -kontrolleja, jotka mahdollistivat scenen keskipisteen kiertämisen pitämällä hiiren oikeaa painiketta pohjassa. Toisessa sovelluksessa kokeilin peleistä tuttua first-person -kontrolleja, jotka käyttävät pelattavan hahmon perspektiiviä realistisuuden luomiseksi. First-person -kontrolleissa ohjaaminen tapahtui näppäimistön WASD-näppäimillä. Kontrolleihin pystyi tekemään säätöjä niiden lähdekoodien kautta. Esimerkiksi kuvakulmaa tai zoomausetäisyyttä pystyi rajoittamaan.

Orbit controls sopi hyvin sovellukseen, jossa esitettiin jotakin mallia. Tekemässäni galleria-sovelluksessa mallin näkeminen ympäriltä rajoitetulla zoomilla oli optimaalinen. Visuaalisissa testeissä, joissa tavoitteena oli tunnelman luominen, käytin first-person kontrolleja.

Three.js scenessä sijaitsevien objektien välisen interatiivisuuden pystyi toteuttamaan raycasterin avulla. Raycaster mahdollisti myös käyttäjän ja sovelluksen välisen interaktiivisuuden.

Raycasteria voidaan käyttää apuna scenessä tunnistamaan erilaisia osumia. Raycasterin ansiosta sovellukseen voidaan luoda esimerkiksi tapahtuma, jossa hiirellä klikattaessa tiettyä objektia saadaan jotakin tapahtumaan. Raycaster luo vektorin sceneen hiirtä klikattaessa, sovellus tunnistaa koordinaatit, ja luo sijaintiin säteen (KUVIO 11). Säteen avulla voidaan tunnistaa oliko kohdalla jokin tietty esine ja siten käynnistää esimerkiksi funktio. (Dirksen 2013, 240.)



KUVIO 11. Raycasterin toimintaperiaate (Lyons 2015.)

Eräässä sovelluksessa käytin tekemiäni kylttien 3D-malleja linkkeinä. Käytin raycasteria apuna luomaan niille mouseover ja mouseclick -eventit. Hiiren siirtyessä kyltin päälle, muuttui kursorin tyyli ja kyltti kääntyi. Näin pystyin viestittämään käyttäjälle, että kylttiä oli mahdollista klikata.

Interaktiivisuutta olisi voinut myös tehdä lisäämällä sovellukseen fysiikat. Three.js itsessään ei sisältänyt fysiikkamoottoria, joten apuna olisi täytynyt käyttää erillistä JavaScript-kirjastoa. Ainakin cannon.js -kirjasto sisälsi fysiikat, ja oli myös yhteensopiva Three.js:n kanssa.

### 4.5 Visuaalisuus

Toimivan käyttöliittymän lisäksi miellyttävä ulkoasu ja muut visuaalisuudet antavat hyvän ensivaikutelman sovelluksesta. Tärkeimpiä tekijöitä scenen visuaalisuuden luomisessa oli hyvä valaistus, varjojen piirtäminen sekä shadereiden käyttäminen myös 3D-mallien ja niiden tekstuureiden luoman ympäristön ehostamiseksi. Käytettävät tekniikat riippuivat paljon sovelluksen käyttötarkoituksesta.

Three.js tarjoaa runsaasti eri valotyyppejä, joita voidaan käyttää halutun tunnelman luomiseen. Tiettyyn tulokseen päästääkseen oli hyvä tietää käytössä olevat vaihtoehdot.

AmbientLight on valo, joka luo himmeän valaistuksen kaikkialle scenessä. AmbientLight ei luo varjoja, eikä sillä ole tiettyä sijaintia josta se valaisee. Sitä käytetään muiden valojen rinnalla lisämään tietynlaista väriä sceneen ja esimerkiksi pehmentämään jyrkkiä varjoja. Käytettävissä on myös pointLight, joka valaisee kaikkiin suuntiin sijaintinsa ympärillä. PointLightille voidaan määrätä kirkkaus ja kantama. Sitä voidaan käyttää esimerkiksi kirkkaasta esineestä syntyvänä valonlähteenä. SpotLight taas on spottivalo, joka luo valokeilan haluttuun suuntaan. SpotLightia voidaan myös käyttää Three.js:ssä luomaan varjo. Sen ominaisuuksiin kuului mm. valokeilan leveys, varjon lähin ja kaukaisin piirto piste, varjon resoluutio ja pimeys. Scenessä sijaitseville objekteille täytyy muistaa laittaa varjon vastaanotto ja piirtäminen päälle, jos halutaan varjojen vaikuttavan kyseiseen malliin. SpotLightin asettaminen vaatii enemmän aikaa, mutta sen hyödyllisyys tunnelman luomiseen on merkittävä. Valolle kannattaa tehdä oma kohdepiste, jotta se saadaan osoittamaan haluttuun sijaintiin. Viimeinen käytettävissä olevista tavallisista valoista on DirectionalLight, joka valaisee kaukaisesta pisteestä. DirectionalLight:ia voi verrata aurinkoon. Sen valaisevuus ei vähene pitkälläkään etäisyydellä. (Dirksen 2013, 66-82.)

Shaderit muuttavat scenen mallit pikseleiksi ruudulle rendauksen aikana. Shadereilla voidaan vaikuttaa siihen, kuinka objektit rendataan ruudulle. Shadereiden kirjoittaminen ei tapahdu JavaScriptista, vaan siihen vaaditaan GLSL-kieltä. (Dirksen, J. 2013. 114-116)

Three.js:n mukana tuli kuitenkin tietynlaisia shadereita käytettäväksi: käytettävissä oli *shaderPass*eja, jotka lisättiin kameralle. ShaderPassit lähinnä vaikuttivat siihen, kuinka koko scene piirretään ruudulle. Käytettävissä oli esimerkkinä filmPass, joka loi ruudulle kohinaa, ja BokehPass, joka teki eräänlaisen kohdistusefektin ruudulle. Shadereita käytettäessä tuli ottaa huomioon niiden vaikutus rendausnopeuteen. Sovellukseen voisi myös toteuttaa asetukset, josta ne saataisiin kytkettyä pois päältä tarvittaessa.

Shadereiden ja valojen yhteistyöllä voitiin luoda monenlaista tunnelmaa. Esimerkiksi, käyttämällä vähäistä valaistusta sisätiloissa ikkunoiden luomien valokeilojen muodostamana ja lisäämällä sceneen sumuisuutta saatiin aikaiseksi hyvin erilainen tunnelma, kuin esimerkiksi käyttämällä vain yhtä kirkasta valoa scenessä (KUVIO 12).

KUVIO 12. Shadereilla ja sopivalla valaistuksella voitiin parantaa tunnelmaa

# 5 TULOKSET JA POHDINTA

Lopputuloksena sain aikaiseksi useamman graafisen testin, selkeän kuvan tavoista esittää grafiikkaa selaimessa ja perusteet HTML5:lla sen luomiselle. Myös ohjelmointirajapintojen hyödyllisyys tuli selväksi ja opein myös käyttämään niitä apunani 3D-grafiikan esittämisessä.

Ensimmäinen Three.js:n avulla toteuttamani työ oli kesäprojektin osalta tehty verkkosovellus, jossa toteutin subwoofer-kotelon suunnittelemiseen ja tilamiseen käytettävän sovelluksen prototyyppivaiheeseen. Käyttäjän antaessa mittoja kotelolle, pystyi hän myös tarkastelemaan 3D-mallia kotelosta verkkosivulla reaaliaikaisesti. Kotelon materiaalia valittaessa muuttuivat myös laatikon tekstuurit. Sovelluksessa malli toteutettiin JavaScriptin kautta, jotta mallin muotoa pystyttiin manipuloimaan. Subwoofer sovelluksen toteuttaminen vaati enemmän ohjelmointitaitoja, kun taas muut tekemäni sovellukset keskittyivät graafisiin ominaisuuksiin hyvänä vastapainona. Sovelluksen osalta minun täytyi myös osata ottaa huomioon työn tilaajan vaatimukset (LIITE1).

Aloittaessani kokeilut omien 3D-mallien hyödyntämisestä selainkäytössä, päätin tehdä eräänlaisen gallerian kokeilualustaksi. Sovelluksessa käyttäjä pystyi valitsemaan 3D-mallin listasta ja tarkastelemaan sitä. Sovellukseen kuului myös asetuksia, jotka mahdollistivat mm. mallin materiaalin vaihtamisen (LIITE2).

Yhtenä sovelluksena kehitin tunnelmallisen linnan käytävän, jota käyttäjä voisi tutkia first-person -kontrollien avulla. Sovellukseen sisältyi myös visuaalisuuden kannalta tärkeitä asetuksia. Käyttäjä pystyi valitsemaan kahdesta eri shadereista, Field of View määrän ja scenen sumuisuuden. Ideana oli myös kokeilla, kuinka esimerkiksi seikkailupelin luominen selaimeen onnistuisi. Haasteena oli toteuttaa seiniin törmäyksen havaitseminen, joka jäikin ajan puutteen vuoksi keskeneräiseksi. Tärkeitä seikkoja testissä olivat kontrollit ja visuaalisuuden luominen valaistuksella sekä shadereilla (LIITE3).

Viimeisenä kokeiluna kehitin interaktiivinen scenen, jonka avulla tarkoituksenani oli tutkia 3D-grafiikan hyödyntämistä esimerkiksi interaktiivisen portfolion luomisessa. Interaktiivisuuteen kuului kameran kääntyminen scenen ympärillä hiiren sijainnin mukaan, valon liittäminen hiireen ja objektien kohdistaminen sekä klikkaaminen hiirellä. Scene oli hämärä tienristeys, johon kuului kylttejä. Kyltit toimivat linkkeinä ja hiirellä niitä osoitettaessa kääntyivät ne hieman. Klikatessa niitä aukesi ikkuna erillisille sivuille. Sceneen kuului muita objekteja, kuten animoitua ruohoa sekä kivejä. Shadereina käytin VignetteShaderia tuodakseni reunojen hämäryyttä esille, ja bleachShaderia, joka vahvisti lisää scenen hämärää tunnelmaa (LIITE4).

3D-grafiikan soveltaminen selaimessa on jo pitkällä: lukuisat työkalut ja selainten yhteensopivuus, tuki laitteistokiihdytykselle ja nykyiset internetin siirtonopeudet mahdollistavat sen nykypäivänä. Valmiit JavaScript-kirjastot mahdollistavat tekniikan käyttämisen kaikille ja säästävät myös aikaa sekä ”pyörän uudelleen keksimistä”. Yhteensopivuuksien luomisessa ja välivaiheiden karsimisessa olisi vielä parantamisen varaa helpottamaan prosessia.

Three.js ajoi asiansa käytössäni. Paras tuki oli ehdottomasti Chrome –selaimella. Firefoxin kanssa esiintyi muistiongelmia; ns.”roskienkeruu” ei tuntunut olevan parhaasta päästä ja selaimen kaatuilua tapahtuikin usein. Sovellukset toimivat myös Internet Explorerissa yleensä ongelmitta, mutta esimerkiksi ”hiirilukkoa” (eng. Pointer Lock) käyttävät kontrollit eivät toimineet lainkaan. Firefox ja Chrome -mobiiliselaimet toimivat myös, mutta laitekohtaisia ongelmia saattoi vielä olla tuen kanssa. Tähän opinnäytetyöhön en harmikseni ehtinyt kokeilemaan muita JavaScript –kirjastoja. Three.js tuntui kuitenkin lunastavansa paikkansa dynaamisena kehitysalustana 3D-verkkosovelluksille. Jos kuitenkin haluaa keskittyä esimerkiksi pelien kehittämiseen, on suositeltavaa valita jokin muu alusta.

JavaScript-kirjaston käyttäminen laajennusten sijasta mahdollisti sovelluksen käyttämisen mobiililaitteilla, sen integroimisen web-sivuille ja muuntamisen siellä DOMia hyödyntäen. Käyttäjältä vaadittiin ainoastaan yhteensopiva selain, ja sovelluksesta riipuen, riittävän tehokas laite pyörittämään sitä. HTML5:n laite- ja selainsoveltuvuudet vain jatkavat paranemistaan, joten tekniikan hyödyntäminen

alkaa olla juuri parhaimmillaan. Pluginien tulevaisuus selaimissa taas ei vaikuta lupaavalta lainkaan.

WebGL sovelluksissa huomaamiani yleisimpiä ongelmia olivat hiiren huono reagointi sovelluksessa ja epäuskottava äänimaailma. Jokus törmäsi selaimen muistiongelmiin tai huonoon suorituskykyyn. Ongelmat eivät kuitenkaan olleet ylitsepääsemättömiä.

Interaktiivista 3D-sovellusta voisi käyttää esimerkiksi verkkokaupassa tuotteiden esittelemiseen, virtuaalisten sijaintien luomiseksi, jonkin asian helpommin esittämisessä, seilainpeleissä tai jopa tyylikkään portfolion kehittämisessä. Tekniikan käyttämiseen voi jo törmätä, mutta ei se vielä jokapäiväistä ole. Jos kyseinen tekniikka kiinnostaa, tarjoaa aihe monia mahdollisuuksia, kunhan muistaa ottaa huomioon optimoinnin ja muut tärkeät puolet. Tekniikka kehittyy edelleen, ja lähitulevaisuus voikin jo tarjota edelleen helpompia tapoja lähestyä aihetta.

**LÄHTEET**

Beal, V. 2015. API – application program interface. Www-dokumentti. Saatavissa: http://www.webopedia.com/TERM/A/API.html. Luettu 17.5.2015

Cabello, R. 2015. Three.js. Www-dokumentti. Saatavissa: https://github.com/mrdoob/three.js/. Luettu 17.5.2015

Despoulain, T. 2012. 3D, (WebGL) Max to Three.js workflow tips and tricks. Www-dokumentti. Saatavissa: http://bkcore.com/blog/3d/webgl-three-js-workflow-tips.html. Luettu 17.5.2015

Digital-Tutors Team. 2015. 3D, Understanding the Difference between Texture Maps. Www-dokumentti. Saatavissa: http://blog.digitaltutors.com/understanding-difference-texture-maps/. Luettu 17.5.2015

Dirksen, J. 2013. Learning Three.js: The JavaScript 3D Library for WebGL. Birmingham: Packt Publishing.

Google. 2010. 20 Things I Learned. Www-dokumentti. Saatavissa: http://www.20thingsilearned.com/en-US/threed/2. Luettu 17.5.2015. Luettu 17.5.2015

Hewitson, J. 2013. Three.js and Babylon.js: a Comparison of WebGL Frameworks. Www-dokumentti. Saatavissa: http://www.sitepoint.com/three-js-babylon-js-comparison-webgl-frameworks/ Luettu 17.5.2015

Khronos. 2010. WebGL, Getting Started. Www-dokumentti. Saatavissa: https://www.khronos.org/webgl/wiki/Getting_Started. Luettu 17.5.2015

Labrecque, J. 2015. Publish Flash content on multiple platforms. Www-dokumentti. Saatavissa: https://helpx.adobe.com/flash/how-to/export-flash-multiple-platforms.html. Luettu 17.5.2015.

Lyons, D. 2015. Intro to WebGL with Three.js. Www-dokumentti. Saatavissa: http://davidscottlyons.com/threejs/presentations/frontporch14/#slide-110. Luettu 17.5.2015

Mozilla Foundation. 2015. CSS. Www-dokumentti. Saatavissa: https://developer.mozilla.org/en-US/docs/Web/CSS. Luettu 17.5.2015

Mozilla Foundation. 2015. HTML. Www-dokumentti. Saatavissa: https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/Introduction. Luettu 17.5.2015

Mozilla Foundation. 2015. What is the DOM?. Www-dokumentti. Saatavissa: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction. Luettu 17.5.2015

Parisi, T. 2014. Programming 3D Applications with HTML5 and WebGL, Chapter 1. Introduction. Www-dokumentti. Saatavissa: http://chimera.labs.oreilly.com/books/1234000000802/ch01.html#browser_realities. Luettu 17.5.2015

Parisi, T. 2014. Programming 3D Applications with HTML5 and WebGL, Chapter 1. Introduction. Www-dokumentti. Saatavissa: http://chimera.labs.oreilly.com/books/1234000000802/ch01.html#html5_colon_a_new_visual_medium. Luettu 17.5.2015

Parisi, T. 2014. Programming 3D Applications with HTML5 and WebGL, Chapter 1. Introduction. Www-dokumentti. Saatavissa: http://chimera.labs.oreilly.com/books/1234000000802/ch01.html#d_coordinate_systems. Luettu 17.5.2015

Parisi, T. 2014. Programming 3D Applications with HTML5 and WebGL, Chapter 2. WebGL: Real-time 3D Rendering. Www-dokumentti. Saatavissa: http://chimera.labs.oreilly.com/books/1234000000802/ch02.html. Luettu 17.5.2015. Luettu 17.5.2015

Pilgrim, M. 2015. Let's Call It a Draw(ing Surface). Www-dokumentti. Saatavissa: http://diveintohtml5.info/canvas.html. Luettu 17.5.2015. Luettu 17.5.2015

Rouse, M. 2005. Definition, Flash. Www-dokumentti. Saatavissa: http://whatis.techtarget.com/definition/Flash. Luettu 17.5.2015

Silverman, D. 2013. 3D Primer for Game Developers: An Overview of 3D Modelin in Games. Www-dokumentti. Saatavissa: http://gamedevelopment.tutsplus.com/articles/3d-primer-for-game-developers-an-overview-of-3d-modeling-in-games--gamedev-5704. Luettu 17.5.2015

Slick, J. 2015. What is Renderin?. Www-dokumentti. Saatavissa: http://3d.about.com/od/3d-101-The-Basics/a/Rendering-Finalizing-The-3d-Image.htm. Luettu 17.5.2015

Unity. 2005. THE BEST DEVELOPMENT PLATFORM FOR CREATING GAMES. Www-dokumentti. Saatavissa: https://unity3d.com/unity. Luettu 20.5.2015

World Origin. 2012. Texturing, UV mapping explained. Www-dokumentti. Saatavissa: https://worldorigin.wordpress.com/2012/05/30/uv-mapping-explained-36/. Luettu 17.5.2015

w3schools. 2015. JavaScript HTML DOM. Www-dokumentti. Saatavissa: http://www.w3schools.com/js/js_htmldom.asp. Luettu 17.5.2015

**LIITTEET**

LIITE 1


3D
centria
Auto rotate:
Free view:
MorphAnimations:
Wireframe:
Centria


PALAA ETUSIVULLE
60 FPS (3-60)
CASTLE
START
SHADERIT:
FILM
BOKEH
FIELD OF VIEW:
FOG DISTANCE:


OPINNÄYTETYÖ
GALLERIA
LINNA
HOTELLI