Tuukka Nisso

Dynaaminen 3D-sisältö PDF-esitteessä

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Mediatekniikan koulutusohjelma

Insinöörityö

26.4.2017

Metropolia

Tekijä Otsikko Sivumäärä Aika	Tuukka Nisso Dynaaminen 3D-sisältö PDF-esitteessä 24 sivua + 1 liite 26.4.2017
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Mediatekniikka
Suuntautumisvaihtoehto	Digitaalinen media
Ohjaajat	Yliopettaja Harri Airaksinen

Insinöörityön tarkoituksena oli tutkia, kuinka paineenalaisten putkistojen tulppaamisen ja haaroittamisen työnkulkua pystyisi esittämään selkeämmin ja tarkemmin käyttäen dynaamista 3D-sisältöä yleisesti käytössä olevissa PDF-esitteissä. Tämän lisäksi tuli yrittää arvioida, toisiko sisältö riittävästi lisäarvoa vaadittavaan työpanokseen nähden. Insinöörityössä luotiin paineenalaisen putkiston tulppaamisen eri vaiheita esittävä PDF-esite. Esitteen 3D-sisältöä lukijan olisi mahdollista hallita PDF:ssä sijaitsevien interaktiivisten painikkeiden avulla.

Paineenalaisten putkistojen tulppaamisessa käytettävien tarvikkeiden 3D-mallit malli,nettiin käyttäen hyväksi tuotteiden tarkkoja CAD-malleja, joista luotiin low poly -mallit 3ds Max -ohjelmalla. Tämän jälkeen low poly -mallit vietiin OBJ-muodossa Adobe Photoshopiin, jossa malleille luotiin tarvittavalla tarkkuudella materiaalit. Photoshopista mallit vietiin U3D-muodossa Adobe Acrobatiin, jossa 3D-malleille luotiin animaatioita JavaScriptin avulla.

Insinöörityössä ei saavutettu aivan kaikkia tavoitteita, sillä pyörimisliikkeen määrittäminen käsin Acrobatissa osoittautui odotettua haastavammaksi, eikä kaikkien mallien pyörimisliikettä saatu mallinnettua vaadittavalla tarkkuudella. Tästä huolimatta dynaaminen 3D-sisältö toi paljon lisäarvoa esitteisiin.

Työn teko osoittautui odotettua haastavammaksi ja työläämmäksi. Työ oli kuitenkin erittäin mielenkiintoinen prosessi, jonka tulokset ja ongelmanratkaisut toivottavasti hyödyttävät myös muita vastaavanlaisissa projekteissa.

Avainsanat	Adobe 3D-PDF, JavaScript, low poly -3D-mallinnus

Author Title Number of Pages Date	Tuukka Nisso Dynamic 3D-content in PDF-brochure 24 pages + 1 attachment 26 April 2017
Degree	Bachelor of Engineering
Degree Programme	Media Technology
Specialisation option	Digital Media
Instructors	Harri Airaksinen, Principal lecturer

The purpose of this thesis was to inspect how to present the work flow of hot tapping plugging under pressure more clearly and accurately in PDF-brochures by adding animated and dynamic 3D-content into the PDF. In addition the second primary purpose was to evaluate the extra value that it brought to the brochures compared to the required work. In this thesis a brochure presenting a hot tapping plugging under pressure using the goods used by Tonisco System Oy was created. The 3D-content in the PDF-is controlled by buttons in the PDF.

The 3D-low poly -models used in the thesis was created with 3ds Max by imitating the precise 3D-models presented by Tonisco System. 3D-models created in 3ds Max was exported as OBJ to the Adobe photoshop, where the materials were made with required accurate. From Photoshop the 3D-models were exported as U3D to the Adobe Acrobat where the animations were made by scripting wit JavaScript.

The thesis didn't accomplish all the objectives that were set for it. Because the rotating movement of the objects was too complicated to be made accurate enough in Acrobat. Even though the rotating movement is missing from the brochures, the extra value for the brochures become to be.

The thesis proved to be more challenging and laborious than planned. Even though it was really interesting process. I hope that the problems and solution in this thesis would become helpful for others working in same kind of processes.

Keywords	Adobe 3D-PDF, low poly -modeling, JavaScript

Sisällys

1 Johdanto 1

2 Esitysmateriaalin luomismenetelmiä 2

2.1 PDF-tiedostomuoto 3
2.2 Polygoni-3D-mallinnus 4
2.3 JavaScript-ohjelmointikieli 6

3 Dynaamisen 3D-PDF-esitteen suunnittelu 7

3.1 Työskentely Adobe Acrobatissa 8
3.2 3D-mallien käyttö PDF-tiedostossa 8

4 Esitteen luominen 11

4.1 Low poly -mallinnus dynaamista 3D-sisältöä varten 11
4.2 Animointi 14

5 Ongelmia ja ratkaisuja 16

6 Yhteenveto 20

Lähteet 22

Liitteet

Liite 1. Esimerkki Acrobatin JavaScript-funktiosta

Lyhenteet

PDF	Portable Document Format. Ohjelmistoriippumaton, siirrettävä tiedostomuoto, valmiiden julkaisujen siirtoon.
ASCII	American Standard Code for Information Interchange. 7-bittinen merkistö, joka sisältää perusaakkoset, numerot, väli- ja erikoismerkkejä.
API	Application Programming Interface, Ohjelmointirajapinta. Määritelmä, jonka mukaan eri ohjelmat pystyvät keskustelemaan keskenään.
U3D	Universal 3D. Tiedostomuoto 3D-grafiikan pakkaamiseen.
PRC	Product Representation Company. Tiedostomuoto 3D-grafiikan pakkaamiseen.
OBJ	Wavefront Object file. Yleisesti tuettu tiedostomuoto 3D-grafiikan pakkaamiseen
STP	STEP 3D-CAD tiedosto. Laajasti tuettu tiedostomuoto 3D-grafiikan pakkaamiseen.
ECMAScript	Euroopan tietokonevalmistajien yhdistyksen standardoima, tavaramerkitty skriptikielispesifikaatio.
CAD	Computer Aided Design. Tietokoneavusteinen suunnittelu. Mallintaminen, jota käytetään erityisesti tuotesuunnittelussa.
NURBS	Non-Uniform Rational B-Splines. Matemaattinen malli käyrien ja pintojen luomiseen tietokonegrafiikalla.
ASE	Autodesk ASCII Scene Export-tiedosto. Autodeskin kattavasti luettu 3D-informaatiota sisältävä tiedostomuoto.

1 Johdanto

Tonisco System Oy käyttää paineenalaisten putkistojen haaroitus, ja tulppausesitteissään yleisesti PDF-muotoisia ohjeistuksia, jotka sisältävät paljon kaksiulotteisia kuvia työn eri vaiheista. Tämän insinöörityön tarkoituksena on tutkia, onko kyseisiä PDF-tiedostoja mahdollista korvata dynaamista 3D-sisältöä sisältävillä PDF-tiedostoilla.

Animoitujen 3D-PDF-tiedostojen hyöty verrattuna staattisiin PDF-ohjeistuksiin olisi ohjeistuksien ja esitteiden selkeyttäminen ja mahdollinen lukijan kiinnostuksen lisääminen materiaalia kohtaan. Kun lukija on interaktiivisessa kanssakäymisessä luettavan materiaalin kanssa, hän todennäköisesti myös keskittyy siihen enemmän.

Esitteen ensisijainen tarkoitus on antaa lukijalle mahdollisimman paljon tietoa asian jatkokäsittelyä varten. Jotta lukija saa esitteestä mahdollisimman paljon tietoa, tulee esitteen olla selkeä ja yksinkertainen. Esitteessä tulee kuitenkin olla riittävästi informaatiota, jotta lukija kykenee tekemään sen pohjalta jatkopäätöksiä. Tämä käsite, riittävä, on pidettävä mielessä koko työn ajan, sillä informaatiota on helppo antaa liikaa tai liian vähän. Riittävä on siitäkin jo vaikea käsite, että eri toimijoille käsite tarkoittaa eri asioita. Insinöörityössä luotavan dynaamista 3D-sisältöä sisältävän PDF-esitteen esisijaisena lukijakunta ovat alan ammattilaiset, ja esitteen on tarkoitus esittää paineenalaisten putkistojen tulppaamisessa tapahtuvia työn vaiheita.

Insinöörityöraportissa kerrotaan, kuinka dynaamista 3D-grafiikkaa sisältävä PDF-tiedosto luodaan käyttäen Adoben Acrobat-ohjelmistoa. Työstä lukija saa kattavan käsityksen työn eri vaiheista ja työhön vaadittavista taidoista.

2 Esitysmateriaalin luomismenetelmiä

Esitysmateriaalina käytettäviä esitteitä on tällä hetkellä kahdenlaisia: Microsoft Wordin kaltaisilla ohjelmilla luotuja PDF-esitteitä, jotka sisältävät paljon tekstiä ja tekstin sisältöä vahvistavaa 2D-grafiikkaa, kuten kuvia tai kaavioita. Vaihtoehtoisesti on esittelymateriaalina käytettäviä usein Microsoft Powerpointin kaltaisilla ohjelmilla luotuja esityksiä, jotka koostuvat useasta sivusta eli diasta. Diat sisältävät usein esittäjän esitystä vahvistavaa sisältöä, kuten tekstiä, kuvia, kaavioita tai videoita. Wordilla luodut esitteet ovat kuitenkin staattisia ja hieman epäselviä, ja niiden sivumäärän saattaessa kasvaa useampaan kymmeneen. Powerpointilla luotu esitys taas on usein dynaamisempi ja selkeämpi, mutta se vaatii usein erillisen henkilön pitämään esityksen. Kumpikaan ohjelmista ei myöskään tue 3D-sisältöä.

Microsoft Word

Word on Microsoftin vuonna 1983 julkaisema tekstinkäsittelyohjelma, joka ei alkuaikoinaan onnistunut saavuttamaan suurta suosiota, sillä sen käyttöliittymä oli niin erilainen verrattuna sen ajan suosituimpaan tekstinkäsittelyohjelmaan, WordStariin. Microsoftin jatkuvan kehityksen myötä Wordista tuli kuitenkin loppujen lopuksi markkinoiden suosituin tekstinkäsittelyohjelma ja se on nykypäivänä markkinoiden ylivoimaisesti suosituin kaupallinen tekstinkäsittelyohjelma. [1.]

Tekstinkäsittelyyn suunniteltu Word onkin tehokas työkalu tekstin ja Wordin omien objektien, kuten kaavioiden luomiseen ja muokkaamiseen. Word tukee myös yksinkertaista multimediaa ja animaatiota. Yhdeksi Wordin ja muiden tekstinkäsittelyohjelmien ongelmaksi on muodostunut muotoilujen ja multimedian hajoaminen, kun tiedostoja avataan toisella käyttöjärjestelmällä tai toisessa ohjelmassa.

Microsoft Powerpoint

Powerpoint on Microsoftin, alun perin Macintoshille, suunnittelema esitysgrafiikka ohjelma. Nykyään ohjelmasta on versiot Windows-ja Ios-käyttöjärjestelmille. Powerpoint on tehokas ja helppokäyttöinen ohjelma diamuotoisen esitysgrafiikan tekemiseen ja esittämiseen. Powerpoint esitykset rakentuvat useista yksittäisistä sivuista, dioista. Dioissa voi olla tekstiä, grafiikkaa, ääntä, elokuvia ja myös muita objekteja, joita voidaan järjestellä diaan halutulla tavalla.

Powerpoint tarjoaa sisällölle kolmea erilaista liikettä:

1 elementtien sisääntulo, painotus ja poistoliike

2 diojen vaihtuvuusliike, joita voidaan animoida usealla eri tavalla

3 mukautetut animaatiot, joilla käyttäjä voi luoda liikettä elementeille.

Powerpoint tarjoaa useita toimintoja, joiden avulla käyttäjä pystyy luomaan ja muokkaamaan ammattimaisia esityksiä. Powerpointin tiedostoissa on Wordin tavoin riskinä niiden sisällön hajoaminen tai toimimattomuus, kun tiedostoja avataan toisella käyttöjärjestelmällä tai ohjelmalla. [2.]

2.1 PDF-tiedostomuoto

PDF (Portable Document Format) on tiedostomuoto, jota käytetään yleisesti dokumenttien esittämiseen ohjelmistosta, laitteistosta tai käyttöjärjestelmästä riippumatta. Jokaisessa PDF-tiedostossa on kaikki tarvittava data dokumentin esittämiseen. [3.] Aluksi PDF-oli tarkoitettu vain tekstimuotoisen materiaalin esittämiseen, mutta PDF-on kehittynyt jatkuvasti eteenpäin ja kykenee nykypäivänä esittämään interaktiivista mediaa ja tukee myös Javascript ohjelmointikieltä. Mahdollisuus upottaa 3D-malleja PDF-tiedostoon tuli mukaan tammikuussa 2005, kun Adobe julkaisi Acrobat 7 ja PDF-1.6 version. [4.] PDF-onkin tarkoitettu materiaalin esittämiseen eikä sen luomiseen. PDF:llä esitettävät mallit on luotu muilla ohjelmistoilla ja tämän jälkeen käännetty muotoon, jota PDF-kykenee käsittelemään.

PDF:n kehitys alkoi vuonna 1992, kun yksi Adoben perustajista, John Warnock, käynnisti projektin nimeltään Camelot, jonka tavoitteena oli saada muunnettua paperiset dokumentit digitaaliseen muotoon. Tällöin näiden dokumenttien lähettäminen ja tarkastelu olisi mahdollista sähköisesti, ja niitä voisi myös tulostaa. Vuoteen 1992 mennessä projekti oli kehittynyt PDF:ksi. Nykyisin formaatti on tiedostomuoto, johon niin yritykset kuin yksityishenkilötkin ympäri maailmaa luottavat. [5.] PDF-oli Adoben kontrolloima formaatti vuoteen 2008 asti, jolloin siitä tuli ISO 32000-1:2008 mukainen avoin standardi [4].

PDF-tiedosto on 7-bittinen ASCII-tiedosto, poikkeuksina tietyt elementit, jotka saattavat sisältää binääri-sisältöä. PDF-tiedoston sisältämät oliot voivat olla suoria, jotka ovat upotettuina muihin olioihin, tai epäsuoria. Epäsuorat oliot on numeroitu olionumerolla ja kehitysnumerolla, ja ne on määritelty ”obj” ja ”endobj”-avainsanoilla. Indeksitaulu seuraa päärakennetta ja määrittää bitti vastineen jokaiselle epäsuoralle oliolle tiedoston alusta lähtien. Tämä rakenne mahdollistaa tehokkaan pääsyn olioihin ja sallii myös muutosten tekemisen näihin olioihin lataamatta koko tiedostoa uudelleen. [1.]

PDF on erilaisen grafiikan ja tiedon jakamiseen tarkoitettu formaatti. Se mahdollistaa muilla ohjelmilla luodun grafiikan nitomisen yhteen yhdeksi tiedostoksi, jonka voi jakaa turvallisesti eteenpäin. PDF-kääntää muiden ohjelmistojen luoman grafiikan itselleen sopivaksi, niin että se näyttää samalta käyttöjärjestelmästä riippumatta. Kun tiedostorakenne on esitetty, se mahdollistaa myös työssä vaadittavan dynaamisen sisällön ohjelmoimisen JavaScriptin avulla. [Liite 1.]

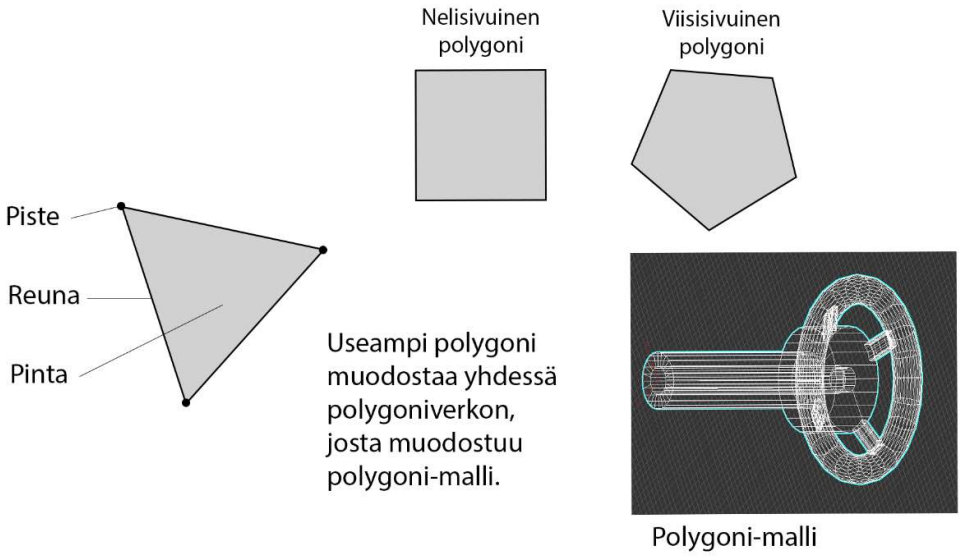
2.2 Polygoni-3D-mallinnus

Polygonit rakentuvat geometriasta, joka perustuu pisteisiin (engl. vertex), reunoihin (engl. edge) ja pintoihin (engl. face). Näiden avulla rakennetaan 3D-malli polygonimallinnuksessa [kuva1]. Tämä on helppokäyttöinen ja yleisesti käytetty tapa luoda 3D-sisältöä elokuvissa, interaktiivisissa videopeleissä ja internetissä.

Polygonit ovat siis suorasivuisia muotoja, joita rajaavat 3D-avaruudessa sijaitsevat pisteitä yhdistävät suorat. Näiden suorien sisäpuolelen rajaamaa aluetta kutsutaan yleisesti pinnaksi. Nämä kolme, piste, suora, pinta, ovat polygonien peruselementtejä. Näitä kolmea peruselementtiä luomalla ja muokkaamalla rakennetaan polygoneja. Yleisimmät polygonit ovat joko kolmi- tai nelisivuisia.

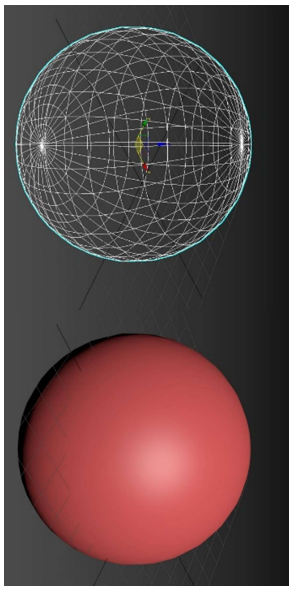
Yksittäinen polygoni on siis yleisesti sama kuin pinta, joka rakentuu alueesta, jota rajaavat kolmea tai useampaa pistettä yhdistävät suorat. Usean pinnan ollessa yhteydessä toisiinsa ne rakentavat pintaverkon, jota kutsutaan polygoniverkoksi. Näistä polygoniverkoista koostuu polygonimalli.[13.]

-

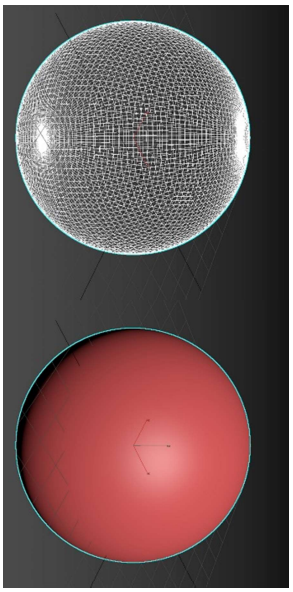


Kuva1. Polygonimallin rakentuminen-

Polygonimallinnuksessa on useita termejä, joista tässä työssä isoksi osaksi nousi low/high poly -mallintaminen [kuvat 2 ja 3]. Koska mallit koostuvat polygoneista myös mallien laskennallinen koko tietokoneiden niitä käyttäessä riippuu pitkälti polygonien määrästä.



Kuva 2. Low poly -malli.



Kuva 3. High poly -malli.

Low poly -mallintamisella tarkoitetaan mallia, joka koostuu polygoniverkosta tai -verkoista, jotka sisältävät suhteellisen pienen määrän polygoneja. Low poly -malleja käytetäänkin paljon erilaisissa tarkoituksissa, joiden yhdistävänä tekijänä on mahdollisimman nopea ja laskentatehoa mahdollisimman vähän käyttävä tarve. Low poly -mallit sisältävät usein high poly -malleja vähemmän yksityiskohtia, minkä vuoksi low poly -malleja käytetään usein kohteissa, joissa täysi realistisuus ei ole tarpeellista.

2.3 JavaScript-ohjelmointikieli

JavaScript on korkean tason dynaamisesti tyypitetty, oliopohjainen ja tulkattava komentosarja- eli skriptikieli. Koska JavaScript tiedostoa ei ennen suorittamista tarvitse kääntää omaksi ohjelmakseen vaan se voidaan tulkita suoraan, sitä kutsutaan skriptiksi. JavaScript vaatii isäntäympäristön toimiakseen, jossa JavaScript-tulkki suorittaa suoritettavan koodin käsky kerrallaan. [7.] JavaScript on standardisoitu ECMAScript- kielen spesifikaation mukaisesti [8].

JavaScript kehitettiin alun perin Netscape Communicationsissa vuonna 1995, tarkoituksena saada tehtyä web-selaimen sisällöstä dynaamisempaa. Huolimatta siitä, että alun perin moni ammattilaisohjelmoija panetteli JavaScriptiä, koska sen kohdeyleisönä pidettiin enemmän web julkaisijoita ja muita ”amatöörejä”. JavaScript on kasvanut yhdeksi käytetyimmistä ohjelmointikielistä juuri sen laajan kohdeyleisön vuoksi. Nykypäivänä jokainen moderni verkkoselain ja eri ohjelmat kasvavissa määrin tukevat JavaScriptiä.

JavaScript on dynaamisesti tyypitetty, mikä tarkoittaa että jokainen tyyppi on yhdistetty arvoon lausekkeen sijaan. Tämän myötä muuttuja, jonka on aikaisemmin määritelty olevan esimerkiksi numero, voidaan myöhemmin määritellä olevan merkkijono. Olio on JavaScriptissä prototyypeillä täydennetty taulukkoihin assosioitu olio. Jokainen merkkijonoavain antaa nimen olion ominaisuuksille, joihin pääsee käsiksi kahdella tavalla: pistemerkinnällä (obj.x = 0) tai sulkumerkinnällä (obj[”x”] = 0). Ominaisuuksia on mahdollista lisätä, muokata tai poistaa dynaamisesti ohjelman ajon eli suorittamisen aikana.

Olioiden ohella toinen erittäin suuri osa JavaScript-kieltä ovat funktiot [koodiesimerkki 1]. Funktiot ovat olioiden kanssa samantapaisia, mutta ne sisältävät toiminnallisia erityisominaisuuksia. Funktio on koodilohko, joka kirjoitetaan valmiiksi, jotta kun koodia tarvitaan, voidaan koodilohkoa kutsua suoritettavaksi funktion nimellä.

```
function nakyvyysSovitin(){
    sovittimet = c3d.scene.nodes.getByName("sovittimet" );
    naky = sovittimet.firstChild;
    naky.visible = true;
    while (true) {
        try{
            naky = naky.nextSibling;
            naky.visible = true;
        }
        catch (err) {
        break
        }
    }
//  app.clearInterval(anim2);
    anim2 = app.setInterval("sovitinLasku()",40);
}
```

Koodiesimerkki 1. JavaScript-funktio, jota kutsuttaessa funktio toteuttaa ennalta kirjoitetun koodilohkon.

3 Dynaamisen 3D-PDF-esitteen suunnittelu

Insinöörityön tarkoituksena oli saada tilaajayrityksen esitteille uusi ja selkeämpi vaihtoehto. Tämä tuli pitää työtä suunnitellessa mielessä vahvasti. 3D-maailmassa työskennellessä asioista tulee helposti tehtyä todella yksityiskohtaisia ja tiedostokooltaan suuria. Tässä työssä oli kuitenkin tärkeää, että tiedostokoot pysyisivät pieninä, jotta tiedostojen jakaminen ja käyttäminen onnistuisi olosuhteista riippumatta. Nykyisten käytössä olevien PDF-tiedostojen keskimääräinen koko vaihtelee kahden megatavun kummallakin puolen. Tavoitteena oli saada uudet, animoidulla 3D-sisällöllä varustetut PDF-tiedostot pidettynä alle viidentoista megatavun kokoisina. Lopullisen tuotteen käytön sujuvuuden lisäämiseksi tämä helpottaisi myös tuotteen tekemistä. Kevyiden, low poly -mallien animointi vaatii luonnollisesti tietokoneelta vähemmän kuin suurten ja raskaiden high poly -mallien.

3D-mallien lisäksi toinen isosti huomioon otettava seikka oli animointi ja se kuinka sen saa toteutettua. Yleensä animaatiot toteutetaan siihen tarkoitetussa ohjelmassa kuten Autodeskin 3ds Max-ohjelmassa. Tämän työn edetessä sain selville, ettei animointia voinut kuitenkaan toteuttaa tällä tavalla, vaan ne tulisi kirjoittaa käsin käyttäen JavaScriptiä. Tätä varten piti myös animaatiot suunnitella hieman yksinkertaisemmiksi, kuin ne olisivat voineet olla, mikäli ne olisi voinut toteuttaa erillisellä ohjelmalla.

3.1 Työskentely Adobe Acrobatissa

Adobe Acrobat on Adobe Systemsin kehittämä ohjelma, joka on tarkoitettu PDF-tiedostojen lukemiseen, luomiseen ja muokkaukseen. Ohjelmasta on olemassa kaksi eri versiota, Reader, joka on tarkoitettu ainoastaan lukemiseen, ja maksullinen Acrobat ohjelmisto, jolla PDF-tiedostoja voi luoda ja muokata. Acrobat kykenee myös lukemaan PDF-tiedostoihin tallennettua tekstiä ääneen.

On olemassa muutamia muitakin ohjelmia, joilla voi luoda ja lukea 3D-PDF-tiedostoja. Näiden ohjelmien ongelmana on, että ne eivät tue toisiaan. Tämän vuoksi käytettää alustaa suunniteltaessa päädyin Adobe Acrobat ohjelmistoon, sillä Adobe Acrobat Reader on yksi yleisimmistä ja käytetyimmistä PDF-lukuohjelmista maailmassa [5].

Jokaisella markkinoilla olevalla tuotteella on luonnollisesti tietyt tiedostomuodot, joita ohjelmat tukevat. Adobe Acrobatissa tuetut 3D-geometriaa sisältävät tiedostomuodot olivat U3D ja PRC. Nämä tiedostomuodot olivat minulle täysin uusia, eikä käyttämistäni 3D-mallinnusohjelmista pystynyt tuomaan geometriaa ulos näissä muodoissa. Tätä varten tuli tehdä tutkimusta, ja loppujen lopuksi Adoben keskustelufoorumeilta löytyi vihje, kuinka Adobe Photoshop-ohjelma pystyisi lukemaan OBJ-tiedostomuotoa, ja tuomaan ulos 3D-geometriaa myös U3D-muotoisena.

Acrobat tukee JavaScriptin käyttöä PDF-dokumenteissaan [9], joten työssä käytetyt animaatiot 3D-sisältöön tuli suunnitella tehtäväksi käyttäen JavaScriptiä. Adobelta löytyy kattavat oppaat [10;11] kuinka päästä käsiksi PDF-tiedoston olioihin ja sen 3D-sisällön olioihin.

3.2 3D-mallien käyttö PDF-tiedostossa

3D-malleja suunniteltaessa tärkein asia mietittäväksi oli jo aikaisemmin mainitsema: kuinka pitää tiedostokoko maltillisena. Tiedossa oli jo aikaisemmista projekteista yrityksen kanssa, että sillä on todella tarkat CAD-ohjelmalla tehdyt mallit olemassa. Työtä suunniteltaessa näiden mallien käyttö oli prioriteettina ajan käytön ja tarkkuuden optimoimiseksi. Mallit oli tarkoitus avata 3ds Max-ohjelmalla ja tehdä niihin tarvittavat pienet muutokset, minkä jälkeen ne olisi viety OBJ-muodossa Photoshopiin.

3ds Max

3ds Max on Autodeskin kehittämä, ammattilaiskäyttöön suunnattu 3D-mallintamiseen ja animointiin suunnattu ohjelma [12]. 3ds Max tukee kolmea tapaa luoda 3D-malleja; Polygoni mallintaminen, NURBS ja pinnan muokkaus-työkalulla.

NURBS-malleja käytetään yleensä suunnitteluprosesseissa, joissa tarvitaan tarkkoja ja täsmällisiä kuvauksia. NURBS-käyrillä on mahdollista esittää kaikkia luonnollisia ja keinotekoisia pintoja. NURBS-mallien pitäisi olla hyviä pyöreiden tai kurvikkaiden low poly -mallien luomiseen. [14.]

Pinnan muokkaus-työkalulla (engl. Surface tool) mallit luodaan tekemällä 3ds Maxilla muotoja (engl. spline), joihin käytetään 3ds Maxin sisältämää muokkainta ”surface”. Tämä muokkain yhdistää jokaisen kolmen tai neljän pisteen ryppään pinnaksi. [15.]

Adobe Photoshop

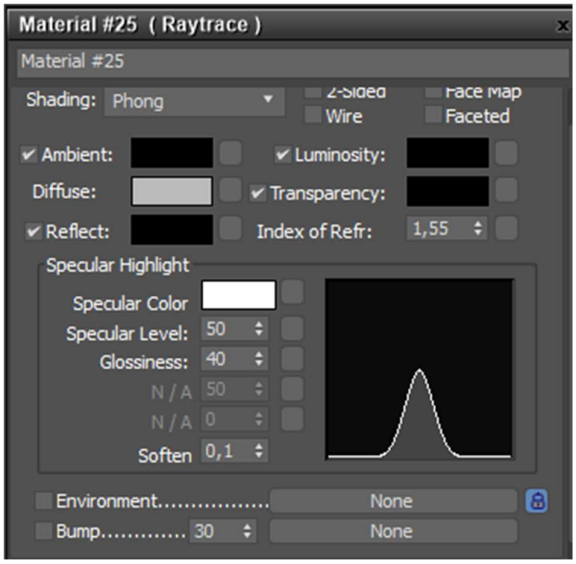
Adobe Photoshop (jatkossa Photoshop) on Adobe Systemsin luoma grafiikkaeditori. Sen suosiosta nykypäivänä kertoo paljon jo se, että usein kuulee kuvankäsittelystä kuulevan termiä ”photoshoppaus”. Usein Photoshopia pidetään ainoastaan kuvienkäsittelyohjelmana, mutta se on aljon enemmän. Ohjelma kykenee myös rajoitetusti muokkaamaan ja luomaan tekstiä, 3D-grafiikkaa ja videota.

Materiaalit

Geometrian lisäksi 3D-mallit tarvitsevat niille ominaisen materiaalin. Ilman kunnollisia materiaaleja ja tekstuureja malli ei näytä visuaalisesti hyvältä, mutta hyvin tehtynä materiaalit ja tekstuurit tuovat paljon lisäarvoa malleille. Kunnollisten materiaalien ja niiden tekstuurien tekoon saa helposti kulumaan paljon aikaa. Onneksi nykypäivänä internetistä löytää valtavan määrän materiaaleja, jotka sisältävät myös tekstuurin. Osa internetin materiaaleista on luonnollisesti maksullisia, mutta laadukkaita materiaaleja löytää myös ilmaiseksi.

3D-mallin materiaalia voisi helposti pitää vain sen sisältämien värien ja kuvioiden summana, mutta materiaali sisältää myös paljon muita ominaisuuksia, jotka tulee ottaa huomioon materiaaleja luotaessa. Yleisimmät materiaaliin vaikuttavat ominaisuudet ovat seuraavat [kuva 5]:

- Diffuse. Ominaisuus, joka määrittää materiaaliin pääasiallisen värin ja/tai kuvion.
- Reflect. Nimensä mukaisesti ominaisuus tarkoittaa pinnan heijastavuutta ympäristöön nähden.
- Luminosity. Tämän ominaisuuden avulla voidaan määrittää materiaalin selkeyttä.
- Transparency. Nimensä mukaisesti ominaisuus määrittää materiaalin läpinäkyvyyden.
- Refraction. Materiaalin ominaisuus, joka kertoo, kuinka valo taittuu osuessaan kappaleeseen.
- Specular. Ominaisuudet, jotka määrittävät kappaleen kiillon.
    - Specular color. Kiillon väri, joka on valosta riippuvainen.
    - Specular level. Aste, joka määrittää, kuinka intensiivinen kiilto on.
    - Glossiness. Ominaisuus, jonka avulla määritetään, kuinka kohtisuorassa valoon nähden kiilto tapahtuu.
    - Soften. Määrittää kiillon pehmeyden.
- Bump. Bumpin avulla materiaaliin saadaan luotua syvyyttä, kuten kuoppia tai muita tehosteita, joilla materiaali saadaan vaikuttamaan kolmiulotteiselta.



Kuva 5. Esimerkki siitä kuinka materiaalin voi luoda 3ds Maxissa.

Tämänkertaiseen työhön materiaaleja suunniteltaessa tärkeimpänä prioriteettina oli niiden selkeys ja yksinkertaisuus. Kyseessä oli esite, jolloin materiaalien tuli näyttää hyvältä, mutta kuitenkin selkeys ja riittävä yksinkertaisuus priorisoiden.

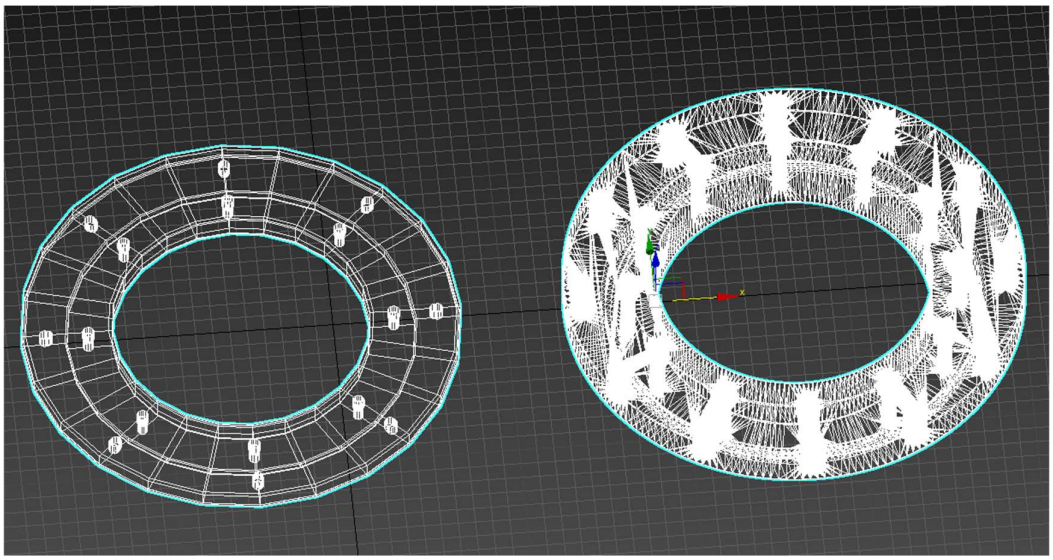
Aikaisemmissa projekteissa 3ds Maxin valmiit materiaalikirjastot ja niiden helppo ja hyvä muokattavuus olivat jo tulleet tutuiksi. Tämän vuoksi materiaalit suunniteltiin tehtävän 3ds Maxissa.

4 Esitteen luominen

4.1 Low poly -mallinnus dynaamista 3D-sisältöä varten

Niin kuin jo luvussa 2 mainittiin; kaikkien työssä käytettävien 3D-mallien tuli olla low poly-malleja. Tämä sen vuoksi, että esitteen tiedostokoko pysyisi mahdollisimman pienenä. Koska polygoni-mallit koostuvat polygoneista, tiedoston koko on suoraan verrannollinen käytettyjen polygonien määrään.

Asiakas antoi työtä varten käyttöön tarkat 3D-mallinsa kyseiseen työhön tarvittavista malleista STP-muotoisena. Nopeasti kuitenkin kävi selväksi, että näiyä malleja en voisi työssä käyttää[kuva 6]. Kokeilussa, jossa yksi yksinkertainen 3D-malli upotettiin PDF-tiedostoon, tiedostokoko ylitti jo 10 MT. Työhön sisältyi useampia kymmeniä malleja, niin joten näitä malleja käytettäessä tiedostokoko kasvaisi aivan liian suureksi.



Kuva 6. Polygonimäärän visuaalinen ero aidon sovittimesta tehdyn CAD-mallin ja low poly -mallin välillä

Tässä vaiheessa kävi ilmi, että kaikki 3D-mallit tulisi tehdä alusta asti uudelleen jäljitellen tarkempia malleja. Tällöin tuli myös koettaa selvittää, minkälainen tarkkuus olisi malleille riittävä. Tämän selville saamiseksi yhdestä mallista tuli tehdä useita eri versioita, toiset tarkempia kuin toiset. Nämä mallit vietiin mallintamisen jälkeen .obj-muodossa Photoshopiin, jossa niille luotiin materiaali, minkä jälkeen mallit upotettiin. Tällä tavalla sai hyvin selville, kuinka tarkasti PDF-piirtää 3D-mallit. Lopputulos näiden kokeilujen perusteella oli, että mallit saisivat olla aika yksinkertaisia, sillä tarkan mallin ja yksinkertaisen low poly -mallin silmin nähtävä ero oli PDF-tiedostossa todella pieni vaadittavaan tiedostokokoon nähden. [kuva 7.]

Kuva 7. Tarkan haarakappaleen mallin (26 910 polygonia) ja low poly -(1 720 polygonia) mallin ero PDF-tiedostossa

Malleja luotaessa myös liialliset yksityiskohdat, kuten kierteet ja urat, pyrittiin karsimaan malleista pois. Nämä yksityiskohdat eivät toisi esitteisiin riittävästi lisäarvoa siihen nähden, kuinka paljon niiden sisällyttäminen kasvattaisi tiedostokokoa.

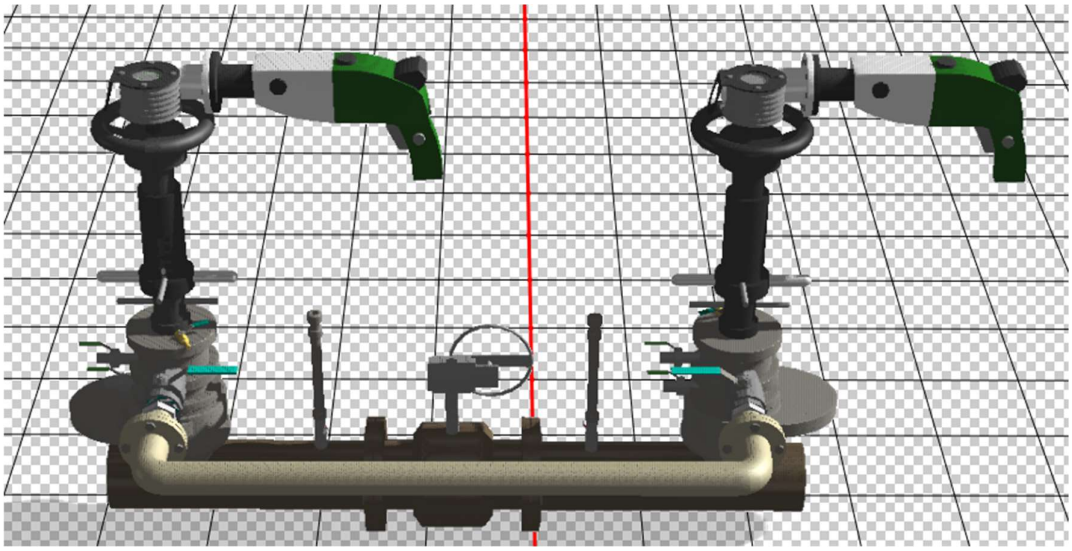
3ds Max

3D-mallit luotiin 3ds Max ohjelmalla, sillä sen oli todettu aikaisemmissa projekteissa olevan käypä ja tehokas tämänkaltaiseen työskentelyyn ja low poly -mallien luomiseen. 3ds Max mahdollistaa mallien luomisen usealla erilaisella tekniikalla, jotka on esiteltynä luvussa 3. Näiden tekniikoiden hyödyntämiseksi ohjelma sisältää kattavasti muokkaimia, jotka helpottavat työskentelyä ja tekevät työskentelystä myös tehokkaampaa. Näistä tekniikoista parhaiten tähän työhön soveltuva oli polygonimallinnus. Moni malli työssä koostui yksinkertaisista muodoista, joten mallit olivat nopea ja tehokas luoda polygon mallinnuksen ”boolean” tavalla kappaleita yhdistelemällä ja erottelemalla. Tästä hyvänä esi-

merkkinä on kuvan 7 haarakappale, joka koostuu kahdesta putkesta, jotka ovat yhdistettynä toisiinsa. Tästä kappaleesta on vielä erotettu 8 sylinteriä, jotta ’on saatu aikaan reiät pulteille. [kuva7.]

Photoshop

Mallit oli vietävä Acrobatiin Photoshopin kautta, jotta ne saatiin käännettyä Acrobatin tukemaan U3D-muotoon. Kun 3ds Maxilla luotu kokoonpanomalli oli valmis, avattaessa 3ds Maxin luomaa OBJ tiedostoa Photoshopilla, se avasi kaikki mallit yhtenä ”objMesh”-mallina. Tämä oli todella ikävä yllätys, sillä jokaisen mallin osalle piti pystyä luomaan omaa materiaalia, ja myöhemmin yksittäisiä malleja tulisi pystyä animoimaan Acrobatissa JavaScriptin avulla. Syytä siihen, miksi näin kävi ei löytynyt. Asia ratkesi loppujen lopuksi tuomalla Photoshopiin vain muutama malli kerrallaan, jolloin mallit pysyivät erillisinä. [Kuva 8.]

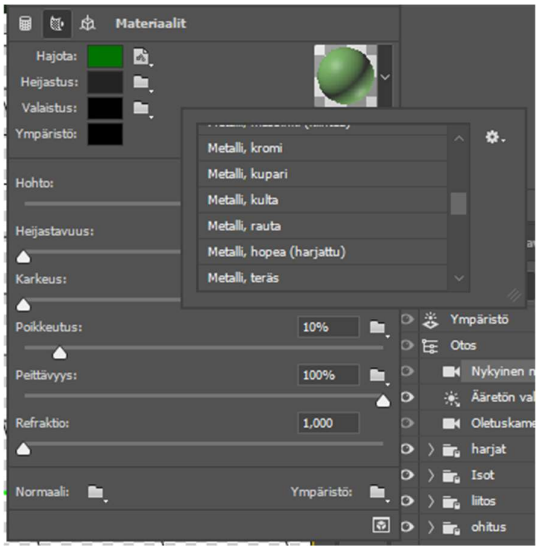


Kuva8. Kokoonpanomalli Photoshopissa.

Materiaalit malleihin

Mallien materiaalit luotiin alun perin aikaisemmissa projekteissa hyväksi todetulla 3ds Maxin materiaalieditorilla. Materiaalien luonnissa käytettiin hyväksi Autodeskin omia materiaalikirjastoja ja omaa aikaisempaa kokemusta ja tietämystä. Photoshop ei kuitenkaan kyennyt avaamaan 3ds Maxissa luotuja materiaaleja, jolloin materiaalit jouduttiin luomaan toiseen kertaan, tällä kertaa Photoshopissa. Aikaisempaa kokemusta materiaalien

luonnista Photoshopilla ei ollut, joten tämän opetteleminen toi hieman lisäviivästymistä työhön. Photoshopiin on kuitenkin 3ds Maxin tapaan saatavilla useita valmiita materiaalikirjastoja. Adoben internetsivuilta löytyi tähän työhön erittäin hyvin sopiva materiaalikirjasto [16], joka sisälsi esiasetukset [Kuva 8] muutamalle eri metallille ja muoville.



Kuva 9. Materiaalien esiasetusten valinta Photoshopissa.

Näiden esiasetusten avulla sai nopeasti ja suhteellisen pienellä vaivalla luotua malleille tarvittavat materiaalit ainoastaan ”diffuse”-ominaisuuksia muuttamalla. Materiaaleja luodessa tuli koko ajan pitää mielessä työn lopullinen käyttötarkoitus: Paineenalaisen putkiston tulppaamisen esittäminen selkeästi 3D-PDF:n avulla. Materiaalien täysi vastaavuus oikeaan maailmaan ei ollut tarkoituksenmukaista.

4.2 Animointi

Yleensä animoitaessa 3D-animaatioita käytetään 3ds Maxin kaltaista animointiin tarkoitettua ohjelmaa. Tähän työhön lähdettiin animoimaan animaatioita käyttäen 3ds Maxia. Muutaman yksinkertaisen animaation luomisen jälkeen animaatioiden JavaScriptiä pyrittiin viemään ulos 3ds Maxista Acrobatiin. Ikävä kyllä, Acrobat ei enää tukenut ASE-tiedostoja, joissa nimaatio olisi skriptattuna JavaScriptillä. Näin ollen työ seisahtui jälleen hetkeksi, jotta saatiin keksittyä, kuinka animaatiot tulisi nyt luoda.

Aloittaessa skriptin kirjoittamista tuli lähestymistapaa animaatioihin hieman muuttaa ja miettiä, mitä animaatio itse asiassa tarkoittaa. Animaatio on yksinkertaisesti vain tietyn

väliajoin muuttuvaa liikettä. Tämä ymmärrys animaatiosta oli eräänlainen ”heureka”-hetki, sillä tähän tarkoitukseen JavaScriptistä löytyy valmis funktio, setInterval(), joka toistaa käskyä, kunnes toisin sanotaan. Tämä funktio ottaa vastaan kaksi parametria;

- ajettava funktio eli funktio jota toistetaan
- aikaväli, jolla toisto tapahtuu.

Funktio pysäytetään käskyllä clearInterval(),joka vastaanottaa parametrina sen setInterval()-funktion nimen, joka halutaan pysäyttää.

setInterval()-funktioita kokeiltiin, ja sille annettiin parametriksi funktio, joka laskisi työssä esiintyviä kappaleita alaspäin aina kutsuttaessa tietyn määrän. Funktiossa on määritelty, että tietyt kappaleet liikkuvat aina yhden negatiivisen yksikön z-akselilla, kun funktiota kutsutaan. Ihmissilmä kykenee näkemään käsityksen mukaan 25 kuvaa sekunnissa, joten setInterval()-funktio määritettiin toistamaan lasku-funktiota 40 millisekunnin välein [koodiesimerkki 3], sillä 1 sekunti on 1 000 millisekuntia ja tämä jaettuna 25:llä on 40.

Asia, johon tuli animoidessa kiinnittää paljon huomiota, oli mallien eli objektien ryhmittäminen hyvin. Useamman kappaleen liikkuessa samalla tavalla samanaikaisesti olisi turhaa kirjoittaa jokaiselle omaa liikettä, vaan välttääkseen ylimääräisen skriptin kirjoittamisen kappaleet kannatti ryhmittää niin, että riitti, kun animoi vain ryhmän liikkumaan useamman erillisen kappaleen sijaan. Malleja ei kuitenkaan pysty ryhmttämään uudestaan enää Acrobatissa, joten ryhmät tuli olla tarkkaan mietittynä sen suhteen, ettei niissä kuitenkaan olisi liikaa kappaleita, jotka eivät aina liiku samansuuntaisesti. Tämä johti siihen, että kokoonpanomallia joutui pompottelemaan useaan otteeseen Photoshopin ja Acrobatin välillä ja tekemään pieniä muutoksia kappaleiden ryhmityksiin.

Ryhmitettäessä kappaleita ja animoitaessa kappaleita ryhmien sisällä haasteeksi muodostui usein, että liike tapahtuu aina suhteessa ylempään tasoon. Tämä tarkoittaa, että koordinaatisto, jossa liike tapahtuu, ei ole sidonnainen päätasoon, vaan kappaleen koordinaatisto on ryhmän sisäinen. Esimerkiksi, jos ryhmä on kallistettuna 45 astetta päätasoon nähden, ryhmän sisällä olevat kappaleet eivät tunnista tuota 45:tä astetta. Mikäli ryhmän sisällä olevia kappaleita haluaa liikuttaa päätasoon nähden, tuo 45 astetta tulee ottaa huomioon.

5 Ongelmia ja ratkaisuja

Projektin aikataulu pääsi venymään heti kun törmäsin ongelmiin projektin alusta lähtien. Tämä myös luonnollisesti heijastui asiakkaan saaman lisäarvon projektista ja projektin hinnan suhteeseen.

Alkuperäisten suunnitelmien mukaan minun oli tarkoitus käyttää hyväkseni asiakkaalta saamiani malleja, jotka kuitenkin osoittautuivat liian tarkoiksi tähän projektiin. Ongelmat asiakkaalta saamien mallien suhteen olivat kuitenkin helposti ratkaistavissa, vaikkakin lisätyötä mallien liiallisesta tarkkuudesta tähän työhön seurasi. Mallien suunnittelu ja uudelleen luominen vaati noin kahden viikon lisätyöt suunnitelmiin, mikä oli yksi syistä projektin pitkittymiseen.

Seuraavat lisätyötä aiheuttaneet ongelmat tulivat ohjelmien yhteensopivuuden kanssa. Kuten aikaisemminkin jo mainitsin, mallien materiaalit jouduin luomaan kahteen otteeseen, sillä aluksi ne luotiin 3ds Max ohjelmassa. Kuitenkin vietäessä malleja 3ds Maxista eteenpäin materiaalit eivät olleet yhteensopivia Photoshopin kanssa. Ratkaisua näiden materiaalien näkymiseen Photoshopissa etsittiin, mutta tuloksetta. Näin ollen materiaalit jouduttiin luomaan toistamiseen uudessa työympäristössä, mikä omalta osaltaan lisäsi jälleen työtunteja.

Samat yhteensopivuusongelmat jatkoivat projektin hidastamista, kun aikaisemmin Acrobatin tukema ASE-tiedostomuoto olisi sisältänyt myös animaatiot, mutta uusin Acrobatin versio ei enää tukenut tätä tiedostomuotoa. Tässä kohtaa ei ollut kuitenkaan keretty tekemään kuin muutama yksinkertainen animaatio, ennen kuin animaation ulosvientiä 3ds Maxista kokeiltiin. Animaatiota koetettiin viedä ulos 3ds Maxista useammassa eri muodossa, mutta yksikään näistä ei onnistunut tuomaan animaatioita ulos JavaScriptinä. Ratkaisu tähän ongelmaan löytyi suhteellisen helposti, vaikkakin työläästi, kirjoittamalla animaatiot käsin.

Ongelmat JavaScriptin kanssa

Acrobat tukee JavaScriptiä [koodiesimerkki 2.], mutta kaikki funktiot eivät ole aivan samoja kun tavallisesti. Tästä mainittakoon esimerkkinä funktio setTimeout(), joka yleensä

kirjoitetaan edellä olevalla tavalla, mutta Acrobatin kirjastosta funktio löytyy nimellä setTimeOut(). Nämä pienet eroavaisuudet joidenkin funktioiden kirjoitusasussa aiheuttivat välillä turhauttavia ongelmia.

```
,

pageIndex = this.pageNum;
annotIndex = 0;
c3d = this.getAnnots3D( pageIndex )[ annotIndex ].context3D; // Valitaan haluttu 3D sisältö
laskuri = 0;
Isot = c3d.scene.nodes.getByName( "Isot" ); // Valitaan haluttu objekti 3D sisällöstä
sulkuLiikeInt = new c3d.Vector3(0,0,-1); // Vektori, joka määrittää liikkeen
var lasku = function(){ // funktion nimeäminen
    Isot.transform.translateInPlace(sulkuLiikeInt); // Liikutetaan valittua sisältöä
                                                    // sulkuLiikeInt vektorin verran
    laskuri = laskuri + 1; // aina kun funktio ajetaan lisätään laskuriin 1
    if (laskuri == 50){     // Kun laskuri on 50 ajetaan tämä
        app.clearInterval(anim); // pysäytetään app.setInterval funktio
        laskuri = 0; // nollataan laskuri
        app.setTimeOut("nakyvyysPultit()", 1000); // Ajetaan seuraava funktio
        }
    }
```

Koodiesimerkki 2. Koodilohko, jota kutsuttaessa oliota ”Isot” liikutetaan alaspäin 50 kertaa, jonka jälkeen kutsutaan funktiota ”nakyvyysPultit()” sekunnin kuluttua.

Tietyt käskyt Acrobatin JavaScriptissä eivät myöskään toimineet aivan samalla tavalla Acrobatissa kuin yleensä. Esimerkkinä setInterval() funktio [koodiesimerkki 3.], jonka toimimaan saamisessa kohdattiin työn edetessä muutamia ongelmia.

```
anim = setInterval('lasku()',40);
```

Koodiesimerkki 3. Yleinen muoto ”setInterval()” funktiolle.

Toistuvista yrityksistä saada toistettua yllä esiteltyä funktiota Acrobat ei tunnistanut käskyä setInterval().

Aluksi selitystä sille, miksei tämä toiminut, ei löytynyt. Osa selityksestä löytyi Adoben JavaScript API-oppaasta. [11.] Sieltä löytyvän selityksen mukaan näitä perusfunktioita käyttääkseen tuli viitata JavaScript-sovellukseen ja tämä onnistui lisäämällä viitte app kirjastoon [koodiesimerkki 4].

```
anim = app.setInterval('lasku()',40);
```

Koodiesimerkki 4. Viittaus ”setInterval()” app-kirjastossa sijaitsevaan ”setInterval()”-funktioon.

Tämäkään ei vielä korjannut tilannetta, eikä selitystä tälle löytynyt mistään. Kirjoitusvirheiden välttämiseksi funktiot kirjoitettiin uudelleen, ja tällä kertaa app.setInterval() funktion sisällä olevan parametri funktio laitettiin kaksinkertaisten lainausmerkkien sisään yksinkertaisten sijaan ja jostakin syystä tämä toimi [koodiesimerkki 5]. Tämä ongelma ratkesi ikään kuin vahingossa.

```
anim = app.setInterval("lasku()",40);
anim = app.setInterval('lasku()',40);
```

Koodiesimerkki 5. Yllä oikeat kaksinkertaiset lainausmerkit ja alla yksinkertaiset.

Tämä ongelman ratkettua työ eteni mallikkaasti, sillä suurin osa animoinnista onnistui tätä yhtä ja samaa funktiota toistaen, muutamia parametreja muuttaen.

Animoinnin ja liikkeen luominen käsin oli yllättävänkin helppoa ja nopeaa, kun kyseessä oli suora liike x-, y- tai z-akselilla. Suurimmat ongelmat työssä tulivat kappaleiden pyörimisen kanssa. Acrobatissa pyöriminen tapahtuu oletusarvoisesti koordinaatisto pisteen 0,0,0 kautta. JavaScript 3D-API-oppaasta [12] löytyy funktio rotateAboutLineInPlace(), joka ottaa vastaan kolme parametria:

- Angle, jolla määritetään kulma, jonka verran kappale pyörähtää
- Start eli piste josta viiva, jonka suhteen kappale pyörii, alkaa
- End eli piste johon viiva, jonka suhteen kappale pyörii, loppuu.

Funktio rotateAboutVectorInPlace() vastaanottaa kaksi parametria;

- Angle, jolla määritetään kulma, jonka verran kappale pyörähtää.
- Axis eli vektoripiste, joka toimii pyörimisen keskipisteenä.

Näiden funktioiden avulla pyörimisen keskipistettä pystyy vaihtamaan, mutta käsin sen määrittäminen tarkasti avaruuskoordinaatistossa osoittautui itselleni toistaiseksi liian hankalaksi.

Tämän pyörimiskeskipisteen löytämistä vaikeutti entisestään se, ettei pistettä pystynyt koskaan näkemään. Koetin löytää tavan, jolla olisin voinut luoda uuden ”dummyn”, eli eräänlaisen apuobjektin aina siihen pisteeseen, jonka suhteen kappale pyöri. Acrobatissa ei kuitenkaan voinut luoda uusia objekteja, joten konkreettisesti pistettä tai viivaa, jonka ympärillä kappaleet pyörivät, ei pystynyt ikinä näkemään. Pisteen hahmottamista vaikeutti erityisesti myös jo aikaisemmin mainittu koordinaatistojen käyttäytyminen ryhmissä: ne ovat sidonnaisia ryhmään eivätkä maailmaan. Käsitykseni mukaan kuitenkin vektori, jonka koordinaatit yllä mainitut funktiot ottavat vastaan olivat sidonnaisia maailmaan. Tämä johti siihen, että kappaleet, jotka olisi pitänyt saada pyörimään tarkasti oman akselinsa ympäri, pyörivät joka kerta hieman vinosti. Tämä ongelma jäi vielä toistaiseksi ratkaisematta.

Ongelmia ajantasaisen tiedon kanssa

Tämän työn useassa eri vaiheessa törmättiin isoon ongelmaan ajantasaisen tiedon kanssa. Erityisesti sen kanssa joutui ongelmiin etsiessäni tietoa Acrobatista. Ensimmäiset vanhat tiedot, jotka eivät pitäneet paikkansa kohdattiin jo suunnitteluvaiheen alussa yrittäessäni löytää tietoa tuetuista 3D-geometriaa sisältävistä tiedostomuodoista. Tällöin törmättiin usein tietoon, että Acrobat tukisi laajalti useita erilaisia 3D-geometriaa sisältäviä tiedostomuotoja [16]. Tämä tieto ei kuitenkaan pitänyt paikkaansa enää tässä työssä käytetyn Acrobat-version kanssa. Adoben internetsivuilta [17] löytyy myös useita kuolleita linkkejä JavaScript materiaaleihin. Ainut linkki toimivaan JavaScript 3D-API-oppaaseen johti JavaScript 3D-API-oppaaseen vuodelta 2007. Se sisälsi vanhentunutta tietoa. Tämä vanhentuneeseen tietoon päätyminen oli välillä todella turhauttavaa.

6 Yhteenveto

Insinöörityön tarkoituksena oli selvittää, olisiko jo pitkään käytössä olleita PDF-esitteitä mahdollista parannella käyttäen saatavilla olevia nykyaikaisia tekniikoita. Vaikka esitystekniikat ja digitaalinen media ovat olleet jatkuvassa ja nopeassa kehityksessä jo todella pitkään, esitteet alalla ovat pysyneet samanlaisina.

Tässä työssä tutkittu, pitkään käytössä olleen tekstistä ja still-kuvista koostuvan PDF-tiedoston korvaaminen PDF-tiedostolla, joka sisältäisi dynaamista 3D-grafiikkaa, osoittautui osittain toimivaksi. Dynaaminen 3D-sisältö selkeytti asian esille tuomista enemmän kuin alun perin odotinkaan. Tämä johtui suurelta osin siitä, että aikaisemmin esitteissä saattoi olla useita kymmeniä kuvia tehtävän työn eri vaiheista. Jotkin kuvat saattoivat olla läpileikkauskuvia edeltävästä kuvasta ja tekstissä kerrottiin, mitä kuvien välillä tapahtuu. Dynaamisen 3D-sisällön avulla selittämisen tueksi pystyi selkeästi esittämään myös kappaleiden liikkeen näyttämään mitä tapahtuu. Samalla erilliset läpileikkauskuvat oli mahdollista jättää pois, sillä Acrobat tukee 3D-mallien läpileikkausta, joten näkymää pystyy vaihtelemaan normaalin ja läpileikatun välillä.

Työ koostui enimmäkseen minulle jo tutuista tekniikoista, low poly -mallintamisesta ja JavaScriptistä. Silti törmäsin työssä jatkuvasti pieniin ongelmiin, jotka viivästyttivät projektin valmistumista. Suurimman osan ongelmista onnistuin selvittämään kuitenkin hyvin. Työn valmistuminen viivästyi parilla viikolla, eikä viimeistä ongelmaa ole vielä ratkaistu, mutta työ on valmistuessaan omasta mielestäni tästä huolimatta onnistunut.

Aikaisemmin mainitsin, että vanhojen 2D-sisältöä ja tekstiä sisältävien PDF-tiedostojen korvaaminen dynaamista 3D-sisältöä sisältävillä PDF-tiedostoilla osoittautui vain osittain toimivaksi. Osittain toimiva johtuu siitä, että tällä tavalla toteutettavaan esitteeseen vaaditaan liikaa työtä verrattuna perinteiseen. Dynaamisen 3D-sisällön sisällyttäminen PDF-tiedostoon vaatii osaamista low poly -mallintamisesta ja kattavaa osaamista JavaScriptistä. Näiden asioiden osaamisen yhdistäminen työhön vaadittuun aikaan esitteiden luomisessa nostaa kustannukset niin korkealle, ettei dynaamista 3D-sisältöä sisältävien esitteiden luoma lisäarvo ole asiakkaalle kannattavaa tällä alalla.

Koska isoin työ oli mallien luomisessa, uskoisin kuitenkin 3D-PDF-esitteiden määrän kasvavan tulevaisuudessa. Isoin haaste on yleisten standardien puute. Esimerkiksi Adoben Acrobatilla luotua 3D-PDF-tiedostoa ei ole mahdollista lukea muiden valmistajien

PDF-lukijoilla. Tämä on suuri ongelma yleistymisen tiellä, sillä universaalius ja käyttöliittymä riippumattomuus on muuten PDF-tiedostojen suurin etu verrattuna muihin esitysmuotoihin.

Näkisin, että dynaamista 3D-grafiikkaa sisältävät PDF-esitteet lisääntyvät erityisesti kulutusasiakkaille suunnatussa esitysmateriaalissa, jossa mallit olisivat mahdollisesti jo valmiina. Lisäarvon määrän ei tarvitse olla silloin yksittäisen tuotteen kohdalla niin suuri, kun se jakautuu suuremmalle määrälle tuotteita, joiden myyntivolyymi on kattava, ja tätä myötä lisäarvo myös kertautuu. Näin tiedon jakamisen aikakautena uskoisin myös Acrobatin JavaScriptin kirjoittamisen helpottuvan ja nopeutuvan, sillä nyt internetissä ei ollut kovin paljoa esimerkki koodilohkoja, joista ottaa oppia ja joita olisi mahdollisesti voinut käyttää itsekin. Tällaiset esimerkkikoodilohkot ovat yleensä erittäin yleisiä ohjelmoijien parissa, joten uskoisin niitä ilmestyvän lisää ajan mittaan sitä mukaa, mitä enemmän tämäntyyppistä skriptiä kirjoitetaan.

Lähteet

1 The history of Microsoft Word – How the Microsoft Word Processor got its start. Verkkodokumentti. Brighthub. <http://www.brighthub.com/computing/windows-platform/articles/46978.aspx>. Luettu 26.4.2017.

2 What is Microsoft Powerpoint? – How Do I Use PowerPoint. Verkkodokumentti, ThoughtCo. <https://www.thoughtco.com/how-do-i-use-powerpoint-2767371>. Luettu 26.4.2017.

3 PDF-Reference – sixth edition, Adobe® Portable Document Format Version 1.7 November (2006). 2006. Adobe Systems Inc.

4 The history of PDF-| How the file format and Acrobat evolved. Verkkodokumentti. Prepressure. <https://www.prepressure.com/PDF/basics/history/5>. Luettu 28.3.2017.

5 Mikä PDF-on? Verkkodokumentti. Adobe Systems Inc. <https://acrobat.adobe.com/fi/fi/why-adobe/about-adobe-PDF.html>. Luettu 28.3.2017.

6 JavaScript. Verkkodokumentti. Wikikirjasto. <https://fi.wikibooks.org/wiki/JavaScript>. Luettu 28.3.2017.

7 JavaScript – MDN. Verkkosivusto. Mozilla <https://developer.mozilla.org/en-US/docs/Web/JavaScript>. Luettu 26.4.2017.

8 Acrobat JavaScript Samples Scripts. Verkkodokumentti. Evermap <https://www.evermap.com/javascript.asp>. Luettu 28.3.2017

9 JavaScript™ for Acrobat® 3D-Annotations API Reference (2015). 2015. Adobe Systems Inc.

10 JavaScript™ for Acrobat® API Reference (2007). 2007. Adobe Systems Inc.

11 Pöllänen Joel. 2015. Rakennusten 3D-mallinnus ja mallien käyttäminen WebGL-sovelluksessa. Insinöörityö. Metropolia ammattikorkakoulu.

12 Exporting to ASCII | 3ds Max | Autodesk Knowledge Network. Verkkodokumentti. Autodesk Inc. <https://knowledge.autodesk.com/support/3ds-max/learn-explore/caas/CloudHelp/cloudhelp/2016/ENU/3DSMax/files/GUID-98B2388D-A3A7-4096-8E81-888A3F9D6069-htm.html>. Luettu 5.4.2017.

13 Polygon modeling | Maya | Autodesk Knowledge Network. Verkkodokumentti. Autodesk Inc <https://knowledge.autodesk.com/support/maya/learn-explore/caas/CloudHelp/cloudhelp/2016/ENU/Maya/files/GUID-7941F97A-36E8-47FE-95D1-71412A3B3017-htm.html>. Luettu 3.5.2017.

14 NURBS Model | 3ds Max | Autodesk Knowledge Network. Verkkodokumentti. Autodesk Inc. <https://knowledge.autodesk.com/support/3ds-max/learn-explore/caas/CloudHelp/cloudhelp/2018/ENU/3DSMax-Reference/files/GUID-2DEA769A-B2DC-4846-8F3F-3049F5EE1778-htm.html>. Luettu 26.4.2017.

15 Surface Modeling | 3ds Max | Autodesk Knowledge Network. Verkkodokumentti. Autodesk Inc. <https://knowledge.autodesk.com/support/3ds-max/learn-explore/caas/CloudHelp/cloudhelp/2018/ENU/3DSMax-Modeling/files/GUID-C7F01818-502C-499B-8FD5-1F6742298DC7-htm.html>. Luettu 26.4.2017.

16 Downloadable 3D-content | PhotoShop. Verkkodokumentti. Adobe Systems Inc. <http://www.photoshop.com/products/photoshop/3d/content>. Luettu 5.4.2017.

17 What 3D-formats can Acrobat 3D-import. Verkkoartikkeli. Adobe Systems Inc. <http://blogs.adobe.com/mfg/2006/11/what_3d_formats_can_acrobat_3d.html>. Luettu 20.3.2017.

18 JavaScript for Acrobat | Adobe Developer Connection. Verkkosivu. Adobe Systems Inc <http://www.adobe.com/devnet/acrobat/javascript.html>. Luettu 20.4.2017.

Esimerkki Acrobatin JavaScript-funktiosta

Funktiota kutsuttaessa se animoi tulppauksessa tapahtuvia porausvaiheen liikkeitä.

```
pageIndex = this.pageNum;
annotIndex = 0;
c3d = this.getAnnots3D( pageIndex )[ annotIndex ].context3D;
laskuri = 0;
porakoneet = c3d.scene.nodes.getByName( "porakoneet" );
porakoneA = c3d.scene.nodes.getByName( "porakoneA" );
porakoneB = c3d.scene.nodes.getByName( "porakoneB" );
porausA = c3d.scene.nodes.getByName( "porausA" );
porausB = c3d.scene.nodes.getByName( "porausB" );
akseliA = c3d.scene.nodes.getByName( "akseliSaha" );
akseliB = c3d.scene.nodes.getByName( "akseliSaha_0" );
syotto = c3d.scene.nodes.getByName( "SyottoUlko_0" );
bodyA = c3d.scene.nodes.getByName( "bodyA" );
bodyB = c3d.scene.nodes.getByName( "bodyB" );
adapteriReikaA = c3d.scene.nodes.getByName( "adapteriReika" );
adapteriReikaB = c3d.scene.nodes.getByName( "adapteriReika_0" );
sulkulevyB = c3d.scene.nodes.getByName( "sulkuLevyB" );
isokahvaA = c3d.scene.nodes.getByName( "isoKahvaA" );
isokahvaB = c3d.scene.nodes.getByName( "isoKahvaB" );

poraLiikeInt = new c3d.Vector3(0,-1,0);
poraLiikeInv = new c3d.Vector3(0,1.75,0);
sulkuLevyLiikeInt = new c3d.Vector3(0,0,3);

var poraus = function(){
                porakoneA.transform.translateInPlace(poraLiikeInt);
                porakoneB.transform.translateInPlace(poraLiikeInt);
                porausA.transform.translateInPlace(poraLiikeInt);
                porausB.transform.translateInPlace(poraLiikeInt);
                akseliA.transform.rotateAboutYInPlace( Math.PI / 7 );
                laskuri = laskuri + 1;
                if (laskuri == 75){
                                poraLiikeInt = new c3d.Vector3(0,-3,0);
                                }
                else if(laskuri == 90){
                                app.clearInterval(anim);
                                laskuri = 0;
                                app.setTimeOut("nostoAlku()",1500);
                }
}
var nostoAlku = function() {
                anim = app.setInterval("nosto()",40);
}
var nosto = function(){
                porakoneA.transform.translateInPlace(poraLiikeInv);
                porakoneB.transform.translateInPlace(poraLiikeInv);
                porausA.transform.translateInPlace(poraLiikeInv);
                porausB.transform.translateInPlace(poraLiikeInv);
                laskuri = laskuri + 1;
                if(laskuri == 150){
                                app.clearInterval(anim);
                                laskuri = 0;
                                app.setTimeOut("sulkuAlku()",1000);
                }
}
var sulkuAlku = function() {
                anim = app.setInterval("sulku()",40);
}
var sulku = function(){
                sulkulevyB.transform.translateInPlace(sulkuLevyLiikeInt);
                laskuri = laskuri + 1;
                if(laskuri == 60){
                                app.clearInterval(anim);
                                laskuri = 0;
                }
}
var start = function(nimi) {
                anim = app.setInterval("poraus()",40);
}
app.setTimeOut("start()",500);
```