OULUN AMMATTIKORKEAKOULU

Tero Pelkonen

Technical Artist -työrooli

Päiväkirjamuotoinen opinnäytetyö

Technical Artist -työrooli

Päiväkirjamuotoinen opinnäytetyö

Tero Pelkonen
Opinnäytetyö
Kevät 2019
Tietojenkäsittely
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietojenkäsittelyn tutkinto-ohjelma

Tekijä(t): Tero Pelkonen
Opinnäytetyön nimi: Technical Artist -työrooli
Työn ohjaaja: Eero Leskinen
Työn valmistumislukukausi ja -vuosi: Syksy 2019 Sivumäärä: 89

Tässä päiväkirjamuotoisessa opinnäytetyössä seurataan aloittelevan Technical Artistin työtehtäviä 10 viikon ajan yrityksessä, joka sijaitsee Oulussa. Päiväkirjamuotoista opinnäytetyötä kirjoitetaan päivittäin, ja viikon lopuksi kirjoitetaan viikkoanalyysi viikon tapahtumista sekä viikolla nousseisiin eri aiheisiin, joita käsitellään tarkemmin. Opinnäytetyöllä ei ole toimeksiantoa, ja se tehdään töiden ohella. Työtehtävät vaihtelevat 3D artistin sekä ohjelmistokehittäjän välillä.

Yrityksen antama työtehtävä on luoda heille uusi työkalu. Se luotiin 10 viikon aikana Python ohjelmointikielellä, ja sillä voi viedä 3D-esityksiä Maya-ohjelmasta Qt:n 3D Studio työkaluun. Päiväkirjan kirjoittamisen alussa opinnäytetyön kirjoittajalla oli vain vähän kokemusta Maya-ohjelmasta sekä Pythonin käyttämisestä.

Tietoperustana hyödynnetään käytettävien ohjelmien kotisivuilta sekä eri ohjelmointikieliin erikoistuneiden sivujen dokumentaatiota. Nämä ovat samoja lähteitä, joista tietoa on haettu oikeita työtehtäviä varten.

Opinnäytetyön tavoitteena on antaa tuleville Technical Artist -työrooliin haluaville kattavan kuvauksen työarjesta ja sen mahdollisista haasteista. Näistä tietäminen voi auttaa samaan työkuvaan valmistautumisessa, sillä rooli ei ole aina perinteisen 3D-mallintajan tai ohjelmistokehittäjän tapainen, vaan se voi olla sekoitus molempia.

Asiasanat: Technical Artist, 3D-mallinnus, Ohjelmistokehitys, Python, Maya, Qt

ABSTRACT

Oulu University of Applied Sciences
Degree of Bachelor of Information Technology

Author(s): Tero Pelkonen
Title of thesis: Technical Artist work role
Supervisor(s): Eero Leskinen
Term and year when the thesis was submitted: Autumn 2019 Number of pages: 89

This thesis is written in diary form and it focuses on following a beginner Technical Artist for 10 weeks, who works in IT-company in Oulu. Diary form thesis is written daily and at the end of the week there will be a week analysis of the work, achievements and the themes the week has had. There is no assignment for this thesis, and it is written along the work. Work tasks go from 3D artist to software developer.

Company gave thesis writer a task to create a new tool. This was done during those 10 weeks by using Python coding language. It can be used to export 3D scenes from Maya to Qt's 3D Studio. At the beginning of writing the thesis the writer had low knowledge of Python and just some experience with Maya.

Software's homepages and different programming language pages will be used as knowledge base. These are the same sources where the actual information would be searched for real tasks.

Purpose of this thesis is to be able to give a glimpse of the actual work that comes with Technical Artist work role. It can help those who are interested in this role to prepare for it. The role is not of a typical 3D-artist or software developer but can include influences from both.

Keywords: Technical Artist, 3D modelling, Software development, Python, Maya, Qt

SISÄLLYS

1 JOHDANTO

Sain tilaisuuden tehdä kesätöitä Oulussa sijaitsevassa Qt Companyn toimistossa. Yritys on perustettu vuonna 1995. Se toimi aluksi nimellä Trolltech, minkä jälkeen Nokia osti sen vuonna 2008. Vuosina 2011 ja 2012 Digia hankki Qt:n omistukseensa osissa, ja vuonna 2016 Qt yhtiöitti Qt Groupin tytäryhtiöksi Digiasta. Tällä hetkellä sillä on 45,6 miljoonaa euroa liikevaihtoa ja se työllistää noin 300 henkeä 12 maassa.

Qt on alustariippumaton ohjelmistojen sekä käyttöliittymien kehitysympäristö valmiilla käyttöliittymäelementeillä, C++ -kirjastoilla ja työkaluilla. Ympäristö sisältää työkalut sekä kehittäjille että suunnittelijoille. Qt kehitysympäristöllä voidaan tehdä ohjelmia ja käyttöliittymiä tehokkaille ja kevyemmille laitteille. (Qt Company 2019, viitattu 15.7.2019.)

Oulun toimistossa toimii yrityksen useita kehitysyksikköjä sekä tiimi, joka kehittää uutta työkalua nimeltä Qt 3D Studio. Sillä voidaan luoda näyttäviä 3D-esityksiä sekä käyttöliittymiä. Oulun toimistossa työskentelee tällä hetkellä noin 60 henkeä.

Työtehtäväni on kehittää Autodesk Maya-työkalua 3D Studio-tiimille. Sillä pitäisi voida tuoda helposti Maya 3D-mallinnustyökalusta 3D-mallit, valot, kamerat ja muut objektit Qt 3D Studioon. Työkalu olisi Mayassa Python- ja MEL-ohjelmointikielellä tehty skripti, jonka pitäisi toimia ohjelmassa lisäosan tapaisesti.

Tämä tehtävä vaatii Python-ohjelmointitaitojen lisäksi 3D-mallinnuksen ymmärtämistä, nopeaa kykyä oppia uusia ohjelmistoympäristöjä, tiedonhakua sekä ryhmätyötyötaitoja. Suurin osa tiedosta löytyy tiimin muilta jäseniltä sekä internetin ohjelmakohtaisista keskustelufoorumeista.
Autodeskin Maya 3D-mallinnustyökalu ei ole minulle entuudestaan tuttu, vaan olen aikaisemmin käyttänyt pelikehittäjäpiireissä yleistä Blenderiä sekä hieman Autodeskin 3DS Maxia.

Jännittävää tässä työtehtävässä on se, että koulutukseni on painottunut paljon verkko-ohjelmointiin, ja tätä työtehtävää varten olen yrittänyt opiskella ylimääräisenä aiheena 3D-mallintamista sekä ohjelmointia omasta mielenkiinnostani. En ollut aikaisemmin törmännyt Technical Artist -työnimikkeeseen, eikä se ollut muutenkaan tuttu tai yleinen. Tämän vuoksi ajattelin, että työtehtävästä kertominen olisi hyvä aihe opinnäytetyölle.

Päiväkirjamuotoinen osuus opinnäytetyöhön kirjoitetaan 6.5.2019–9.8.2019 välisenä aikana. Päivittäistä sekä viikkoanalyysiä kirjoitetaan 10 viikon ajan. Päivittäisessä päiväkirjaosuudessa käyn läpi päivän tavoitteita, tapahtuneita työtehtäviä, työympäristöä sekä työhön liittyviä ajatuksia. Viikoittaisessa analyysissä käyn läpi mennyttä viikkoa, sen haasteita ja mitä olen oppinut. Viikkoanalyysiin sisältyy teoriaa kullakin viikolla esille nousseeseen teemaan liittyen.

2 NYKYTILANTEEN KUVAUS

2.1 Oman nykyisen työn analyysi

Työnkuva sekä ohjelmistoala olivat minulle täysin uusia. Olen aikaisemmin työskennellyt enemmän laitteiden ja niihin liittyvien ohjelmistoympäristöjen parissa asiakaspalveluhenkisesti. Nyt minun tehtävänäni on luoda minulle uuteen ohjelmaan uusi työkalu. Olen löytänyt paljon yhtäläisyyksiä edellisen työurani sekä nykyisen työkuvan kanssa, ja tämä on auttanut uusien asioiden opettelussa. Technical Artistin roolissa voi tehdä päivittäin seuraavia asioita:

- Tavoitteiden jakaminen pieniksi tehtäviksi. Nämä merkataan yrityksen ja tiimin sisäiseen tehtäväjärjestelmään.
- Ohjelmointi, skriptaus tai koodaus. Koodikielen tulkinta ja kirjoittaminen.
- 3D-mallinnus ja työskentely. Mahdollista käyttää useita eri ohjelmia.
- Ohjelmaympäristöihin tutustuminen. Eri ohjelmilla on useita eri ominaisuuksia ja työkaluja, joiden tietäminen voi olla hyödyllistä tai välttämätöntä. Koskee erityisesti 3D-mallinnusohjelmia.
- Yrityksen sisäiseen viestintään osallistuminen. Sähköpostit ja tiimipalaverit.
- Koulutukset. Verkossa videolta tai firman kokoushuoneissa.
- Tiedonhaku. Verkosta tai työkavereilta. Verkosta löytyvät API (suom. ohjelmointirajapinta) -kirjastot ovat välttämättömiä.

Tyypillisenä työpäivänä valitsen jonkin tehtävän omasta tehtävälistasta. Tämän jälkeen keskityn asian selvittämiseen hakemalla tietoa ja koodaamalla pieniä testejä nähdäkseni vaikutukset. Parhaimmat ohjeet löytyvät verkosta eri keskustelufoorumeilta. Noilla ohjeilla voin hakea tarkempaa tietoa API-kirjastoista, ja luoda koodia, joka toimii haluamallani tavalla. Sovellan onnistuneita testejä tehtävän tavoitteeseen, ja sen onnistuessa uusi koodi lisätään pääkoodiin.

Työssäni on myös tärkeää yleisesti omaksua uusia asioita hyvin nopeasti, ja olen kehittänyt tähän oman tapani. Kirjoitan kaikki omasta mielestä tärkeät huomiot muistilapuille, ja tällä tavoin eri aiheisiin tulee pysähdyttyä ja mietittyä, mitä ne tarkoittavat. Myös sosiaaliset taidot ovat tärkeitä, sillä ympärilläni on ihmisiä, joilla on kymmenien vuosien työkokemus. Oikein kysyttynä heiltä voi saada parasta apua työtehtäviin.

Pääohjelmointikielenä käytän Pythonia, joka on hyvin yleinen, mutta koulutukseni ei keskittynyt tähän. Toinen erityisesti tähän työhön liittyvä ohjelmointikieli on MEL, joka on Mayalle luotu oma kielensä. 3D-mallinnukseen käytän pääasiassa Autodesk Mayaa, jolle lopullinen työkalu pitää luoda. Joskus käytän myös Blender työkalua, sillä se on toinen hyvin yleinen 3D-mallinnusohjelma ja jolla on nopea mallintaa yksinkertaisia muotoja. Työkielenä käytän suomea sekä englantia. Yritys on hyvin kansainvälinen, joten englannin kielen taito pitää olla hyvä.

Ohjelmoijana olen aloittelija. Minulla on kuitenkin paljon eri taustaista työhistoriaa, joka paikkaa tätä puutetta. Työtehtävä on mielestäni haastava, ja sen tekeminen on hyvin itsenäistä. Tuo itsenäinen tekeminen hieman yllätti, sillä olen tottunut, että minulle annetaan työtehtävät. On ollut kuitenkin mukava huomata, että löydän itse uusia tehtäviä ja ohjaudun niihin automaattisesti.

Aikaisempi työkokemukseni auttaa minua uusien ohjelmien käytössä, tiedon hakemisessa ja ongelmien tulkinnassa. Käytettävät ohjelmat ovat olleet minulle uusia, mutta tunnistan niistä hyvin nopeasti minulle oleelliset osiot ja toiminnot. Näin en hukkaa keskittymistä ylimääräisiin asioihin. Tiedonhaussa yritän ensin oppia itse ja tarvittaessa etsin oikean henkilön kenestä voi olla apua. Tärkeää on, että on ensin itse tehnyt jotain, jonka jälkeen toisen näyttämästä asiasta voi oppia paremmin, kun ymmärtää asian taustoja. Ongelmien tulkinnassa huomaan nopeasti syy-seuraussuhteen, kun jokin ei toimi oikealla tavalla. Etsin vaikuttavat tekijät, ja niitä testaamalla yritän löytää muuttujan, joka aiheuttaa ongelman. Aikaisemmassa asiakaspalvelutyössä tein tätä päivittäin, ja samat metodit tuntuvat pätevän ohjelmointiin.

Technical Artist -työrooli on minusta mielenkiintoinen, ja siihen tuntuu kuuluvan monia osa-alueita. Työtehtävien ohella minun pitää laajentaa 3D- sekä ohjelmointitaitoja. Minusta tuntuu, että tätä roolia varten on tärkeämpää oppia vähän kaikesta, kuin keskittyä ja erikoistua muutamaan asiaan.

2.2 Sidosryhmät työpaikalla

Työroolini suorittaminen on hyvin itsenäistä tai on ainakin tähän asti ollut. Siitä huolimatta työpaikalla on tietenkin sidosryhmiä, joiden kanssa olen toiminnassa. Sisäiset sidosryhmät ovat minulle esimieheni, Buddy-työkaveri, tiimijäseneni ja muiden tiimien jäsenet. Yrityksessä ja etenkin Oulun

osastossa on useita tiimejä, jotka keskittyvät eri tuotteiden ja palveluiden kehittämiseen sekä ylläpitoon. Ulkoisiin liitosryhmiin kuuluvat asiakkaat ja yhteistyökumppanit. Sidosryhmät on kuvattu kuviossa yksi.

KUVIO 1. Sidosryhmät

Esimieheltäni saan päätyötehtävän ja tarvittaessa ohjausta sen suorittamiseen. 3D Studio tiimissä on yhdeksän jäsentä. Työtehtäväni on irrallinen muiden 3D Studio tiimin jäsenten kanssa paitsi Buddyn. Hän on tiimin vanhempi Technical Artist, ja hänet määrättiin minulle Buddy-työkaveriksi. Tämä on yrityksen sisäinen käytäntö uusien työntekijöiden perehdyttämisessä. Häneltä saan ohjausta ja tarvittaessa apua teknisiin ongelmiin. Hänellä on kuitenkin omat tehtävänsä, joten meidän ei tarvitse tehdä jatkuvasti yhteistyötä. Tiiminjäseniltäni saan tarvittaessa teknistä apua ja työskentelemme samoissa tiloissa, joten pidämme hyvää työilmapiiriä. Muiden tiimien kanssa tulee toimittua pääasiassa kahvitiloissa, ja tällä hetkellä meillä ei ole yhteistyökuvioita. Arvelen kuitenkin, että jotain yhteistyötä on luvassa jossakin vaiheessa.

Tekemäni työkalu päätyy jossakin vaiheessa asiakkaille käyttöön. Saamani palautteen perusteella tiedän, mihin suuntaan työkalua kannattaa kehittää. Tämän hetkinen yhteistyökumppani on Technopolis, jolta vuokraamme työtilat. Tarvittaessa heiltä saa työympäristöön liittyviä palveluita.

2.3 Vuorovaikutustaidot työpaikalla

Vuorovaikutustaidot ovat tärkeitä työroolissani, vaikka työtehtävät suoritetaan itsenäisesti. Monen ongelman kanssa ratkaisu löytyy nopeasti, kun ongelmaa pohtii vieruskaverin kanssa ääneen. Tässä pitää ottaa huomioon työkaverin keskittymistilanne, ettei vain sattuisi häiritsemään. Tärkeimmät vuorovaikutustilanteet ovat esimiehen sekä Buddyn kanssa. Esimieheltä saadaan pääsuunnat tekemiselle, ja Buddy ohjaa tekemistä. Näistä keskusteltaessa pitää kuunnella tarkasti, mitä he kertovat ja esittää tarvittavat lisäkysymykset. Tiimimme pitää maanantaisin ja keskiviikkoisin seisottavan tiimipalaverin, jossa käydään läpi jokaisen jäsenen tehtyjä ja tulevia tehtäviä.

Kun olen saanut tekemäni työkalun siihen pisteeseen, että se julkaistaan asiakkaille käyttöön, voin joutua asiakaspalveluhenkisiin vuorovaikutustilanteisiin. Näissä on tärkeää, että osaa olla virallinen, hillitty ja osaa antaa itsestä ja edustettavasta yrityksestä hyvän vaikutelman.

Pidän itse ensivaikutelman antamista tärkeänä taitona. Uuteen työpaikkaan kannattaa mennä virkeänä ja raikkaana. Työpukeutuminen on todella vapaata Suomessa, mutta näin ei ole kaikkialla maailmassa. Kun työskentelee kansainvälisessä yrityksessä, niin on todennäköistä, että on tekemissä ulkomaisten yrityksien kanssa. Tämän vuoksi on järkevää panostaa myös työvaatteisiin.

Toimistollamme pidetään joka perjantai aamukahvikeskustelu, jossa käydään läpi yrityksen sekä toimiston asioita. Halutessaan kuka tahansa voi tuolloin kysyä asioista ja kommentoida. Mielestäni tällainen toiminta auttaa toimistoa pitämään matalaa hierarkiaa, jolloin jokainen voi halutessaan vaikuttaa asioihin.

3 PÄIVÄKIRJA

3.1 Ensimmäinen viikko: Tutustuminen, 3D-mallintaminen ja Python-ohjelmointi

Maanantai 6.5.2019

Ensimmäinen työpäivä uudessa työpaikassa on aina hyvin jännittävää, enkä aseta sille tavoitteita. Päästyäni toimistolle etsin esimieheni, ja hän näytti minulle työpisteeni. Se sijaitsi 3D Studio-tiimistä kaukana tilajärjestelyiden takia. Työkoneeksi sain Mac-tietokoneen, jollaista en ollut aikaisemmin käyttänyt. Yrityksellä on käytössä tarkistuslista uusille työntekijöille, jonka avulla perehdytys voidaan aloittaa. Minulle määrättiin myös Buddy-työkaveri, joka toimii läheisenä perehdyttäjänä esimiehen lisäksi. Hän on tiimin ainoa Technical Artist ja vastannut Maya työkalun tekemisestä tähän mennessä.

Ensimmäisenä kävimme läpi työpaikan yleisiä käytäntöjä koskien työ- ja taukotiloja, lounaita sekä työaikaa. Tämän jälkeen aloitin tarvittavien sovelluksien asentamisen sekä käyttötilien luomisen. Itselle entuudestaan tuntemattoman käyttöjärjestelmän käyttäminen oli hieman normaalia hitaampaa. Sain myös kutsun henkilöstökokoukseen, ja kävimme siellä läpi tarkemmin lakisääteisiä asioita kuten työterveysetuja, vakuutukset ja palkanmaksun.

3D Studio tiimissä käytetään viikoittain kahdesti Scrumin jotain muotoa projektinhallintaan, ja tämä näkyi kokouksena keskellä päivää. Scrum on projektinhallinnan viitekehys, joka on yleinen nykyään ketterässä ohjelmistokehityksessä. Näkyville tuotiin Kanban taulu, jossa näkyi eri työtehtäviä. Jokainen henkilö kertoi viimeksi tehdystä työstä sekä siitä, mitä aikoo seuraavaksi tehdä, jotta työtehtävä saataisiin valmiiksi. Omalla vuorollani esittäydyin tiimille.

Loppupäivästä uppouduin Maya-tutoriaaleihin, ja tein itselleni muistiinpanoja. Verkossa on tarjolla useampia videotutoriaalipalveluita, joissa opetetaan lyhyesti ja tehokkaasti eri työkalujen käyttöä, sekä siihen liittyvää teoriaa. Valitsin Pluralsight-sivuston, sillä Buddy suositteli sitä. Ensimmäinen opetusvideo oli hyvin tehty.

Tiistai 7.5.2019

Tämän päivän tarkoituksena oli tutustua mahdollisimman paljon Qt 3D Studio-työkaluun. Seuraavana päivänä osallistuin siihen keskittyvään koulutukseen ja oli hyvä keksiä sinne valmiita kysymyksiä.

Aloitin työpäivän lukemalla jonkin aikaa dokumentteja yrityksen omasta ohjelmointikielestä QML:stä, sillä olin kuullut siitä mainittavan monessa yhteydessä. Pian kuitenkin Buddy-työkaverini tuli näyttämään Qt 3D Studio-työkalua ja keskustelimme tulevasta Mayan Qt 3D Studio-työkalusta. Hänellä oli näyttää esimerkkejä jo tekemästään työstä, joten siirryimme hänen työpisteelleen. Kävimme läpi hänen jo luomaa Python-koodia, jolla voi tuoda jo jonkinlaisen version tehdystä esityksestä Mayasta 3D Studioon. Samalla hän esitteli eri työkaluja, joita oli tehnyt 3D-mallintamista varten.

Hänellä on käytössä kolmen näytön työasema, joista ensimmäinen on koodaamista varten käännetty pystyyn. Näin mahdollisimman monta riviä koodia tulee näkyviin. Toinen näyttö on suurempi, ja sitä voi käyttää kosketuskynän kanssa 3D-mallintamisessa. Kolmas näyttö on sähköposteja ja muita perustoimintoja varten.

Loppupäivän keskityin 3D Studioon ja aloitin lukemalla Qt:n verkossa julkaisemat ohjeet. Tämän jälkeen loin uuden projektin ja aloin testaamaan eri toimintoja. Huomasin ohjelman kaatuvan tietyssä pisteessä, joten kirjoitin muistion työasemalle, missä tilanteissa ohjelma kaatui. Yksi tehtävistäni on ilmoittaa bugeista Qt:n bugien seurantajärjestelmään, mihin minulla ei vielä tuolloin ollut pääsyä.

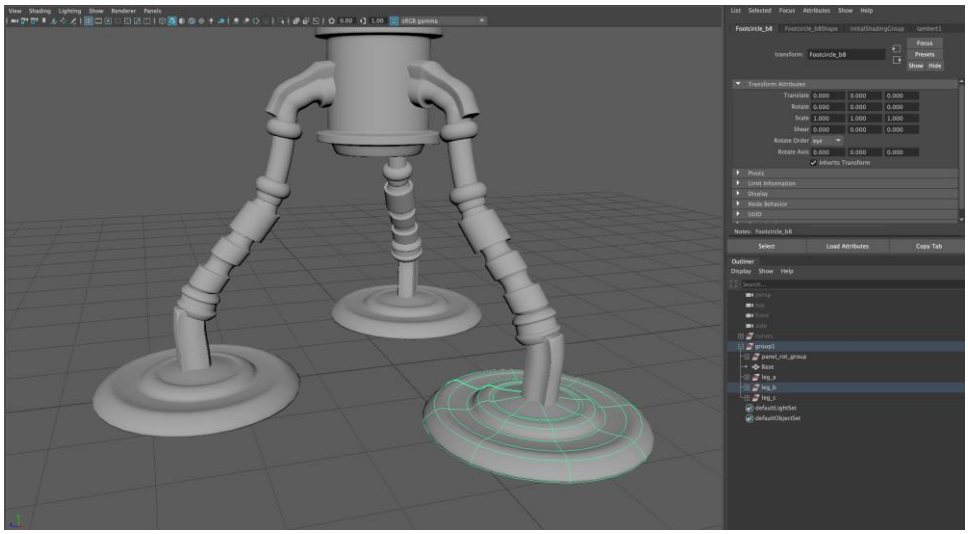
Paria tuntia ennen työpäivän päättymistä ajatukseni olivat täynnä 3D studiota ja päätin jatkaa eilen aloitettua Maya-tutoriaalia. Opin, että tässä ohjelmassa on todella käteviä ominaisuuksia 3D-mallien muokkausta varten, joihin liittyy vielä kyky muuttaa näitä määriteltyjä asetuksia jälkikäteen. Esimerkiksi Blender 3D-työkalussa tällaiset asetukset ovat yleensä kadonneet eri työvaiheiden jälkeen.

Keskiviikko 8.5.2019

Tälle päivää oli sovittu Qt 3D Studio-käyttäjäkoulutus, joka alkoi heti aamusta. Tavoitteeni päivälle oli oppia mahdollisimman paljon tästä ohjelmasta. Koulutuksen piti entinen työkaverini aikaisemmasta työpaikastani ja mukana oli myös viime viikolla aloittanut alalla selvästi kokeneempi uusi työkaveri.

Qt:n 3D Studio on yllättävän yksinkertainen työkalu, jolla voi luoda näyttäviä 3D-esityksiä. 3D Studiolla luotuja esityksiä voidaan käyttää Qt-kehitysympäristössä luoduissa käyttöliittymissä visuaalisen tiedon antamiseen tai muuten näyttävänä elementtinä. Kävimme läpi peruskäyttäjän kaikki työkalut ja testasimme eri ominaisuuksia. Ohjelman ymmärtämisessä auttoi se, että on käyttänyt vastaavanlaisia työtapoja esimerkiksi Adoben animaatiosovelluksissa. Koulutuksella sai hyvän kuvan, mitä kaikkea ohjelmalla pystyy tekemään.

Koulutuksen jälkeen palasin Maya-harjoituksen pariin. Harjoituksessa edettiin käyttäen vektorikurvityökaluja, joista minulla on vähemmän kokemusta. Näillä työkaluilla voidaan esimerkiksi luoda jonkin esineen ulkomuoto kiekon malliseksi, kun tehdään sen läpileikkauksen muotoinen kurvi, ja toisella työkalulla tämä kurvi pyöräytetään 360 astetta, niin se luo nopeasti perinteisestä mallintamisesta poikkeavia muotoja. Tämän työkalun tuloksia näkyy kuviossa 2. Harjoituksessa käytettiin myös työtapaa, jossa luodaan haluttu reitti vektorikurvilla, ja tämän jälkeen toinen vektorimuoto seuraa tätä reittiä luoden narumaisen grafiikan. Nämä molemmat työtavat vaikuttivat todella hyviltä, ja aionkin käyttää niitä tulevissa 3D-mallinnuksissa, jos mahdollista.



KUVIO 2. Vektorikurvi

Torstai 9.5.2019

Päivän tavoitteeksi asetin, että pääsisin tutustumaan jo Buddyn tekemään Maya-työkalun koodiin, sekä kehittää taitoja Maya-ohjelmalla. Aamusta jatkoin edellisten päivien aikana tekemääni 3D-harjoitusta ja sain sen valmiiksi.

Kävin ilmoittamassa esimiehelleni mitä olin suunnitellut päivälle. Tiedustelin samalla, voisinko saada toista näyttöä käyttöön, sillä 3D-työskentely yhdellä näytöllä on hieman haastavaa. Hän otti yhteyttä yrityksen IT-tukeen, ja he alkoivat toimittaa minulle näytön saman päivän aikana.

Seuraavaksi kysyin Buddylta, onko hän jakanut Maya-työkalun koodia esimerkiksi Gitissä. Git on versionhallintaohjelmisto, jolla tehtyä työtä voi versioida sekä hajauttaa koodin kehittämistä useammalle henkilölle. Tätä hän ei ollut vielä tehnyt, joten hän loi projektin Gittiin ja teki minulle oman oksan (engl. branch). Siellä voin tehdä työkalun koodilla omia testejä ja kehittämistä.

Seuraavaksi työpisteelleni tuli IT-tuen mies asentamaan toista näyttöä. Hänen lähdettyä latasin Buddyn projektin ja aloin tulkitsemaan jo tehtyä työtä. Vietin koodin parissa noin tunnin, kunnes päätin, että on järkevämpää harjoitella lisää 3D-mallintamista Mayalla.

Aloitin uuden 3D-harjoituksen, joka käsittelee syvemmin Mayan eri työkaluja. Ohjeita ja komentoja tuli harjoituksessa todella monta ja työpisteeni alkoi täyttyä muistilapuista, joissa luki pikanäppäimiä ja selostuksia eri toiminnoille.

Perjantai 10.5.2019

Tämän päivän tavoitteeksi asetin Pythonin opettelun Mayassa, sekä 3D-harjoituksien jatkamisen. Työpäivä alkoi muuten perinteisesti, paitsi aamulla oli yrityksen perinteinen perjantain kahvihetki, jossa käydään läpi toimistossa tapahtuvia asioita. Uudet työntekijät esitellään näissä tapahtumissa. Minun vuoroni tuli esittäytyä, ja kerroin omasta koulutus- ja työhistoriastani. Yksi työntekijä tiedusteli artistin taustaani sekä ihmetteli, mitä Technical Artist tekee. Kerroin, että rooliin kuuluu 3D-mallien ymmärtämisen lisäksi työympäristössä malleihin liittyvä ohjelmointi.

Kokouksen jälkeen jatkoin 3D-malliharjoitusta. Sen aikana minulla tuli sellainen olo, että mallintaminen tuntuu jo jonkin verran luontevalta Maya-ympäristössä, joten voisin aloittaa Python-ohjelmoinnin harjoittelun Mayassa.

Aivan aluksi aloin tulkitsemaan Buddy-työkaverini tekemää koodia. Kuviossa kaksi näkyy työkalun tekemä uip–tiedoston sisältö. Uip-tiedosto sisältää 3D Studio esityksien rungon, joka koostuu samanlaisista tietohierarkioista kuin yleiset html- tai xml-tiedostot. Kuviossa kolme näkyy pätkä Buddyn tekemän koodin luomaa uip-tiedostoa ja kuviossa neljä koodia, joka luo uip-tiedoston.

```
        </Graph>
        <Logic >
            <State name="Master Slide" component="#Scene" >
                <Add ref="#Layer" />
                <Add ref="#__Container" name="__Container" />
                <Add ref="#materials/lambert1" name="materials/lambert1" />
                <State id="Scene-Slide1" name="Slide1" >
                    <Add ref="#pSphere1" importfile="../models/sceneExport/sceneExport.import" />
                    <Add ref="#lambert1" name="lambert1" referencedmaterial="#materials/lambert1"
                        sourcepath="../materials/lambert1.materialdef" />
                    <Add ref="#pCube1" importfile="../models/sceneExport/sceneExport.import" />
                    <Add ref="#lambert1_001" name="lambert1" referencedmaterial="#materials/lambert1"
                        sourcepath="../materials/lambert1.materialdef" />
                    <Add ref="#pCone1" importfile="../models/sceneExport/sceneExport.import" />
                    <Add ref="#lambert1_002" name="lambert1" referencedmaterial="#materials/lambert1"
                        sourcepath="../materials/lambert1.materialdef" />
                    <Add ref="#directionalLight1" importfile="../models/sceneExport/sceneExport.import" />
                    <Add ref="#pointLight1" importfile="../models/sceneExport/sceneExport.import" />
                    <Add ref="#areaLight1" importfile="../models/sceneExport/sceneExport.import" />
                    <Add ref="#camera1" importfile="../models/sceneExport/sceneExport.import"
                        position="21.935 1.562 -1.761" />
                </State>
            </State>
        </Logic>
    </Project>
```

KUVIO 3. Alustava uip–tiedosto

```
#-------------------------------------------------------------------------------##
#            Qt 3D Studio .UIP file generator from maya scene                   ##
#-------------------------------------------------------------------------------##

##issues

## Scaling is not correct
## Sometimes gives Error: RuntimeError: file <maya console> line 85: Non camera object in the list. #

from xml.dom.minidom import Document
import maya.cmds as cmds
import pymel.core as pm
import math
import os

##------------------ UIP.FILE --------------------#

doc = Document()

##------------------ GLOBALS --------------------#

root_node = doc
pro_node = doc
settings_node = doc
customcolor_node = doc
scene_node = doc
logic_node = doc
scene_node = doc
layer_node = doc
resolutionX = 1920
resolutionY = 1080


##-------------------------- CREATE MATERIAL FUNCTIONS ------------------------##
## currently only creates one default material with model name


def create_default_material(name,modelparent):
    # CREATE NODE
    material_node = doc.createElement("Material")
    material_node.setAttribute("id", name)
    material_node.setAttribute("name", name)
    # CREATE REF
    add_node = doc.createElement("Add")
    state_node.appendChild(add_node)
    add_node.setAttribute("ref", "#"+name)
    # ASSIGN TO PARENT
    modelparent.appendChild(material_node)

##---------------------------- CALCULATE TRANSFORMS -------------------------##

def calculate_transforms(name):
    transforms = []
    # GET POSITION
    posX = cmds.getAttr(name+'.translateX')
    posY = cmds.getAttr(name+'.translateY')
    posZ = cmds.getAttr(name+'.translateZ')
    # GET ROTATION
    rotX = cmds.getAttr(name+'.rotateX')
    rotY = cmds.getAttr(name+'.rotateY')
    rotZ = cmds.getAttr(name+'.rotateZ')
    # GET SCALE
    sclX = cmds.getAttr(name+'.scaleX')
    sclY = cmds.getAttr(name+'.scaleY')
    sclZ = cmds.getAttr(name+'.scaleZ')
    transforms.extend([posX,posY,posZ,rotX,rotY,rotZ,sclX,sclY,sclZ])
    return transforms


##-------------------------- CREATE OBJECT FUNCTIONS ------------------------##


##-------------- CREATE CAMERA ----------------#
```

KUVIO 4. Alustava työkalukoodi

En ollut varma, että onkohan tämä kaikista tehokkain tapa oppia aiheesta ja päädyin etsimään verkkovideokurssia samalta sivustolta, josta olen tehnyt 3D-mallinnusharjoituksia. Sivustolla oli yli

sata Maya-kurssia, joista löysin ainakin viisi hyödylliseltä vaikuttavaa verkkokurssia. Osassa kursseista käytiin läpi myös Qt:n hyödyntämistä Mayalla. Mayan käyttöliittymä on tehty Qt:lla.

Valitsin kurssin, joka vaikutti käyvän läpi aloittelijatason Python-ohjelmointia Mayalla. Tein kurssia noin puolet päivästä, ja sain sen lähes läpäistyä. Mieleeni tuli heti useita eri tapoja, kuinka voisin hyödyntää itseohjelmoituja työkaluja 3D-mallintamisessa.

Keskustelin työpäivän aikana myös Buddyn kanssa tämän kesän tavoitteista. Myös esimiehemme liittyi keskusteluun, ja päätimme, että tulevina viikkoina pitäisimme kokouksen projektin etenemisestä ja dokumentaation oikein tekemisestä.

Viikkoanalyysi

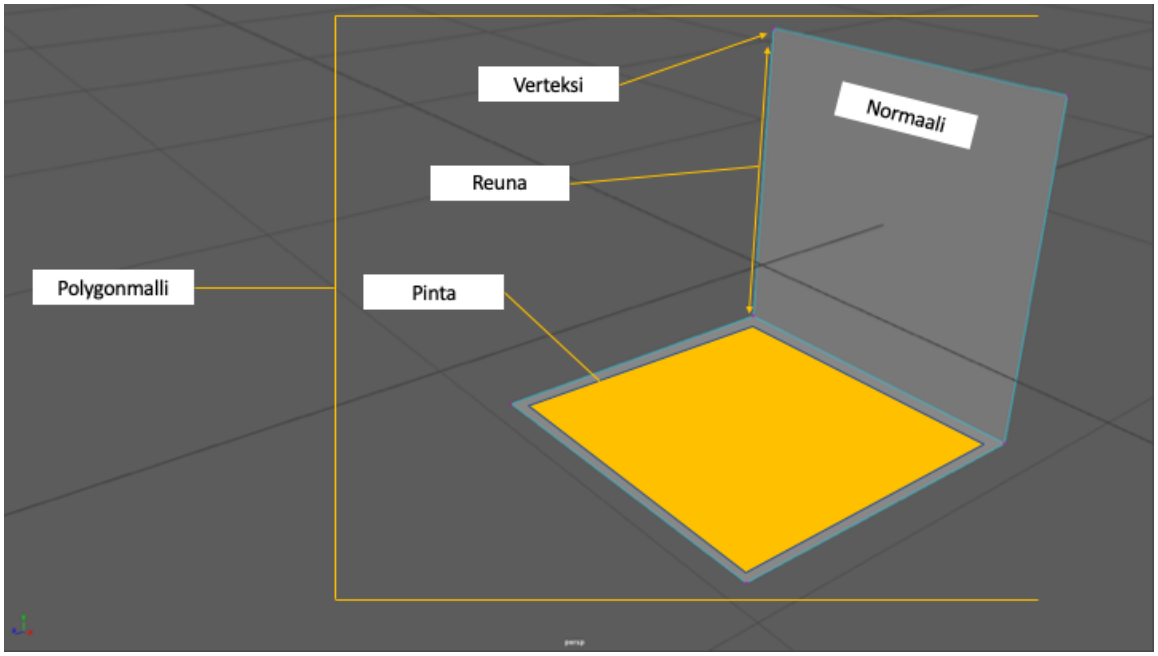
Tämä viikko meni mielestäni todella mallikkaasti ja tuntui nopealta. Pystyin hyvin keskittymään tärkeisiin aiheisiin joka päivä. Opin yrityksestä, tutustuin moneen työkaveriin, tutustuin tulevissa töissä käytettäviin 3D mallintamistyökaluihin, harjoittelin Python-koodin käyttöä Maya-ohjelmassa ja opin paljon yrityksen omasta 3D Studio-ohjelmasta. Paljon pintaraapaisuja monesta aiheesta, mutta olen tyytyväinen siihen, kuinka paljon opin tällä viikolla.

Selvittämistyötä jouduin tekemään eniten 3D- sekä Python-harjoituksien aikana. Opetusvideoista oli todella paljon hyötyä, ja niiden lisäksi otin aiheista lisäselkoa. Tavallinen 3D-mallinnus on minulle tuttua, joten Mayan käyttämisen aloittaminen ei ollut vaikeaa.

Kaikki 3D-mallinnusohjelmat alkavat kolmiulotteisilla peruskappaleilla, jotka ovat yleensä yksinkertaisia muotoja tai geometriaa. Alkupiste tai origin on kohta, jolla arvo on nolla kolmelle eri ulottuvuudelle. Nämä ulottuvuudet ovat x,y ja z. Työstettävää objektia voidaan muokata monipuolisemmaksi 3D-mallinnusohjelman eri työkaluilla. (Crease 2019, viitattu 2.6.2019.)

Näiden ulottuvuuksien käyttö vaihtelee. Kaikissa käyttämissäni ohjelmissa x on horisontaalinen sivulta sivulle. Y- ja z-ulottuvuuden kanssa on ollut vaihtelua. Blender käyttää z-ulottuvuutta vertikaaliseen ylös-alas-tietoon ja y-ulottuvuutta syvyystietoon. Mayassa nämä menevät päinvastoin.

Kulmapiste (verteksi) on sijainti paikkaulottuvuudessa. Yleensä se yhdistää kahta tai useampaa reunaa. Reuna (engl. edge) on kahden kulmapisteen välinen viiva. Kun reunoja, jotka jakavat samoja pisteitä on vähintään kolme, syntyy pinta (engl. face). Pinnoista muodostuu polygonimalli (mesh) ja jokaisella pinnalla on normaali (engl. normal), joka kertoo, onko pinta sisä- vai ulkopuoli. Pelkästään ulkopuoli tulee näkyviin, kun tuotos renderöidään. Kuviossa viisi näitä ilmennetään visuaalisesti.



KUVIO 5. Yksinkertainen 3D-malli

Mayalla voi luoda kolmen tapaisia malleja. Polygoneita, NURBS (Non-Uniform Rational B-Splines) sekä subdivision surfaceja. Kullakin näillä on omat vahvuutensa:

- **Polygon-mallinnuksella** lisätään ja muokataan yksinkertaisia pintoja.
- **NURBS** mahdollistaa helpon luomisen sileille ja kurvikkaille pinnoille.
- **Subdivision surface** mahdollistaa polygon-tyyppisen mallinnuksen NURBS lopputuloksella. Objektin muokkaaminen on yksinkertaista ja pinnoista tulee sileämpiä kuin tavallisella polygon-mallintamisella.

(Autodesk Maya 2019, viitattu 2.6.2019.)

Mielestäni tällä viikolla ei ollut ongelmia minkään asian suhteen, mutta tavallisia haasteita oli varsinkin jo tehdyn Python-koodin kanssa. Toisen tekemän koodin tulkitseminen vie aikaa ja tässä

tapauksessa Maya-ympäristö ja siihen liittyvä ohjelmointi on uutta. Olen ollut ennenkin tällaisessa tilanteessa, ja uskon, että osaan tulevina viikkoina perehtyä oikeisiin asioihin, jotta itsenäinen työskentely olisi heinäkuussa mahdollista.

3.2 Toinen viikko: Työkalun ohjelmoinnin aloittaminen

Maanantai 13.5.2019

Päivän tavoitteeksi asetin oppia jotain uutta Maya-työkalun kehitystä varten. Viime viikolla ehdin hieman tutustumaan Maya Command moduuliin. Se sisältää Pythonille samoja komentoja, joita Mayan oma ohjelmointikieli Maya Embedded Language (lyh. MEL) käyttää. Sen avulla voidaan tehdä jokainen toiminto, jonka käyttäjä voi tehdä työkalun ikkunan kautta ja paljon enemmän.

Moduulilla tarkoitetaan ohjelmoinnissa itsenäistä osiota, jolla voi vastaanottaa ja palauttaa tietoa sekä käsitellä sitä omissa funktioissaan. Toinen ohjelmoinnissa käytetty koodiin tuotava lisäosa on kirjasto (engl. Library). Kirjaston piirre on se, että ne ovat toimintakohtaisia ja sisältävät hyödyllisiä funktioita, joita käyttäjä voi kutsua. Kolmas yleinen ohjelmoinnissa käytettävä termi on ohjelmointirajapinta (engl. API = Application Programming Interface). Se tarkoittaa ohjelman kykyä vastaanottaa käskyjä ja vaihtaa tietoja. Näin esimerkiksi kaksi ohjelmaa pystyy vaihtamaan tietoa keskenään.

Seuraavaksi päätin edetä seuraavaan moduuliin, jota jo tehdyssä työssä on käytetty. Se on nimeltään xml.dom.minidom. Sillä voidaan luoda xml-tiedostoja käyttäen solmuja (engl. Node). Solmu on tietotekniikassa tiedon palanen, joka voi sisältää ominaisuuksia, eri tyyppisiä tietoja ja toisia solmuja. Ne auttavat luomaan isompia tietokokonaisuuksia. Ammattikorkeakoulututkinnossani xml-tiedostojen käsittely oli ollut vähäistä, joten ensimmäiseksi luin pari tuntia tästä tiedostomuodosta ja tein muistiinpanoja. Samalla törmäsin hyviin teksteihin solmuista ja jatkoin niihin tutustumista.

Parin tunnin jälkeen sain luotua ensimmäisiä xml- ja uip-tiedostoja. Uip-tiedostoja käytetään Qt 3D Studiossa kuvaamaan esitystä, ja toimivan sellaisen luominen on yksi tämän kesän tavoitteista. Ongelmia tuotti aluksi tiedoston kirjoittaminen, sillä Mac-tietokoneessa kaikki ei toimi samalla tavalla kuin Windows-laitteessa. Stack Overflow-sivuston avulla löysin keskustelun, jossa käytiin läpi samantyyppistä ongelmaa, ja sain siellä olevilla vinkeillä korjattua oman koodini.

Olin todella tyytyväinen päivän etenemiseen, sillä ymmärsin mallikoodia nyt paljon paremmin. Päätin loppupäivän jatkaa mallinnusharjoituksia. Huomasin myös, että 3D-mallinnus sujui nyt todella sulavasti.

Tiistai 14.5.2019

Tämän päivän tavoitteena oli saattaa 3D-mallinnusharjoitus loppuun ja edetä Python-koodausharjoituksien kanssa. Aloitin aamupäivän jatkamalla 3D-mallin tekoa, ja sain sen ensimmäisen tunnin jälkeen valmiiksi. Mallintamista varten verkkokursseja olisi vielä useampia, mutta mielestäni on hyödyllisempää opetella jotain uutta ja erilaista Maya-työkalulla.

Yksi seuraavista kehitysaskeleista valmiin mallin kanssa voi olla materiaalien työstäminen ja teksturointi. Pelkän mallintamisen jälkeen mallilla on pelkkä muoto, mutta ei mitään ominaisuuksia. Materiaaleilla annetaan polygonimalleille pintaominaisuudet, joita voi olla värit, tekstuurit ja muut kehittyneemmät ominaisuudet. Etsin sopivia kursseja, jotka harjoittaisivat näissä, tallensin ne muistiin, ja päätin jatkaa Python koodausta.

Löysin uuden koodaukseen keskittyvän verkkokurssin ja suoritin sitä lopun päivää. Alussa toistettiin paljon samaa, mitä olen oppinut aiemmilla kursseilla, mutta opin myös uutta. Viimeisen tunnin aikana uutta tietoa tuli todella paljon, ja jouduin toistamaan kohtia useasti. Alkoi tuntua siltä, että tarvitsisin jo jotain varsinaisia työtehtäviä pelkän harjoittelun lisäksi.

Keskiviikko 15.5.2019

Tämän päivän tavoitteena oli oppia mahdollisimman paljon Python-koodauksesta. Edellisenä päivänä en pystynyt keskittymään hyvin viimeisen työtunnin aikana. Tuolloin verkkokurssissa käytiin läpi käyttöliittymän ohjelmointia. Aloitin tuon osion uudestaan alusta virkeällä mielellä, ja sain hyviä muistiinpanoja ja oivalluksia.

Jaksotin harjoittelun niin, että tein kaksi tuntia koodaukseen liittyviä harjoituksia, ja seuraavat kaksi tuntia 3D-mallin materiaaleihin liittyviä harjoituksia. Puolelta päivin pidimme ryhmäkokouksen, jossa kaikki taas kertoivat, kuinka työskentely oli edennyt. Seuraavana päivänä Buddyn oli määrä palata töihin, joten oli mahdollista saada häneltä tuolloin ohjausta työtehtävien aloittamiseen.

Torstai 16.5.2019

Buddy palasi toimistolle, ja tavoitteenani oli päästä alkuun työtehtävien kanssa. Tapasin hänet heti aamusta, ja sovimme kokouksen lounaan jälkeen. Kävimme tuolloin läpi Maya-projektin tavoitteita, sekä työtehtävien järjestystä. Jatkoin aamulla edellispäivän Python-harjoituksia, kunnes pidimme kokouksen.

Pidimme kokouksen kahdestaan, ja kuuntelin ensin Buddyn pitämän esityksen siitä, millainen kuva hänellä on lopullisesta työkalusta. Minulla heräsi useita kysymyksiä, ja mielikuvani lopullisesta tuotteesta selkeni. Buddy piirsi ja hahmotti ideaansa taululle. Otin tarkasti muistiinpanoja työkalun toiminnallisuudesta sekä käyttöliittymäideoista. Kävimme läpi kaikkea työkaluun liittyvää ja ideoimme sitä yhdessä. Kokous meni hyvin luontevasti ja oli kiinnostava.

Sovimme, että aloitan tutkimustyön Mayan virallisista lisäosista, ja jatkan tämän jälkeen työkalun kehittämistä. Tärkeää olisi saada työkalusta ensimmäinen versio, joka sisältäisi käyttöliittymän. Sillä pitäisi voida myös määritellä 3D Studion projektin juurikansio, minne tulevat tiedostot luodaan.

Perjantai 17.5.2019

Tämän päivän tavoitteeksi asetin Maya-työkalun kehittämisen aloittamisen. Edellisen päivän kokouksessa ensimmäiseksi tehtäväksi asetettiin annettavan lähdepolun tarkistaminen Qt 3D Studion projektikansioksi. Ajattelin myös, että hyvin yksinkertaisen käyttöliittymän tekeminen olisi hyvä idea.

Loin kaksi testiprojektia Qt 3D Studiolla ja tarkistin, millaisia projektikansioita ja tiedostoja ohjelma tekee. Kaikissa projektijuurikansioissa oli samat kahdeksan alikansiota sekä uia-tiedosto. Ajattelin, että tarkistuksena riittäisi funktio, joka käy läpi asetetun polun ja tarkistaa, löytääkö se nuo alikansiot sekä uia-tiedoston. Nämä tiedostot sisältävät 3D Studio projektien asetuksia.

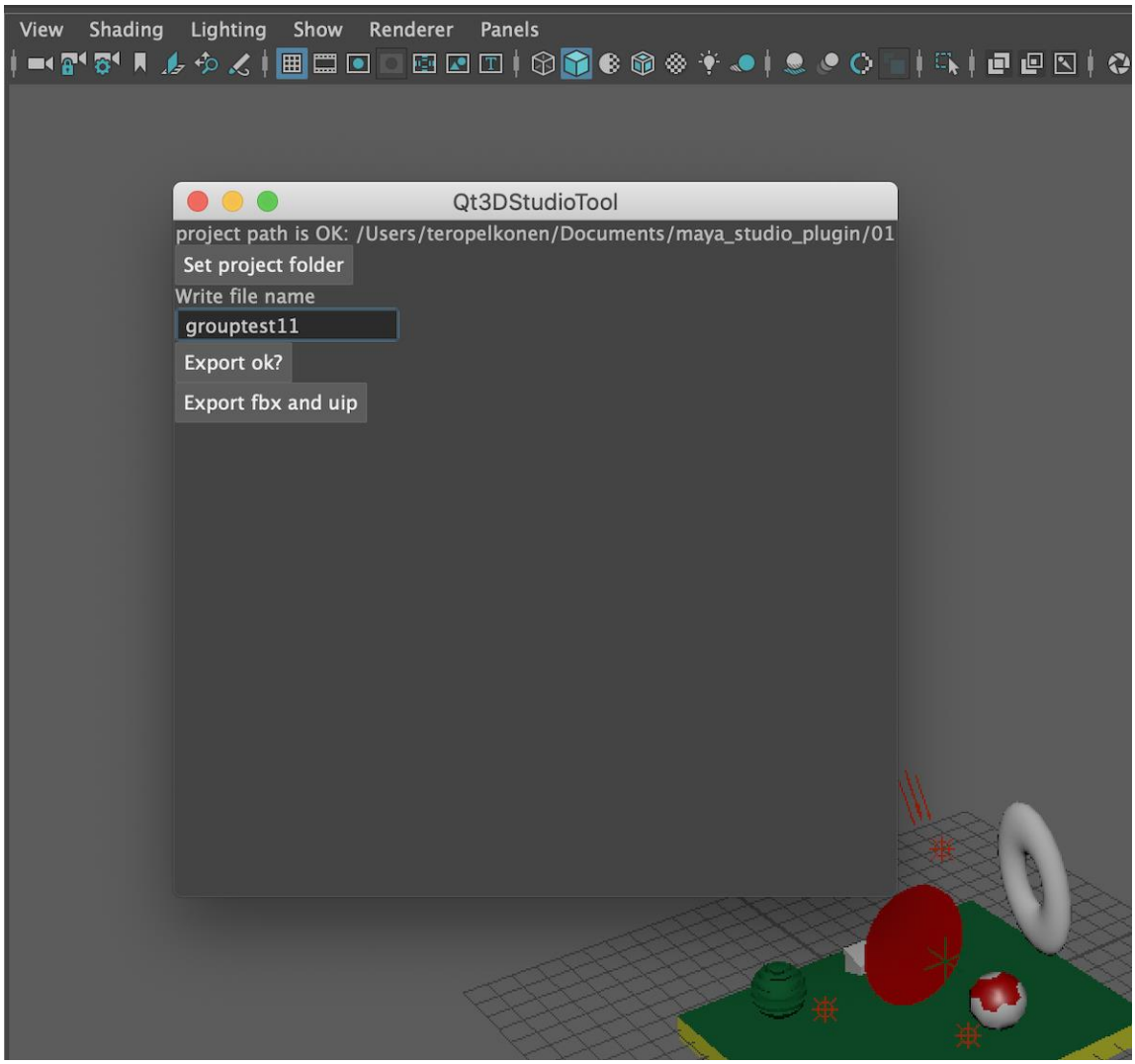
Polun asettamiselle oli Buddy jo tehnyt osittain valmista koodia, jossa määritetään polku käyttäen Maya Cmds ja Os moduulia. Jälkimmäinen on minulle uusi tuttavuus, ja hakukone-etsinnöillä löysin ohjeita siitä. Sillä voidaan tehdä kansioihin liittyviä toimia. Minulle tärkeä toiminto oli os.path.exists

(polku), jonka avulla pystyin tarkistamaan, olivatko alikansiot olemassa annetussa polussa. Tiedoston tarkistamista varten loin toisen funktion, jossa kävin läpi kaikki polun alikansiot ja tiedostot. Jos se löysi tiedoston uia päätteellä, niin se palautti True-totuusarvon.

Testasin tarkistusta, ja alustavasti se toimi oikein. Voi toki olla, että siihen luodaan vielä lisää ominaisuuksia, mutta ensimmäiseksi askeleeksi itselle olin siihen tyytyväinen. Päivälle oli vielä jäljellä fbx-tiedoston vienti koodilla oikeaan paikkaan, sekä käyttöliittymän kehittäminen. Fbx-tiedosto on yleinen tiedostomuoto 3D-mallien siirtämiseen. Päänvaivaa aiheutti erityisesti käyttöliittymän ohjelmointi Maya Commands moduulilla, sillä sen logiikka ei avautunut minulle heti.

Viikkoanalyysi

Viikko oli hyvin Python-ohjelmointikeskeinen, vaikka 3D-mallinnusta tuli myös tehtyä. Kuva työtehtävästä selkeni huomattavasti torstain kokouksessa, ja sain konkreettisen rungon, kuinka edetä. Olen tyytyväinen päivittäiseen työskentelyyni, sillä sain vietyä asioita eteenpäin tai opittua jotain uutta. Suurimmat haasteet koin Maya Commands moduulin kanssa tehdessäni alustavaa käyttöliittymää, ja uskon että sen kanssa joudun vielä haasteisiin. Olen kuitenkin törmännyt vastaavanlaisiin haasteisiin aikaisemmin, ja tiedän, että asian ratkeaminen voi vaatia vain yhden oivalluksen, jonka jälkeen työskentely sujuu ongelmitta. Kuviossa kuusi näkyy yksi versio ensimmäisistä käyttöliittymistä.



KUVIO 6. Yksinkertainen käyttöliittymä

Opintoni ovat painottuneet verkkosovelluksien kehittämiseen, ja sieltä pystyin hyödyntämään hyvin solmuihin HTML-tiedostorakenteen osaamista. Tämä nopeutti niiden ymmärtämistä.

Olen opiskellut useampia ohjelmointikieliä Python mukaan lukien, ja silti sen käyttäminen tuntui aluksi erikoiselta, koska siinä ei käytetä sulkuja. Toinen erikoisuus oli ohjelmoinnissa käytettävä Self-määritelmä, joka annetaan funktioihin. Näihin tietenkin tottuu nopeasti.

Python on yksinkertainen, ja siinä on helposti opittava syntaksi, joka on myös helppo lukuista. Se on korkea tasoinen ohjelmointikieli, jota on hyvä käyttää skriptaukseen sekä liimakieleksi jo

olemassa olevien komponenttien väliin. Se myös tukee moduulien käyttöä. Tämä rohkaisee tekemään koodia, joka sisältää moduulipaketteja ja koodin uudelleenkäyttöä. (Python 2019, viitattu 2.6.2019.)

Python-kielellä tehty ohjelma voi olla kolmesta viiteen kertaa pienempi kuin Java-kielellä kirjoitettu, ja se saadaan kirjoitettua nopeammin. Python-ohjelmoijan ei tarvitse käyttää aikaa muuttujien tyyppijulistuksiin. Python-koodi saattaa toimia hitaammin kuin Java. (Python 2019, viitattu 2.6.2019.)

Pythonin olio-ohjelmointi ominaisuus on lähes sama kuin JavaScriptissä. Samoin kuin JavaScriptissä, Python tukee ohjelmointityyliä, joka käyttää yksinkertaisia funktioita ja muuttujia ilman kytkemistä luokkamäärityksiin. Toisin kuin JavaScriptillä Pythonilla voidaan luoda isompia ohjelmia ja valjastaa käyttöön oikea olio-ohjelmoinnin tyyli, missä luokilla ja perimisellä on iso rooli. (Python 2019, viitattu 2.6.2019.)

Alkuviikon tutkimustyö xml-tiedostomuodon, solmujen sekä xml.dom.minidom moduulin käyttöön oli hyödyllisin asia, jonka opin tällä viikolla. Opin tunnistamaan tiedoston rakennetta ja mahdollisia ominaisuuksia. Näiden avulla pystyn tulkitsemaan jo tehtyä työtä paremmin. Myös perjantaina itse koodattu tarkistusfunktiosarja toi onnistumisen tunnetta.

Xml.dom.minidom on minimaalinen toteutus DOM (engl. Document Object Model) käyttöliittymästä, joka sisältää muiden yleisten ohjelmointikielien tapaisen API:n. Xml.dom.minidom-dokumentaatio oli minulle hieman hankalalukuista tekstiä, mutta käyttäessä moduulia sen kieli on niin itseään selittävää, ettei ongelmia ole vielä syntynyt. (Python 2019, viitattu 2.6.2019.)

3.3 Kolmas viikko: Työkalu saa lisää ominaisuuksia ja Blender-harjoitukset

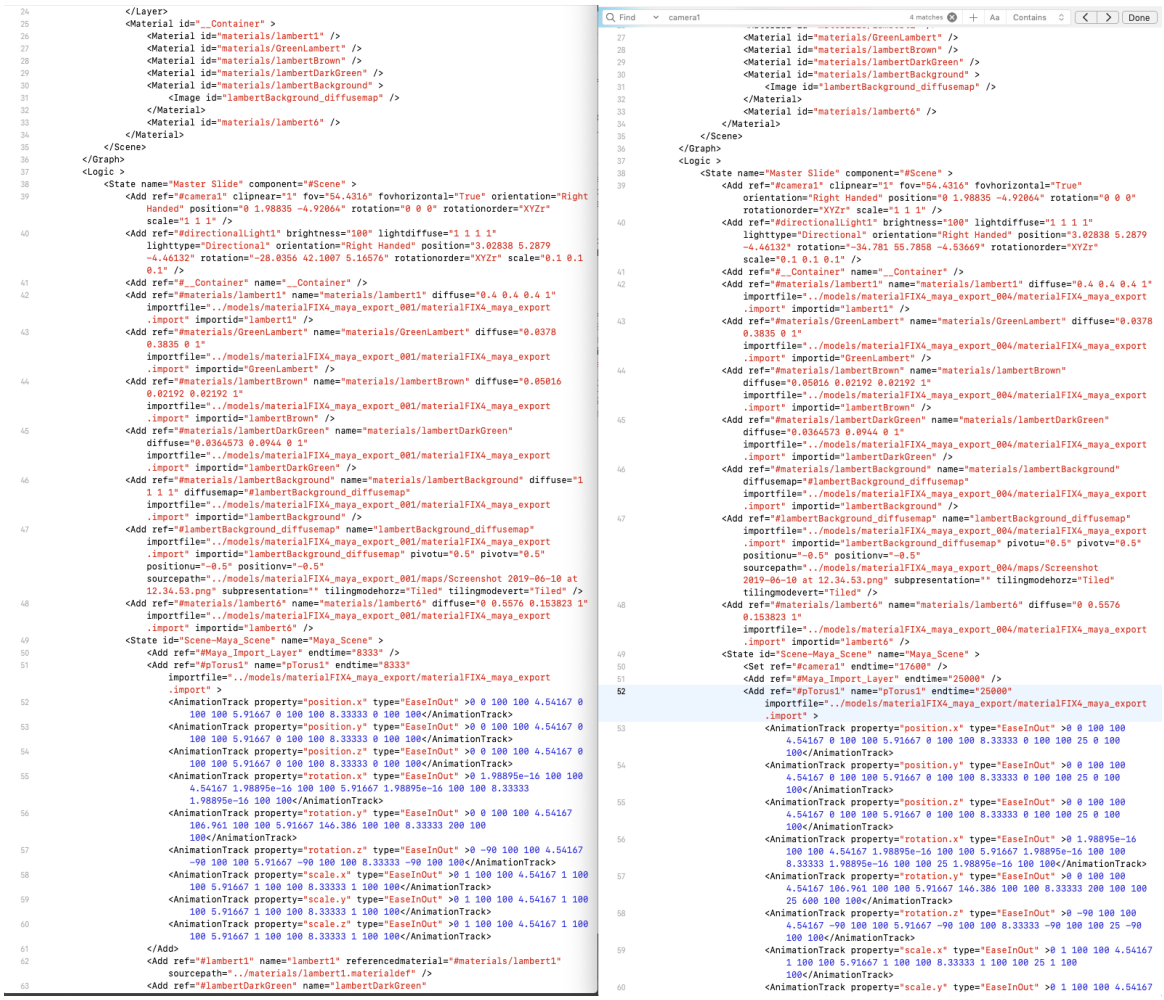
Maanantai 20.5.2019

Päivän tavoitteeksi asetin edistymisen Maya-työkalun kehittämisen kanssa. Ensimmäiseksi aamulla luin viime viikolla tehdyt koodinpätkät ja tein parannuksia niihin. Nyt esimerkiksi käyttöliittymäikkuna toimii paremmin. Seuraavaksi pitäisi selvittää, kuinka Maya-esitys (engl. scene) saadaan vietyä ulos fbx-tiedostona käyttäen Pythonia.

Lyhyen verkkohaun jälkeen olin jo löytänyt kaksi tapaa tehdä tiedoston vieminen ohjelmasta. Loin uuden Python-koodin ja testasin ensimmäistä vaihtoehtoa. Sain pelkästään virheilmoituksia, ja lähdin niiden avulla etsimään lisäohjeita eri verkkofoorumeilta. Ilmeni, että muutkin Maya-ohjelmoijat olivat törmänneet vastaavanlaiseen. Heillä oli toiminut eri komento, jota sitten kokeilin onnistuneesti. Lisäsin onnistuneen tiedoston viemisen viime viikolla tehtyyn työhön, jonka myötä skriptillä voi valita juurikansion, tarkistaa sen ja luoda sinne fbx-tiedoston, joka sisältää esityksen 3D-mallit.

Seuraavaksi aloitin Buddyn tekemän uip-tiedoston kääntäjän tulkitsemisen. Kääntäjä tekee esityksestä uip-tiedoston, joka on periaatteessa xml-tiedosto eri tiedostopäätteellä. Hän ja esimieheni eivät olleet paikalla, joten etsin toisen työkaverin tiimistäni, jolta pystyin tarvittaessa tarkistamaan, olenko ymmärtänyt uip-tiedostoa oikein.

Aloitin lukemalla koodin pääkohdasta, missä järjestyksessä funktiot tapahtuvat, mikä milläkin parametreilla. Pidin näytillä 3D Studion ja työkalun tekemää xml-tiedostoja sekä koodia. Vertasin xml-tiedostoja keskenään, ja otin samalla muistiinpanoja eri solmujen merkityksestä sekä niiden hierarkiasta. Kuviossa seitsemän näkyy esimerkki verrattavista tiedostoista.



KUVIO 7. Tiedostojen vertailu

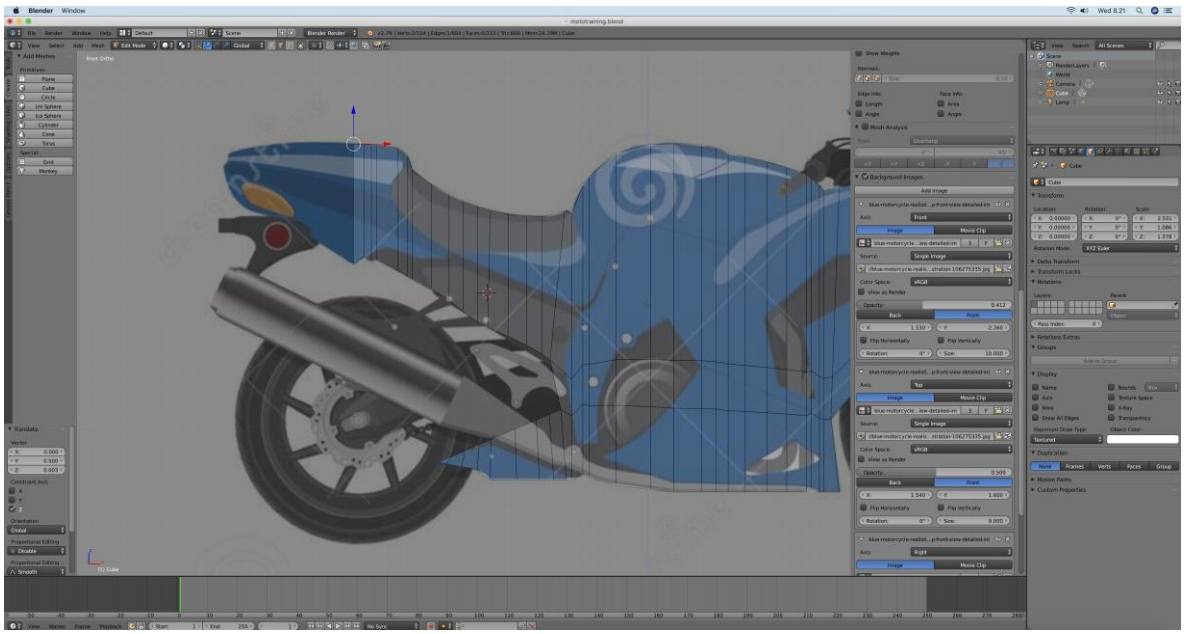
Tiistai 21.5.2019

Eilen edistystä tapahtui paljon Python-koodauksessa, joten päätin jatkaa samaa tänään. Buddyn tekemässä koodissa oli maininta kameran väärästä FOV:ista eli Field of Visionista. Sen tarkoitus on kertoa, kuinka laajalla otannalla kamera näyttää objektit linssin läpi. Sama periaate kuin oikeissa kameroissa. Päättelin, että ongelma voi johtua siitä, kuinka sama asia lasketaan ja nimetään Mayassa sekä Qt 3D Studiossa. Mayassa tätä kutsutaan Field of Visioniksi ja 3D Studiossa Angle of viewksi.

Aloitin vian selvittämisen etsimällä tietoa Mayan kameroiden periaatteesta. Tämän jälkeen tutkin koodia, jossa solmu kameralle luodaan. Tulostin konsoliin tulevia arvoja ja muuttujia, ja luin niistä lisätietoja. Konsoli on ohjelman ikkuna, josta nähdään ohjelman toimintaan liittyviä viestejä. Sinne voidaan myös tulostaa omia viestejä, jotka ovat olennaisia ohjelmoinnissa ja vikojen korjaamisessa. Huomasin, että funktio ei tulostanut oikeaa arvoa annetuilla parametreillä. Tilanne ratkesi vaihtamalla parametri Vertical field of vision parametriin Horizontal view of vision.

Tämän jälkeen keskityin materiaaleihin. Tämä osio koodista on vielä todella alkutekijöissä, joten en uskonut, että saisin sen kanssa aikaiseksi hirveästi ilman ohjausta. Lopulta vietin lähes koko päivän koodaten tätä osiota ja tutkien, miten materiaalit saisi siirrettyä.

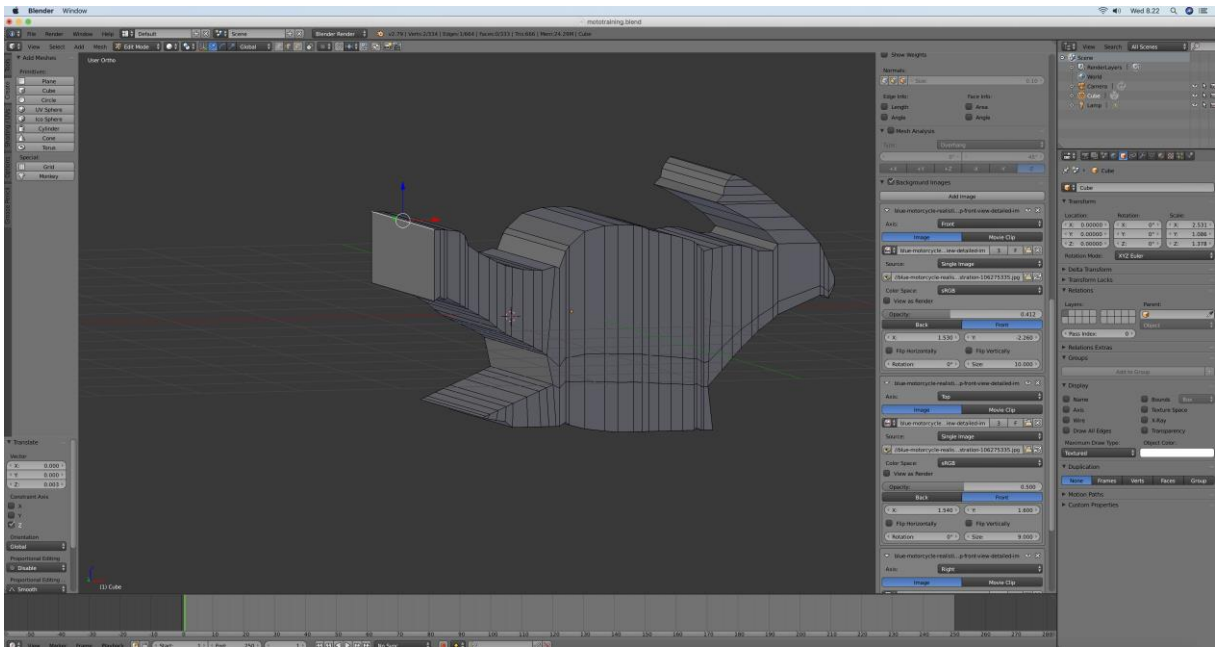
Lopulta huomasin, että nyt pitää tehdä jotain muuta, sillä keskittyminen ohjelmointiin ei enää onnistunut. Päätin testata pitkästä aikaa Blender 3D-mallinnustaitojani. Etsin verkosta piirustukset yksinkertaisesta moottoripyörästä ja aloin mallintamaan. Kuvioissa kahdeksan näkyy piirustus sekä alustavan mallin luominen. Mieleni virkistyi ja Blender-työkalujen käyttö palasi muistiin nopeasti. Mallintamisen kautta saan esimerkki 3D-malleja ja voin huomata asioita luotavan työkalun käyttäjän näkökulmasta, vaikkei mallinnusohjelma ole Maya.



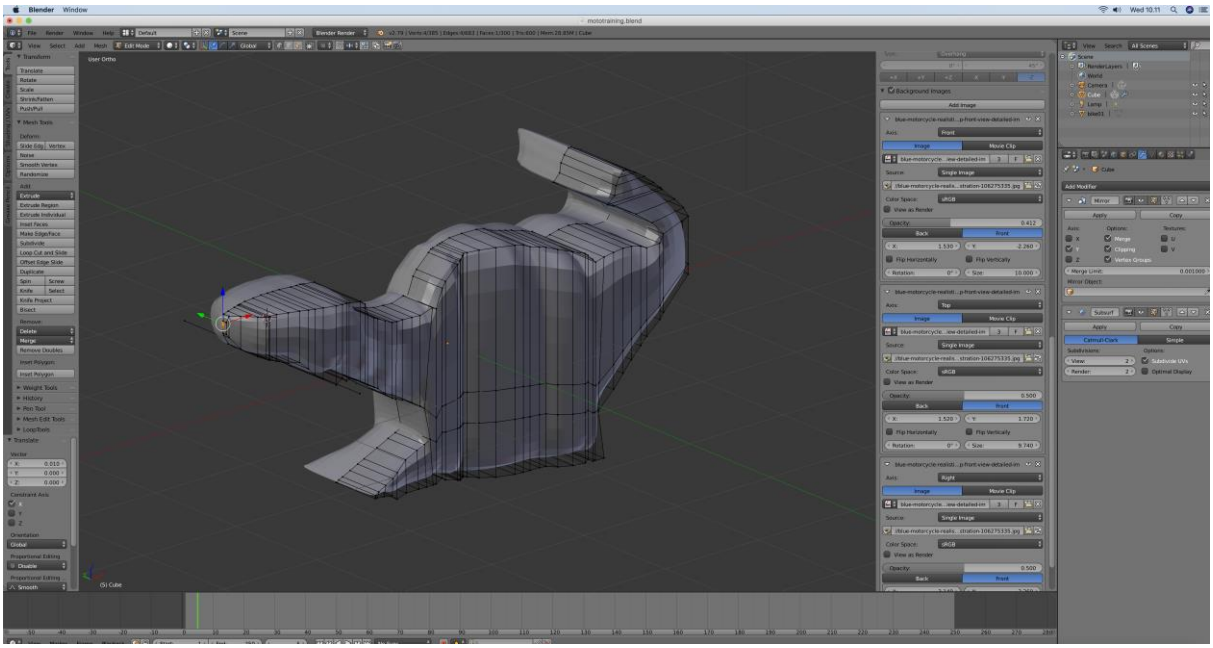
KUVIO 8. Blender-harjoittelun aloitus

Keskiviikko 22.5.2019

Tämän päivän tarkoituksena olisi jatkaa Blender-harjoittelua, sekä kehittää 3D Studio työkalun Python-koodia. Aloitin Blenderillä. Ensimmäisenä päätin kokeilla Mirror- sekä Subdivision-modifikaattoreita. Nämä ovat yleisimmät työkalut, joita olen käyttänyt mallintamisessa. Mirrorilla voidaan heijastaa malli halutun akselin toiselle puolelle, jolloin puoliksi katkaistu malli saadaan valmiiksi työstämällä vain toista puolta. Subdivision lisää malliin muotoja laskutoimituksien avulla. Näin voidaan muokata yksinkertaisempaa 3D-mallia helposti, ja saada monipuolisempi lopputulos. Kuvio yhdeksän ja 10 näyttävät, kuinka nämä työkalut nopeuttavat työtä.



KUVIO 9. Malli, jossa mirror-modifikaattori



KUVIO 10. Malli, jossa mirror- sekä subdivision-modifikaattorit

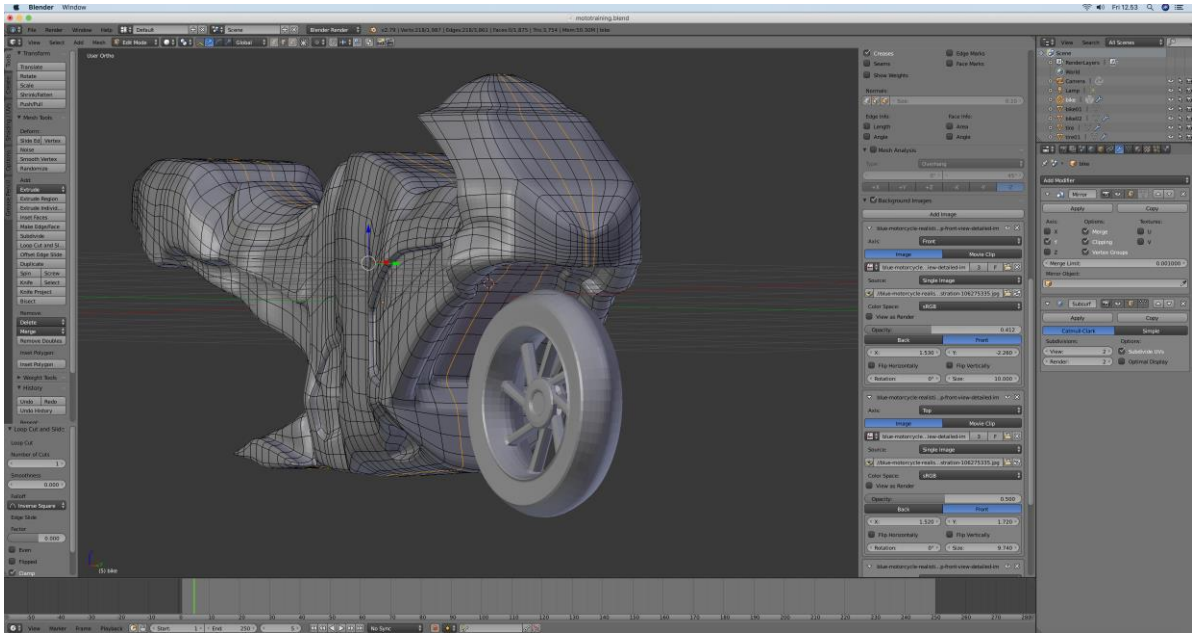
Noin tunnin päästä päätin jatkaa koodaamista, ja halusin esittää kysymyksiä asiantuntijalle. Hain työpisteelleni 3D-tiimistä tutun työkaverin ja esitin hänelle kysymyksiä liittyen työkaluun ja hankalimpaan haasteeseen. Kyse oli siitä, että työkalu ei toimi, ellei käyttäjä tuo 3D Studio projektiin Mayasta vietyä fbx-tiedostoa. Tämän pitäisi olla automaattista, ja vaatisi 3D Studio tiimin osallistumista. Käymme asiasta hyvän keskustelun ja sovimme, että asia nostetaan seuraavassa tiimikeskustelussa esille. Ilmeni myös, että 3D Studio hyväksyy vain tietyillä asetuksilla viedyt fbx-tiedostot. Selvitin loppupäivän, kuinka nuo asetuksien määritykset voidaan lisätä työkalun koodissa.

Torstai 23.5.2019

Tämän päivän tavoitteeksi asetin koodin kehittämisen. Luon uuden 3D Studio projektin, jonne lisään Mayassa tehdyn 3D-mallin, joka sisältää kolmea eri materiaalia. Tämän jälkeen aioin avata tämän projektin uip-tiedoston ja tuoda vierelle oman työkalun tekemän uip-tiedoston ja aloittaa vertailun.

Löysin nopeasti eroja eri objektien hierarkioissa ja ne piti korjata. Keskityin ensin mesh-objektien materiaaleihin. Löysin tavan listata jokaisen objektin omat materiaalit yksittäisille listoille, mikä on hyvä edistysaskel.

Loppupäivästä jatkoin 3D-harjoittelua, sillä halusin päästä lisäämään pian materiaaleja itsetehtyyn malliin ja ehkä löytää tätä kautta koodiin jotain uutta korjattavaa. Kuviossa 11 näkyy Blenderissä tehtävän mallin eteneminen.



KUVIO 11. Malli saa lisää muotoja

Perjantai 24.5.2019

Torstaina löydetty tapa eritellä materiaaleja oli tärkeä toiminto, joten otin tämän päivän tavoitteeksi käydä läpi kaikki esityksessä olevat objektit ja nähdä, että ne tulevat oikein uip-tiedostoon.

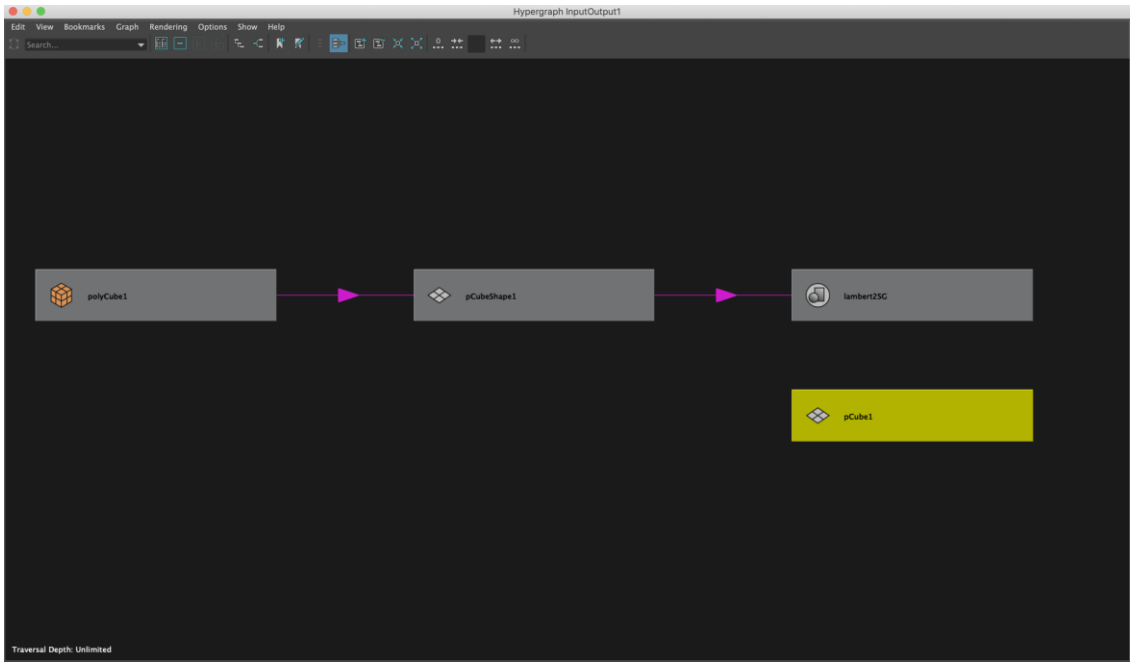
3D Studiossa näkymät toimivat niin kutsutuissa dioissa (engl. slide), joita vaihtamalla voidaan saada eri näkymiä. Perusasetuksina 3D Studio projektissa on päädia (engl. master slide) sekä esitysdia (engl. scene 1 slide). Nykyinen koodi ei suorita tuota hierarkiaa oikein, niin keskityin ensimmäisenä sen korjaamiseen. Ratkaisin ongelman tarkistamalla kirjoitettavien tietosolmujen

järjestystä, ja keskityin seuraavaksi 3D Studion tasoihin (engl. layer). Tasoille kerätään kaikki esityksessä mukana olevat objektit, ja niitä voi olla useampi per dia. Päädiassa on päätaso (engl. master layer), joka on siitä erikoinen, että se pysyy koko esityksen samana ja näkyy jokaisessa dianäkymässä.

Onnistun luomaan tasot ja diat oikein, niin että tuotavat 3D-mallit pitävät sisällään oikeat materiaalit. Tästä tyytyväisenä päätän jatkaa 3D-mallinnusharjoituksia loppupäivän.

Viikkoanalyysi

Tämä viikko oli hyvin kehittävä, ja työ eteni paljon. Tein myös paljon hyvää yhteistyötä muiden tiimin jäsenien kanssa, ja opin heiltä paljon uutta 3D Studioon liittyvää asiaa. Isoin haaste oli ymmärtää Mayan sisällä tapahtuvia toimintoja, kun käytän Maya Commands moduulia. Tätä varten luin paljon Mayan solmujärjestelmästä, jonka ymmärtäminen auttaa paljon. Kuviossa 12 näkyy kuution solmurakenne Mayan riippuvuusnäkymässä ja kuviossa 13 koko esityksen hierarkianäkymä.



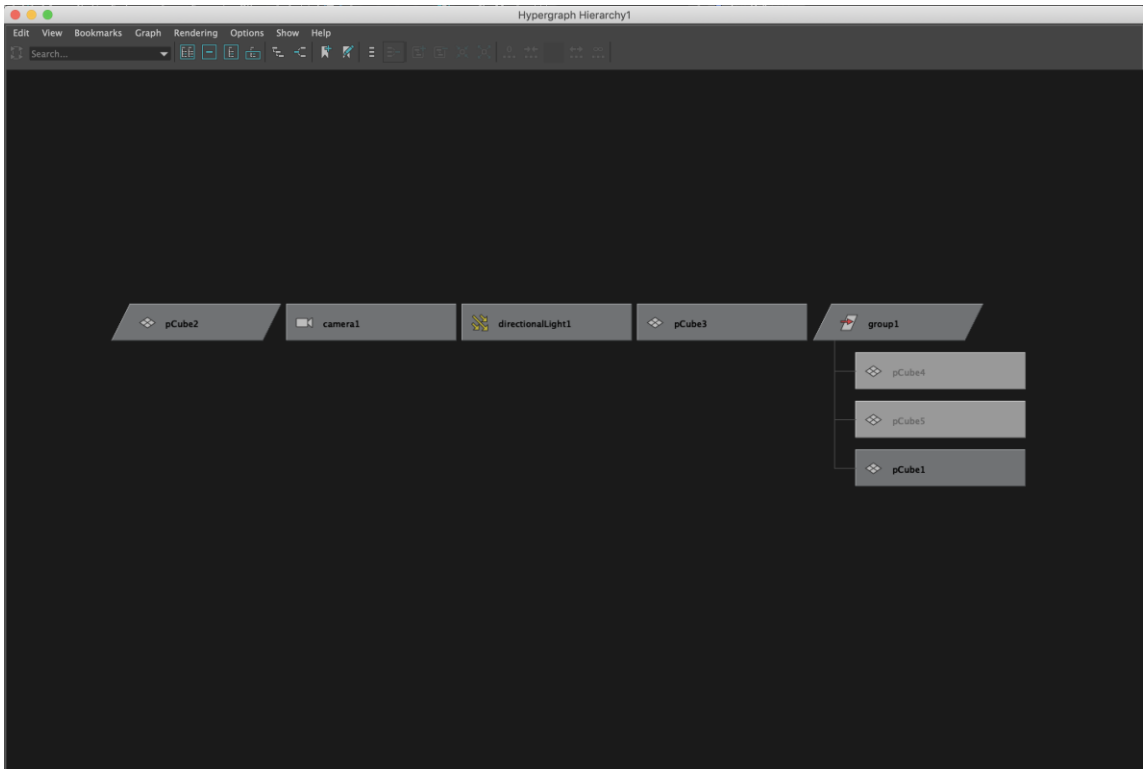
KUVIO 12. Kuution solmuhierarkia riippuvuusnäkymässä

Maya on rakennettu solmujen avulla. Mikä tahansa objekti, kuten laatikko, on rakennettu useammasta solmusta:

- **Luomissolmu (engl. creation node)** rekisteröi tiedot valinnoista, kun laatikko luodaan.
- **Muutossolmu (engl. transform node)** rekisteröi tiedot objektin sijainnista, rotaatiosta sekä skaalasta.
- **Muotosolmu (engl. shape node)** varastoi tiedot laatikon vertekseistä eli tiedot muodoista.

Mayassa käytettävät visuaaliset manipulaattorit, kuten sijainti nuolet, rotaatio pallot tai skaalauskahvat vaikuttavat solmuihin ja niiden ominaisuuksiin. (Maya 2015, viitattu 2.6.2019.)

Mayassa on kaksi tapaa tarkkailla solmuja esitysnäkymässä. Ne ovat hierarkia ja riippuvuus. Hierarkianäkymässä nähdään vanhempi-lapsisuhde solmujen välillä ja riippuvuusnäkymässä nähdään eri solmujen välinen tiedonjako sisään ja ulostulona. (Maya 2015, viitattu 2.6.2019.)



KUVIO 13. Esityksen hierarkianäkymä

Hierarkianäkymässä nähtävä vanhempi-lapsisuhde auttaa erityisesti animoinnissa. Kun vanhempaa muokataan, se vaikuttaa myös sen lapsiin. Riippuvuusnäkymä on tavallaan sarja ohjeita,

kuinka lopullisen 3D-tuotoksen kuuluu näkyä: ”Luo tuo laatikko tuohon, muokkaa sen muotoa tällä tavalla ja anna sille tällainen väriominaisuus”. (Maya 2015, viitattu 2.6.2019.)

DAG hierarkialla (engl. Directed Acylic Graph) tarkoitetaan vanhempi-lapsisuhdetta kaikissa solmuissa, joista objekti muodostuu. Se on tavallaan erittely, kuinka objekti rakentuu geometriasta. Muutossolmut sisältävät sijainnin, rotaation sekä skaalan lisäksi vanhemmuustiedot. Niillä voi olla useampia lapsisolmuja ryhmitettyinä hierarkiassa alaspäin. Nämä ryhmittelyt mahdollistavat esimerkiksi animoinnissa nopeuttavia toimintoja. Muotosolmu ei sisällä muuta kuin muodon luomiseen tarvittavat tiedot. Jokainen näkyvä geometria tarvitsee Mayassa, siis vähintään kaksi solmua. Muotosolmun, joka kertoo, minkä näköinen geometria on ja muutossolmun, joka kertoo missä geometria sijaitse, miten päin se on ja millä skaalalla. (Maya 2015, viitattu 2.6.2019.)

Näiden solmujen ymmärtäminen on työni kannalta välttämätöntä, sillä kaikki tieto, joka siirretään Qt 3D Studioon, haetaan niistä. On oma työnsä löytää Pythonilla oikeat käskyt Maya Commands ja Pymel moduulia käyttäen, mutta onneksi niin monet ovat selvittäneet samoja asioita eri 3D-verkkofoorumeilla. Samalla tulee luettua koko ajan enemmän Mayan dokumentaatiota ja kyky ymmärtää sitä paranee.

Pymel on moduuli, joka sisältää samoja toimintoja kuin Maya Commands tai MEL, mutta sen merkintätapa on hieman erilainen, ja se keskittyy paremmin tietyntyyppisiin toimintoihin Mayan sisällä. En lähde niitä tässä avaamaan, sillä käytän tuota moduulia niin vähän.

3.4 Neljäs viikko: Materiaaliongelmia ja kaatuminen

Maanantai 27.5.2019

Edellisen viikon esimieheni oli työmatkalla, ja ajattelin, että tänään olisi hyvä käydä läpi saavutettuja tavoitteita ja tulevia tehtäviä. Hän vaikutti tyytyväiseltä tehtyyn työhön ja yritin mahdollisimman selkeästi kertoa senhetkisestä tilanteesta. Sovimme, että otan käyttöön työpaikalla käytetyn JIRA-järjestelmän, johon voi merkitä projektikohtaisia bugeja sekä kirjata työtehtäviä.

Tehtävien purkamisen lisäksi kerroin mielipiteeni Maya-työkalun haasteista, ja sovimme, että tutkin aihetta jossakin välissä. Yksi tärkeä toiminto on saada työkalulta tuotu tiedosto automaattisesti tuotua 3D Studioon, ja tämä tarkoittaisi 3D Studion päivittämistä työkalua varten. Kello kahdeltatoista oli tiimipalaveri, jossa kaikki kertoivat taas viimeaikaisista tekemisistään.

Lopun päivää keskityin Python-koodin muokkaamiseen. Yritin selvittää, miten objektit 3D Studiossa tulisivat näkyviin. Tähän asti ne ovat jääneet aina pimentoon, vaikka ne ovat näkyvissä esityksen aikajanalla. Ongelmaksi ilmeni tuotujen valojen Cast Shadows-ominaisuudesta, jonka poiskytkemällä objektit ilmestyivät näkyviin. Toinen ilmennyt ongelma oli, että kaikki siirretyt objektit ilmestyivät peilikuvana 3D Studioon. En ehtinyt kuin huomata tuon ongelman, mutta huomenna jatkan tuon asian selvittämistä.

Tiistai 28.5.2019

Päivän tavoitteeksi otin jatkaa koodaamista. Heti alkuun sain korjattua ongelman, jossa kaikki tuodut objektit olivat horisontaalisesti väärin päin. Tämä ilmeisesti johtui puuttuvasta solmujen orientation-ominaisuudesta, jota en ollut aikaisemmin huomannut ja määritellyt. Tuon määrittelyn jälkeen kaikki tuntui pelaavan oikein. Halusin testata vielä erityyppistä tapausta, ja loin Mayalla uuden työympäristön, jossa oli erilaisia objekteja eri paikoissa. Ohjelma kaatui, kun yritin päivittää objektien sijaintia.

Mayan koodieditori (engl. script editor) vaikuttaa toimivan eri ohjelmaosuudessa, sillä kaikki 3D-näyttämöllä tehty työ pysyi ehjänä ja koodieditorissa tekemäni koodit katosivat. Onneksi olin tietyin

väliajoin tallentanut tehtyä työtä ja sain sitä kautta palautettua eilen valmiiksi saadun koodin. Minulla oli myös tuoreessa muistissa aamulla tehdyt muutokset, joten päivitin ne nopeasti takaisin koodiin.

Tätä tehdessä huomasin, että uudessa Maya-esityksessä olevat materiaalit eivät tulleet objektikohtaisesti uip-tiedostoon. Toinen ongelma oli, että objektit ilmestyivät taas peilikuvana, vaikka lisäsin orientation-ominaisuuden objektien solmuihin.

Aloitin materiaaleihin liittyvän ongelman selvittämisen, ja tein sitä lopun päivää. Suurin ongelma on ymmärtää, milloin käsittelee mitäkin osaa Maya-objektista ja miten eri osiin pääsee käsiksi. Mayalla on käytössään Maya Commands sekä PyMel moduulit ja kummallakin on omat ominaisuutensa, ja jo tehty koodi yhdistelee niitä molempia. Aion jatkaa huomenna tämän materiaali ongelman selvittämistä.

Keskiviikko 29.5.2019

Tämän päivän tavoitteeksi otin saada materiaaliasia selvitettyä koodipuolella. Ensimmäisenä aamulla etsin Buddyn ja menimme työpisteelläni. Näytin hänelle tehtyjä työvaiheita ja ongelmia, joita työstän tällä hetkellä. Minun pitäisi saada selville, kuinka yksittäiselle 3D-mallille saadaan iteroitua oikeat materiaalit.

Tekemäni työkalun merkitsemät materiaali näkyvät uip-tiedostossa, mutta ne eivät tulleet näkyviin 3D Studiossa oikein. Sisään tuotu fbx-tiedosto sen sijaan näyttää materiaalit oikein. Avasin toimivan uip-tiedoston, ja huomasin, että sen materiaaliviitesolmuissa on useaan liitetty ylimääräinen 001-pääte. Tämä viittaisi siihen, että ohjelma tekee näistä itse uniikkeja ja että se saattaisi olla syy, miksi työkaluni tekemät materiaalit eivät näy. Kävin 3D-tiimin luona keskustelemassa materiaalipuoleen erikoistuneen työkaverin kanssa, ja hän vahvisti ajatukseni oikeaksi.

Lisäsin koodiin toimintoja, jotka tarkistavat, onko materiaalin nimi uniikki kokonaisuutta sekä objektia kohden. Tarpeen mukaan koodi sitten lisää juoksevan luvun materiaaliviitteen tunnukseen. Tarkistin koodin, ja se näytti samalta kuin mallikappale.

Tarkistin tilanteen 3D Studiossa, ja kaikki materiaalit ilmestyvät, mutta vain väärässä järjestyksessä. Tämä tapahtuu, jos yhdellä objektilla on useampi kuin yksi materiaali. Koodia tarkkailemalla ja parilla nopealla testillä löysin kohdan, joka vaikuttaa tähän. Se on objektin alla oleva materiaaliviitejärjestys.

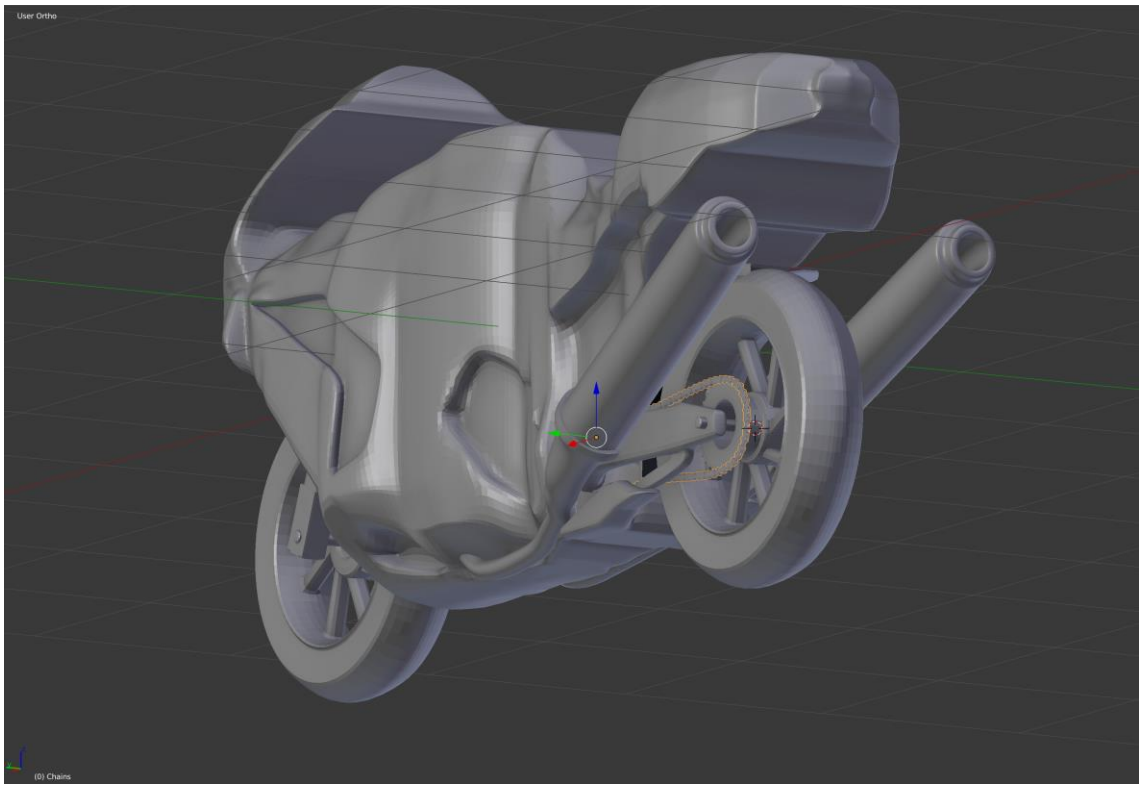
Yritin selvittää, miten materiaalit järjestyvät Mayasta ja miten se päätyy uip-tiedostoon. En löytänyt tähän ratkaisua, vaikka muillakin on verkossa ollut vastaavaa ongelmaa esimerkiksi pelimoottoreiden kanssa työskennellessä. Ideoimme esimieheni kanssa, että 3D Studion puolelta voitaisiin vaikuttaa järjestykseen niin, että voin matkia sen järjestämistavan omaan koodiini.

Perjantai 31.5.2019

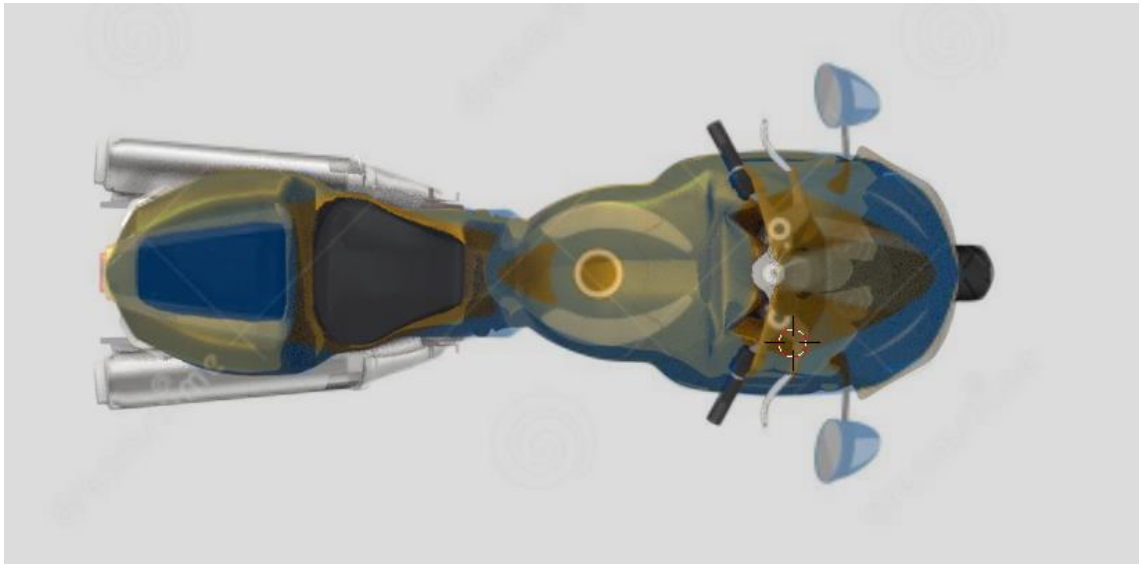
Edellinen päivä oli helatorstai, joten en tehnyt töitä silloin. Tälle päivälle olin ajatellut keskittyä hieman tällä viikolla tehtyyn koodiin sekä kouluttaa itseäni. Sähköpostiini oli tullut viesti uusien työkalujen aktivoinnista. Latasin uudet 3D-työkalut ja testailin niitä lyhyesti. Niiden käyttötarkoitus on 3D-mallinnuksen myöhäisempää käsittelyä varten, kun malli on valmis ja sille aletaan luoda materiaaleja ja tekstuureja.

Testasin tehdyllä koodilla valojen toimivuutta, ja ne toimivat yllättävän hyvin. Buddy ei ollut tänään paikalla, joten en saa häneltä vahvistusta, kuinka hyvälle tasolle valojen siirtyminen Mayasta 3D Studioon pitää saada. Sijainnit, rotaatiot ja skaalat tulivat samoilla arvoilla molemmissa ohjelmissa.

Jatkoin Blenderillä 3D-malliharjoitusta, jossa tein moottoripyörää suunnittelukuvista. 3D-mallinnus sujuu mukavasti ja uskoin että pääsen pian testaamaan esimerkiksi tänä päivänä aktivoituja työkaluja siihen. Pyörän runko alkoi olla hyvällä mallilla, ja työstin enemmän hankalia yksityiskohtia, kuten kettinkiä ja pienempiä osia. Kuvioissa 14 ja 15 näkyy mallin kehittyminen.



KUVIO 14. Pieniä yksityiskohtia



KUVIO 15. Pyörän muoto yläsuunnasta

Loppupäivästä kävin puhumassa esimieheni kanssa työkalun käyttötavasta, ja hän kehotti minua tutustumaan Mayan muihin lisäosatyökaluihin. Etsin ja asensin niistä muutaman. Minusta tuntui, että skriptipohjainen työ olisi ollut järkevä ratkaisu, sillä muutamalla ohjeella Mayan peruskäyttäjä

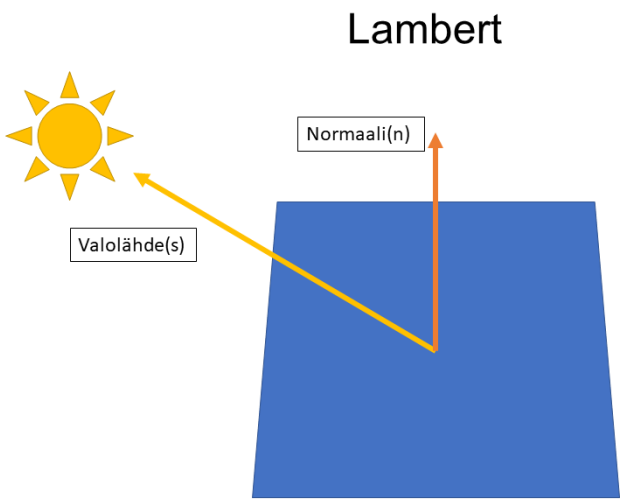
pystyy käyttämään tekemäämme koodia. Tämä tarkoittaa sitä, että käyttäjä käynnistää itse koodin käyttäen Mayan koodieditoria. Tutkin tietenkin löytyikö tähän muuta vaihtoehtoa.

Viikkoanalyysi

Tällä viikolla keskityin erityisesti materiaalien toimintaan. Opin hieman lisää Mayan materiaalipuolesta ja testailemalla opin kuinka materiaalit sijoittuvat 3D Studiossa objekteihin. Olen tyytyväinen, että nuo asiat selvisivät näin varhaisessa vaiheessa ja että minulla oli jokin idea, kuinka tuo voitaisiin korjata. Työkalun koodaamisessa en keksinyt itse mitään uutta, mutta löysin ja sovelsin toisten tekemää koodia, jolla meidän työkalumme toimii tarkemmin. Uskon, että minun pitää opetella enemmän 3D-mallinnuksen teoriaa, että ymmärtäisin paremmin näitä asioita.

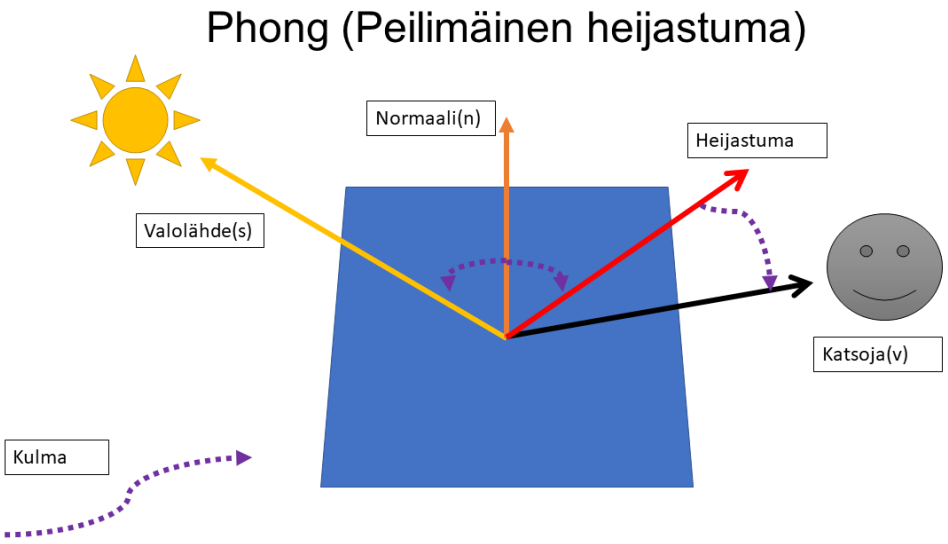
Tämän viikon aikana olen kehittynyt erityisesti uip-tiedoston tulkinnassa. Objektien horisontaalinen kääntyminen sekä monimateriaaliongelma ratkesivat nopeasti, kun osasin tulkita koodista niihin vaikuttavat asiat ja sain tiimiltäni vahvistusta näille. Tein myös ensimmäistä kertaa enemmän töitä materiaalin parissa.

Materiaalisolmut antavat objektit pinnalle ne ominaisuudet, jotka halutaan näkyvän lopputuloksessa. Materiaalisolmut määrittelevät pinnan käyttäytymisen valon kanssa. Pintamateriaalit (engl. surface materials) ovat yleisiä, ja niille voidaan määrittää esimerkiksi värin, heijastumisen, peilautumisen, läpinäkyvyyden ja valovoimaisuuden. Siirtymämateriaaleilla (engl. displacement material) voidaan luoda joko illuusio tai oikeaa lisägeometriaa objekteihin käyttäen tekstuurikarttoja. Tilavuusmateriaalilla (engl. volumetric material) voidaan luoda näyttämön lopputulokseen oikean maailman tuntuisia tilaefektejä kuten savua ja pölyä. (Maya 2019, viitattu 15.7.2019.)



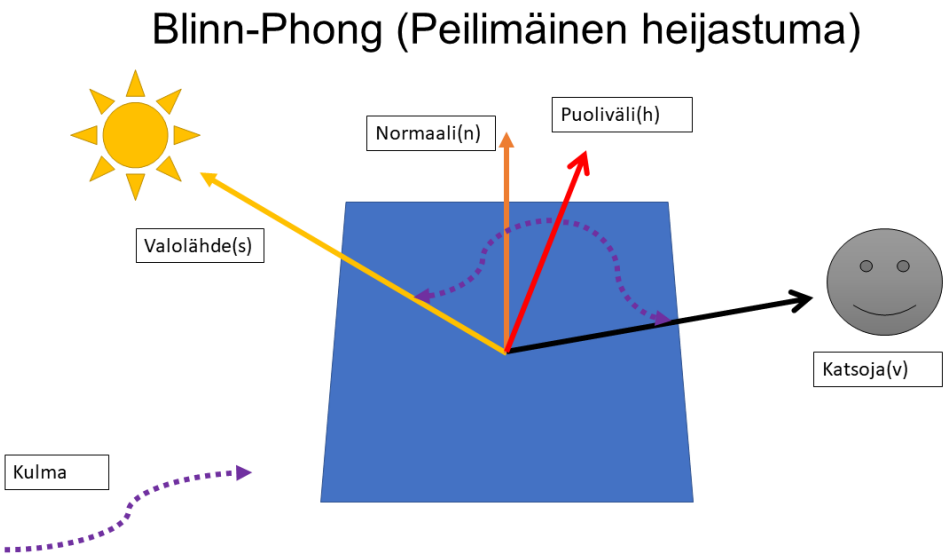
KUVIO 16. Lambert

Materiaaleja käsiteltäessä on tärkeää myös ymmärtää jotain 3D-mallintamisessa käytetyistä valoista. Kaikkein yksinkertaisin ja todella yleinen valotustyyppi on Lambert, joka esitetään kuviossa 16. Se ottaa huomioon valonlähteen sekä valaistavan pinnan normaalin suunnan. Mitä suoremmassa normaalin kulma on valonlähteeseen, sitä tehokkaamman valaistuksen pinta saa. (Byl 2019, Viitattu 15.7.2019.)



KUVIO 17. Phong

Toinen yleinen valaistus on Phong, joka esitetään kuviossa 17. Tämä valaistus laskee, kuinka valo heijastuu valotettavalta pinnalta kameralle. Heijastuma on voimakkaimmillaan pinnan normaalin vastapuolella, kun kulma on sama kuin valonlähteellä. Tähän voimakkuuteen vaikuttaa materiaalin heijastusominaisuudet sekä katsojan kulma heijastukseen. Heijastumat myös säteilevät pinnalta ympäristöön (engl. fall off) ja tähän vaikuttavat juuri pinnan materiaalin ominaisuudet. Tämä valaistustekniikka vaatii paljon laskutoimintoja. (Byl 2019, Viitattu 15.7.2019.)



KUVIO 18. Blinn-Phong

Blinn-Phong on tuoreempi valaistustekniikka, johon on lisätty puolivälin laskenta. Sen toiminta visualisoidaan kuviossa 18. Puolivälin laskeminen nopeuttaa valon heijastumisten laskemista, sillä kaava on aina sama. Puoliväli on valonlähde lisättynä katsojaan. Tällä valaistuksella ei saavuteta niin näyttäviä tuloksia, mutta sen nopeuden vuoksi sitä voidaan käyttää esimerkiksi peliteknologioissa hyväksi. (Byl 2019, Viitattu 15.7.2019.)

3.5 Viides viikko: Ensimmäinen toimiva versio ja materiaalien käsittelyä

Maanantai 3.6.2019

Tämän päivän tavoitteeksi otin selventää työtehtäviä, että Maya työkalun tekeminen etenisi tehokkaammin. Aamupäivästä tarkistin viime viikon töitä ja kävin keskustelemassa esimieheni kanssa. Tämän jälkeen Buddy tuli työpisteelleni ja esitin hänelle viime viikon saavutuksia.

Periaatteessa kaikki työtä varten olevat koodikappaleet ovat jossain muodossa valmiina ja niistä pitäisi kasata nyt ensimmäinen versio. Pohdimme myös yhdessä viime viikolla noussutta materiaaliongelmaa ja hänellä oli idea fbx-tiedoston materiaalijärjestyksen toiminnasta. Kytkimme Mayassa Polygonien näytä pinnat-ominaisuuden päälle. Se näyttää jokaisen pinnan tekemisjärjestyksessä ja antaa järjestysnumeron. Polygonimalli, jolle on määritetty useita materiaaleja, vaikutti saavan niiden järjestyksen sitä mukaa, kun pintojen järjestysluku kasvoi.

Päätin tulevat työtehtävät, joihin aion keskittyä tulevina päivinä:

- Kokoa ensimmäinen yksinkertainen versio työkalusta
    - Yhdistä fbx- ja uip-tiedoston tekemiskoodit
    - Luo yksinkertainen käyttöliittymä
- Korjaa työkalun materiaalien indeksointi vastaamaan 3D Studiota
- Selvitä, kuinka objektista saadaan kerättyä materiaalit pintojen järjestyksen mukaan.

Tiimikokouksessa ilmoitin nämä loppuryhmälle ja selvitin materiaaliasiaa lopun päivää.

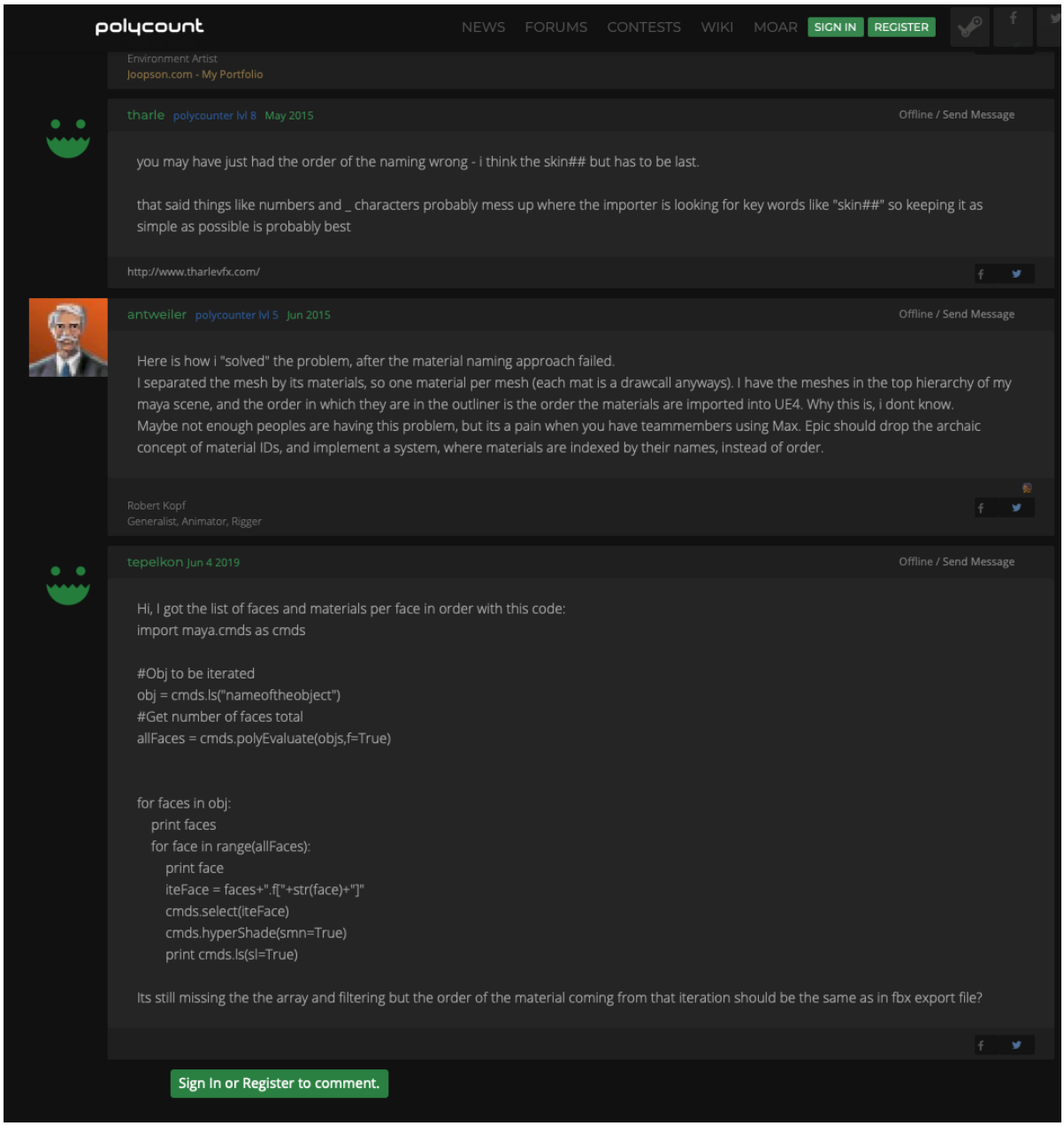
Tiistai 4.6.2019

Edellispäivänä olin aloittanut materiaalin selvittämisen pintakohtaisesti polygonista. Tavoitteeni tälle päivää oli saada tuo järjestys toimimaan koodissa niin, että saan järjestettyä uip-tiedostossa olevan materiaalijärjestyksen vastaamaan fbx-tiedostoa. Edellispäivänä minua vaivasi, miksei tekemäni koodi toiminut.

Heti aamusta testasin ideat, joita edellisenä päivänä olin kehitellyt, mutta ne eivät toimineet. Jatkoin keskustelufoorumeiden selailua, etsin mahdollisia verkkokursseja asiaan liittyen ja yritin googlella löytää uusia asiaan liittyviä sivuja. Laitoin myös viestiä Buddylle, että voisiko hän tulla vilkaisemaan ongelmaa tuoreilla silmillä.

Esittelin hänelle edellispäivän aikana tehtyjä koodiyrityksiä. Olin yrittänyt omasta mielestä kaikkea, mitä Mayaan liittyvillä foorumeilla aiheesta oli ollut keskustelua. Yksi metodi toimi, mutta vain jos klikkasin käyttöliittymässä hiirellä haluttua polygonia tai pintaa. Se ei toiminut, jos yritin syöttää koodiin kohdetta toisenlaisena parametrinä. Selitin, mikä koodissa toimii ja mikä ei. Buddy tarkkaili koodia ja ehdotti pientä muutosta. Siirsin erään koodin pätkän ulos iteraatiokohdasta, jota työstin ja koodi alkoi toimia osittain. Kävin läpi kohdat ja paransin koodia. Pian koodi kävi läpi oikein valitun objektin eri pinnat ja ilmoitti niille määrätyt materiaalit.

Olin todella tyytyväinen, että koodi alkoi toimimaan, sillä en ollut törmännyt vastaavaan Maya keskustelufoorumeissa, jossa pohdittiin samaa pulmaa. Kirjauduinkin kahdelle foorumille, joista olin hyötynyt eniten ja julkaisin niissä tekemäni koodin siltä varalta, että siitä olisi jollekin muulle hyötyä tulevaisuudessa. Kuviossa 19 näkyy kaappaus viestistäni.



KUVIO 19. Ratkaisuni Polycount foorumille.

Seuraavaksi keskityin saamaan materiaalien uniikin indeksoinnin valmiiksi. Jos 3D Studiossa on kaksi samannimistä materiaalia samassa projektissa, niin uudempi saa 001-päätteisen tunnuksen. Tuo luku tietenkin kasvaa, jos materiaaleja on useampia, ja tämä pitää ottaa myös huomioon tekemässäni koodissa. Tein koodia loppupäivän, ja eri tapojen testaamisen jälkeen päädyin versioon, jossa jokainen materiaali tallennetaan sellaisenaan listaan. Materiaalin uniikkiutta tarkistaessa lasketaan, montako edellistä versiota on jo listalla, ja siitä saadaan indeksiluku.

Keskiviikko 5.6.2019

Tarvittavat koodit työkalua varten ovat siinä vaiheessa, että niistä voisi kasata ensimmäisen toimivan version. Tämän päivän tavoite oli keskittyä tähän. Ensimmäisenä piti selvittää miten työkalu latautuisi itsestään, kun Maya käynnistetään. Löysin nopealla Internetistä hakemisella foorumin, jossa pohdittiin tapoja ladata skripti itsenäisesti Mayan sisällä.

Suunnitelmani on luoda tällä itsestään latautuvalla skriptillä oma valikko Mayan valikoiden lisäksi. Tuolta valikosta voi käynnistää Qt 3D-työkaluikkunan, jossa kaikki toiminta tapahtuu. Tekemästäni koodista valmiita osia on työkaluikkuna, joka tekee myös mallien viemisen fbx:nä sekä Qt 3D Studio-projektikansion tarkistuksen. Tämä koodi on tehty oliomuotoon. Toinen valmis koodi on Maya-esityksen skannaava uip-tiedoston luomiskoodi. Tämä koodi on proseduraalinen.

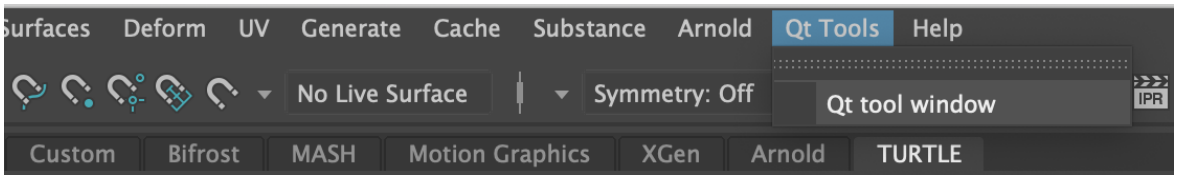
Olio-ohjelmoinnissa asioita kuvataan olioina, joille määritellään toiminnot niihin liittyvinä metodeina sekä muuttujatiedot, joita ne voivat pitää sisällään. Olion kuvaamisen jälkeen siitä kutsutaan instanssi, joka on olion ilmentymä. Tuo ilmentymä suorittaa sille määriteltyjä metodeja ja päivittää sekä käyttää muuttujatietojaan. Olio-ohjelma rakentuu useiden olioiden välisestä viestinnästä ja tiedon jakamisesta kutsumalla toistensa metodeja. Tämän monimuotoisuuden vuoksi päätin, että työkalun ikkuna on oltava oliomuodossa (Sorva 2017, viitattu 16.7.2019.)

Proseduraalisessa ohjelmoinnissa määritellään funktioita, jotka eivät liity olioihin. Ohjelmassa on pääfunktio, joka käskee alifunktioita suorittamaan niille määritetyt tehtävät. Tieto liikkuu funktioiden parametreissä sekä palautuksissa. Myös globaalien muuttujien käyttäminen on mahdollista, mutta ei aina suositeltavaa. Selkeän toiminnallisuuden vuoksi olin päättänyt, että uip-tiedoston luova koodi on proseduraalisessa muodossa. (Sorva 2017, viitattu 16.7.2019.)

Olin enimmäkseen työstänyt proseduraalista koodia, ja olio-Python-koodin tekeminen tuotti hieman hankaluuksia ja eteni hitaasti. Tähän vaikutti vielä Mayan koodieditoriympäristö, joka ei päivittänyt koodeja toimimaan varmasti uudella tavalla, ellei ohjelmaa käynnistänyt uudestaan. Sain kuitenkin edettyä uuden koodin kanssa pienillä muutoksilla ja testaamalla.

Päivän aikana sain luotua monta haluamaani ominaisuutta työkalulle. Mayan käynnistyessä sen työkaluvalikkoihin ilmestyy tekemäni valikko Qt-työkaluille. Kuvio 20 havainnollistaa tuota valikkoa. Sieltä valitsemalla Qt-vientityökalun aukeaa työkaluikkuna, josta voi valita projektikansion, joka

sisältää nappeja. Yhtä nappia painamalla työkaluikkunan olioinstanssi kutsuu toista tekemääni moduulia, jonka olen testiksi tehnyt.



KUVIO 20. Työkaluvalikko

Torstai 6.6.2019

Ensimmäinen versio toimivasta työkalusta oli niin lähellä valmistumista, että otin sen tämän päivän tavoitteeksi. Heti aamusta jatkoin edellisen päivän koodin työstämistä.

Tein uutta koodia aika isoissa erissä, ja testasin sitä tasaiseen tahtiin. Yllätyin, kun ongelmia ei tullutkaan, ja pian minulla oli ikkuna, jolla pystyi valitsemaan juurikansion, nimeämään tiedoston ja viemään uip- sekä fbx-tiedoston halutulla nimellä juurikansioon.

Pyysin Buddya vilkaisemaan senhetkistä tilannetta. Testasimme työkalua, ja ongelma syntyi heti, kun yritimme luoda samannimisen tiedoston vanhan päälle. Täysin uuden tiedoston luominen ei ole tuottanut ongelmia. Sovimme, että aloitan työkalun testaamisen eri tavoilla ja jatkan tuon vian selvittämistä.

Toimistoomme ilmestyi uusia kalusteita, ja työpisteeni siirtyi 3D-tiimin tiloihin. Uudella työpisteellä jatkoin koodin selvittämistä ja lisäsin siihen uusia tarkistuksia. Halusin, että ohjelma tarkistaa erilaisia käyttäjätilanteita. Jos käyttäjä ei kirjoita tiedostonimeä, se sisältää sopimattomia kirjaimia tai kun siirrettäviä objekteja ei ole olemassa Maya-esityksessä, työkalu ei antanut tallentaa tiedostoja,

Ongelmia tuotti edelleen saman xml-tiedoston luominen kahdesti. Ongelmaa ei ollut, jos käytin pelkkää proseduraalista koodin pätkää, mutta kun ajoin saman koodin oliokoodin kautta, vika ilmeni. Aioin jatkaa asian selvittämistä seuraavana päivänä.

Perjantai 7.6.2019

Edellisenä päivänä taistelin xml-tiedoston kanssa, joka ei suostunut uudelleenkirjoittamaan itseään. Tämän päivän tavoitteeksi otin selvittää tämän ongelman.

Testasin ensimmäisenä koodia uudestaan, ja se toimi omana skriptinään. Heti, kun skriptiä käytettiin oliomuotoisesta ikkunaskriptistä, se lakkasi toimimasta ja varoitti, että kahta dokumenttia ei saa olla olemassa. Päättelin, että tämä liittyy jotenkin olion instansseihin. Valikkokoodi luo aina vain yhden ikkuna olioinstanssin ja tarkistaa jos yritetään luoda uutta ja vanha on olemassa. Se poistaa tuolloin vanhan instanssin uuden tieltä.

Kokeilin pienemmän xml-tiedoston luomista omassa skriptissään, ja tein koodin omana olioluokkanaan. Tuosta luokasta tein instanssin ikkunaskriptissä, ja se ei vieläkään suostunut kirjoittamaan uutta xml-tiedostoa samalla nimellä vanhan päälle. Lisäsin vanhan instanssin poistamisen, mutta koodi ei vieläkään toiminut. Vasta, kun yritin proseduraalisen moduulin uudelleen lataamista ikkunaskriptistä, koodi alkoi toimimaan.

Seuraavaksi kokeilimme Buddy kanssa, kuinka työkalu toimii Windows laitteella. Saimme virheilmoituksen kameraan liittyen, ja aloitin sen selvittämisen. Vika johtui kamerasolmun huonosta määrittelystä. Asiaa selvittäessä löysin muitakin vikoja, jotka liittyivät kovakoodattuihin kansioreitteihin. Kovakoodauksella tarkoitan muuttuja-arvoja, jotka ovat määritelty kerran ja joihin funktioiden toiminnat eivät vaikuta.

Loppupäivästä selvitin vielä ongelmaa, jossa kaikki objektit ilmestyivät peilikuvana, vaikka niiden sijaintiarvot olivat Mayassa ja 3D Studiossa samat. Ongelma ratkesi, kun Mayassa sijainti, rotaatio ja skaala niin sanotusti jäädytettiin nollaan ennen fbx-tiedoston vientiä.

Viikkoanalyysi

Tämän viikon aikana selvitin mielestäni todella useaa asiaa ja työkalun valmistuminen eteni hyvin. Sain kaikki koodiosiot valmiiksi ensimmäistä toimivaa versiota varten. Selvitin polygoni mallien

materiaalijärjestyksen Mayassa, mihin en löytänyt mistään suoraa ohjetta. Kuviossa 21 näkyy tekemäni koodi.

```
# CREATE MATERIALS
print name
#print doObj
obj = cmds.ls(str(name))
allFaces = cmds.polyEvaluate(obj,f=True)
for faces in obj:
    #print faces
    for face in range(allFaces):
        #print face
        iteFace = faces+".f["+str(face)+"]"
        cmds.select(iteFace)
        cmds.hyperShade(smn=True)
        material = cmds.ls(sl=True)
        origName = material
        #print material[0]
        if not material[0] in objMaterials:
            objMaterials.append(material[0])
```

KUVIO 21. Materiaali per pinta

Loin materiaalien iteroinnille tarkistusfunktiot, joilla tarvittaessa materiaali saa uniikin numeroinnin nimeensä. Kuviossa 22 näkyy tätä varten luotu koodi.

```
#CHECK IF UNIQUE INDEX NUMBER NEEDS TO BE ADDED AND ADDS
for objMaterial in objMaterials:
    #print objMaterial
    originalName = objMaterial
    eachMaterial.append(objMaterial)
    #print eachMaterial
    if objMaterial in allMaterials:
        #print "It is already"
        indexNum = eachMaterial.count(objMaterial)-1
        #add _001 or running number
        objMaterial=str(objMaterial)+"_"+str(indexNum).zfill(3)
        materialIndex += 1

        allMaterials.append(str(objMaterial))
        finalMaterials.append(str(objMaterial))
        # CREATE material REF for materialdef to Master Slide
        scene_ref = doc.createElement("Add")
        scene_ref.setAttribute("ref","#"+objMaterial)
        scene_ref.setAttribute("name",originalName)
        scene_ref.setAttribute("referencedmaterial","#materials/"+originalName)
        scene_ref.setAttribute("sourcepath","../materials/"+originalName+".materialdef")
        #assign to parent slide
        if animExists:
            substate_node.appendChild(scene_ref)
        else:
            state_node.appendChild(scene_ref)
    else:
        allMaterials.append(str(objMaterial))
        # CREATE material REF for materialdef to Master Slide
        scene_ref = doc.createElement("Add")
        scene_ref.setAttribute("ref","#"+objMaterial)
        scene_ref.setAttribute("name",originalName)
        scene_ref.setAttribute("referencedmaterial","#materials/"+originalName)
        scene_ref.setAttribute("sourcepath","../materials/"+originalName+".materialdef")
        #assign to parent slide
        if animExists:
            substate_node.appendChild(scene_ref)
        else:
            state_node.appendChild(scene_ref)
        finalMaterials.append(str(objMaterial))
```

KUVIO 22. Materiaalien tarkistus ja indeksointi

Selvitin, kuinka Mayassa saadaan skripti lataantumaan ohjelman käynnistyessä. Tämä koodi yllätti kaikessa yksinkertaisuudessaan. Se näkyy kuviossa 23. Koodissa Mayan käynnistyessä ladataan valikon oma moduuli ja kutsutaan sen luomismetodia.

```
import maya.cmds as cmds
import qtmenu
reload(qtmenu)

def starter():
    print "Qt starter here"
    qtmenu.createMenu()

scriptJobNum = cmds.scriptJob(event = ["NewSceneOpened",starter])
```

KUVIO 23. Maya käynnistäjäkoodi

Xml-tiedoston uudelleenkirjoitusongelma selvisi, vaikken tiedä tarkkaa syytä, mistä se johtui. Sain myös siirrettyä työkalun onnistuneesti Windows-laitteelle, ja pääsin toteamaan, että se toimii siellläkin.

Olen tyytyväinen viikon aikana tehtyihin päivityksiin. Oli hienoa huomata, että koodiongelmien kanssa auttaa sama keino, jota olen käyttänyt aikaisemminkin työelämässä. Yritän ratkaista ongelman kaikin tavoin ensin itse, ja sitten pyydän jonkun henkilön tuoreilla silmillä vilkaisemaan ongelmaa. Hän näkee yleensä ongelman todella nopeasti, sillä olen itse sokaistunut tekemälleni työlle.

3.6 Kuudes viikko: Käyttöliittymän parantamista ja merkistöongelma

Maanantai 10.6.2019

Edellisenä perjantaina ratkaisin 3D-mallien peilikuvakäyttäytymisongelman, ja tämän päivän tavoitteeni oli lisätä tämä korjaus koodiin. Työkalun toimintaa parantava toimenpide on nimeltään Freeze transformation-toiminto. Se jäädyttää polygonien sijainti-, rotaatio- sekä skaala-arvot nolliin.

Tein tämän toimenpiteen muutaman kerran ja keräsin konsolista toimenpiteen tiedot ylös. Tarkistan vielä Autodesk-verkkosivulta, mitä mikäkin asetus tekee. Selvisi, että tämä ei koske valoja tai kameraa. Lisäsin koodiin toiminnon ja testasin, miltä viety esitys näytti 3D Studiossa. Objektit tulivat oikein, mutta kamera ja valot näyttivät tulevan väärin z-sijaintiulottuvuudella.

Kameran z-arvo oli 19, ja kun muutin sen-19, se näytti oikealta. Päätin kertoa koodissa kameroiden ja valojen z-arvon miinus yhdellä, jolloin positiiviset arvot muuttuivat negatiiviksi ja toisin päin. Nyt esitykset näyttivät ilmestyvän oikein.

Seuraavaksi ryhdyin ratkaisemaan seuraavaa isoa puutetta työkalussa. Koodissa ei otettu huomioon ryhmiä. Minun piti miettiä logiikka, kuinka ryhmät saadaan luotua niin, että ne koskevat kaiken tyyppisiä objekteja eivätkä sekoita esitystiedostoa.

Tiistai 11.6.2019

Tämän päivän tavoitteena oli ratkaista ryhmien luomis- ja hierarkiaongelma. Tehdyssä koodissa oli jo funktio, joka loi ryhmän ja lisäsi sen yhden pääsolmun alaiseksi. Minun piti keksiä, miten työkalu käsitteli ryhmät ja osasi lisätä objektit ja ryhmät oikeassa järjestyksessä oikeisiin solmuihin.

Ensimmäiseksi loin funktion, joka tunnistaa ryhmät. Mayassa ryhmät ovat muuttuja (engl. transform)-solmuja, joilla ei ole muita ominaisuussolmuja liitettynä itseensä. Muilla objekteilla on aina jokin solmu muuttujan lisäksi. Ryhmäsolmu sisältää myöhemmin ryhmän jäsen solmuja. Löydettyjen ryhmien nimet lisätään globaaliin listamuuttujaan talteen. Tämä lista iteroidaan vielä kertaalleen

create_group()-funktiolla, jolloin jokaisesta ryhmästä luodaan oma solmu. Kuviossa 24 näkyy tekemäni koodilisäykset objektien iterointiin.

```
def loop_objects():
    global doObj
    global cam
    global objGroups
    # Find all groups in scene
    getTrs = cmds.ls(type="transform")
    for transformObj in getTrs:
        if not cmds.listRelatives(transformObj,s=True):
            print transformObj
            objGroups.append(transformObj)
    #Create group nodes
    for grp in objGroups:
        create_group(grp)
```

KUVIO 24. Etsi ja luo ryhmät

Jokaista 3D Studion hyväksymää objektityyppiä varten on olemassa omat create_object-funktio, joka ottaa argumentiksi annetun nimen. Näissä funktiossa luodaan objekteille omat solmut ja lisätään ne uip-tiedoston hierarkiaan. Lisäsin loppupäähän tarkistuksen, joka selvittää ilmestyykö argumentissa annettu nimi jonkin ryhmän alaisena. Nämä tarkistukset ilmenevät kuvioissa 25 ja 26.

```
    add_node.setAttribute("orientation","Right Handed")
    add_node.setAttribute("rotationorder","XYZr")
    #CHECK IF BELONGS TO Group
    results = checkInGroups(str(name))
    print(results)
    groupnode = results[1]
    if results[0] == True:
        print "Group detail:"+groupnode
        print groupNodes
        # ASSIGN TO GROUP
        addToGroup(camera_node,groupnode)
    # ASSIGN TO PARENT
    else:
        layer_node.appendChild(camera_node)
    state_node.appendChild(add_node)
```

KUVIO 25. Kamera solmun tarkistus

```
#Check if object appears in any of the groups and return True and name of the group
def checkInGroups(item):
    global objGroups
    for groupp in objGroups:
        print groupp
        list = cmds.listRelatives(groupp)
        for obj in list:
            if str(obj)==str(item):
                print item+" is found in list"+str(groupp)
                return [True,groupp]
    print item+" not found in groups"
    return [False,None];

#Adds obj to group whrere it belongs
def addToGroup(obj,groupname):
    global groupNodes
    for grp in groupNodes:
        if grp.attributes["name"].value == groupname:
            grp.appendChild(obj)
            print obj.attributes["id"].value+"Was added to group "+grp.attributes["name"].value
        else:
            print "Not in this group"
```

KUVIO 26. Ryhmien käsittelyfunktiot

Kun tarkistusfunktio löytää annetun objektin nimen, jonkin ryhmän alasolmuista, se lisää objektisolmun siihen. Jos nimi ei löydy minkään ryhmän solmuista, niin objekti lisätään yleissolmun alle.

Tekemäni koodi toimi niin kuin pitikin. Luodut ryhmät ja ryhmien sisäiset ryhmät tulivat oikein uip-tiedoston hierarkiaan.

Keskiviikko 12.6.2019

Työkalun toimintaa ei oltu testattu vielä kunnolla Windows-järjestelmässä, joten tämän päivän tavoitteena oli käydä tätä läpi. Olin ottanut koulutietokoneeni mukaan, jossa on Windows-käyttöjärjestelmä sekä Maya 2017 asennettuna. Siirsin tarvittavat viisi skriptitiedostoa tietokoneelleni ja käynnistin Mayan.

Työkalu käynnistyi hyvin, ja loin yksinkertaisen esitysnäkymän, jonka voisin siirtää 3D Studioon. Asetin tiedostopolun, nimesin tiedoston, valitsin objektit ja aktivoin työkalun. Sain heti "UnicodeEncodeError #Ascii"-vikailmoituksen. Ensimmäiseksi ajattelin, että tämän pitää liittyä pelkästään Windows-käyttöjärjestelmään. Tarkistin, mitkä asiat olivat erillä tavalla kuin normaalisti käyttämälläni Mac-koneella, ja löysin mahdollisen syyllisen vikaan. Antamassani kansiopolussa oli ö-kirjain. Mac-tietokoneella en ollut käyttänyt skandinaavisia kirjaimia ollenkaan. Loin vastaavanlaisen kansion, jossa oli ö:tä ja ä:tä, ja testasin myös Mac-tietokoneella. Sama vikakoodi ilmestyi. Aloitin vian selvittämisen Mac-työkoneellani.

Tietokone ei tallenna kirjaimia, numeroita tai kuvia. Se tallentaa bittejä. Bitillä voi olla kaksi arvoa. Kyllä tai Ei, Totta tai Ei totta, 1 tai 0. Jotta nämä bitit voisivat esittää mitään, tarvitaan sääntöjä. Tätä varten on olemassa merkistöt, joita on useita eri symbolikokoelmille. Esimerkiksi A-kirjain Ascii-merkistössä on 01000001. Ascii-merkistö sisältää 95 ihmisluettavaa merkkiä mukaan lukien kirjaimet A:sta Z:hen pieninä ja suurina sekä pilkut. Yhteensä se sisältää 128 merkkiä ja oli pitkään käytetyin merkistö. Se ei kuitenkaan riittänyt kuin osaan merkeistä, joita maailmalla käytetään. (Zentgraf 2015, Viitattu 16.7.2019.)

Lopulta luotiin Unicode-merkistö, josta piti tulla uusi standardi yhdistäen kaikki merkistöt. Se ei käytä bittejä, ja se voi sisältää kaikkien maailman maiden kielien symbolit. Unicode käyttää

taulukkoa, johon mahtuu 1 114 112 merkkisymbolia. Tämän taulukon koodaamiseen biteiksi on olemassa useita enkoodauksia. Yleisimmät ovat UTF-8, UTF-16 ja UTF-32. Merkki A UTF-8 enkoodauksella on bitteinä 01000001 ja UTF-32 enkoodauksella 00000000 00000000 00000000 01000001. (Zentgraf 2015, Viitattu 16.7.2019.)

Ensimmäisenä paikansin koodista kohdan, jossa vikakoodi ilmenee. Vika ilmeni ensimmäisenä fbx-tiedostoa luodessa. Luomisfunktio otti parametrinä tiedostopolun string-muodossa sekä jonkin asetuksen totuusarvona. Vika oli selvästikin string-muuttujassa.

Hain hakukoneella tietoja vikailmoituksesta, ja löysin useita keskusteluita koskien Python-kielen merkistöä (engl. encoding). Maya käyttää 2.7 versiota Pythonista, joten rajasin tiedonhaun siihen liittyen. Ongelma vaikuttaisi olevan, että parametrinä menevä kansiopolku ei sisällä oikeanlaista merkistöä, kun se sisältää skandinaavisia kirjaimia. Aikaisemmin olen ollut tekemisissä merkistöjen kanssa web-ohjelmoinnissa, mutta ohjelmistopuolella tämä oli täysin uutta.

Löysin pian funktion, jolla voisin kääntää tiedostonnimimuuttujan merkistön oikeaan muotoon käyttäen UTF-8 -merkistöä. Lisäsin tämän kääntäjäfunktion string-muuttujalle juuri ennen fbx-tiedoston luomista. Tämä kääntäminen näkyy kuviossa 27. UTF-8 on minulle jotenkin tuttu myös web-ohjelmointipuolelta, ja tiedän, että sen pitäisi sisältää skandinaavisia kirjaimia. Fbx-tiedoston luominen alkoi toimimaan tämän jälkeen. Seuraavaksi ongelma oli uip-tiedoston luomisessa.

```
#Export FBX to Project Root folder
 def exportFbx(self, *args):
     self.checkObjs()
     exportFile = self.projectPath+"/"+self.sceneName.getText()
     print exportFile
     #Export settings
     pm.mel.FBXExportHardEdges(v=True)
     pm.mel.FBXExportTangents(v=True)
     pm.mel.FBXExportReferencedAssetsContent(v=True)
     pm.mel.FBXExportColladaTriangulate(True)
     #TODO CHECK AXISCONVERSIONMETHOD CORRECTLY
     #pm.mel.FBXExportAxisConversionMethod("none")
     #pm.mel.FBXExportInAscii(v=True)
     pm.mel.FBXExportFileVersion("FBX201600")
     "Freeze Transforms for mesh objects"
     cmds.makeIdentity(cmds.ls(sl=True),apply=True,translate=True,rotate=True,scale=True,normal=False,pn=True)
     #CONVERT PATH INPUT TO System encoding
     finalExport = exportFile.encode('utf-8')
     pm.mel.FBXExport(s=True, f=finalExport)
     """Make uip.file"""
     path = self.projectPath+"/"
     exists= os.path.isfile(path+self.sceneName.getText())
     reload(uipmaker)
     #CONVERT FILENAME TO UTF-8
     #nameValue = self.sceneName.getText()
     #uipName = nameValue.encode('utf-8')
     uipmaker.launchMaker(path,self.sceneName.getText())
     cmds.select(cl=True)
```

KUVIO 27. Fbx-tiedoston nimimuuttujan kääntäminen

Uip-tiedostosta vastaa uipmaker moduulissa sain samaa vikailmoitusta, ja aloin lisäämään merkistön muuttamista niihin kohtiin, jossa ohjelma ei enää edennyt. Pian olin jo kohdassa, jossa uip-tiedoston kirjoittaminen pitäisi tapahtua, mutta kirjoitusfunktio ei hyväksynyt string-muuttujaa usean eri kääntämisyrityksen jälkeen. Verkon keskustelufoorumeilta löysin henkilön, jolla oli ollut samanlainen ongelma. Hänellä ratkaisu oli lisätä koodiin codecs-niminen moduuli ja korvata tiedoston luovan open-funktion codecs.open-funktiolla, jonne pystyy lisäämään parametrinä halutun merkistön. Tämä koodi näkyy kuviossa 28. Uip-tiedoston luominen onnistui tämän jälkeen.

```
def write_file():
    global destinationFolder
    global systemEncoding

    print type(destinationFolder)
    print destinationFolder

    #xml_file = open(os.path.expanduser(destinationFolder), "w")
    xml_file = codecs.open(os.path.expanduser(destinationFolder), "w", encoding='utf-8')
    xml_file.write(doc.toprettyxml())
    xml_file.close()
```

KUVIO 28. Uip-tiedoston kirjoittamisen merkistön korjaus

Olin tyytyväinen, että työkalu pystyi nyt käsittelemään skandinaavisia kirjaimia sekä tiedostopolussa että luotavien tiedostojen nimissä. Testasin koodia Windows-tietokoneella ja sain taas saman vikailmoituksen. Selvitin asiaa, ja pyysin myös työkaveriani vilkaisemaan vikaa, sillä hän on Linux-järjestelmämies ja tuntee varmasti paremmin merkistöt sekä niiden käyttämisen.

Aluksi löysimme Windows-käyttöjärjestelmälle sopivan merkistön, ja ajattelin, että työkalusta pitää luoda omat versiot Mac- sekä Windows-käyttöjärjestelmille. Työkaverini kuitenkin löysi myös Pythonin Sys-järjestelmiin erikoistuneesta moduulista toiminnon, joka pystyy haistelemaan tietokoneen järjestelmälle sopivan merkistön. Teen tästä toiminnosta muuttujan, jonka lisään samoihin paikkoihin, joissa käytin utf-8 merkistöä, ja koodi toimii molemmilla järjestelmillä samalla koodilla. Kuvioissa 29 ja 30 näkyvät päivittämäni muutokset merkistöjen muuttamiselle uudella muuttujalla.

```
def exportFbx(self, *args):
    self.checkObjs()
    exportFile = self.projectPath+"/"+self.sceneName.getText()
    print exportFile
    #Export settings
    pm.mel.FBXExportHardEdges(v=True)
    pm.mel.FBXExportTangents(v=True)
    pm.mel.FBXExportReferencedAssetsContent(v=True)
    pm.mel.FBXExportColladaTriangulate(True)
    #TODO CHECK AXISCONVERSIONMETHOD CORRECTLY
    #pm.mel.FBXExportAxisConversionMethod("none")
    #pm.mel.FBXExportInAscii(v=True)
    pm.mel.FBXExportFileVersion("FBX201600")
    "Freeze Transforms for mesh objects"
    cmds.makeIdentity(cmds.ls(sl=True),apply=True,translate=True,rotate=True,scale=True,normal=False,pn=True)
    #CONVERT PATH INPUT TO System encoding
    finalExport = exportFile.encode(self.systemEncoding)
    pm.mel.FBXExport(s=True, f=finalExport)
    """Make uip.file"""
    path = self.projectPath+"/"
    exists= os.path.isfile(path+self.sceneName.getText())
    reload(uipmaker)
    #CONVERT FILENAME TO UTF-8
    #nameValue = self.sceneName.getText()
    #uipName = nameValue.encode('utf-8')
    uipmaker.launchMaker(path,self.sceneName.getText())
    cmds.select(cl=True)
```

KUVIO 29. Qtwindow oliokoodin merkistön päivitetty muuttuja

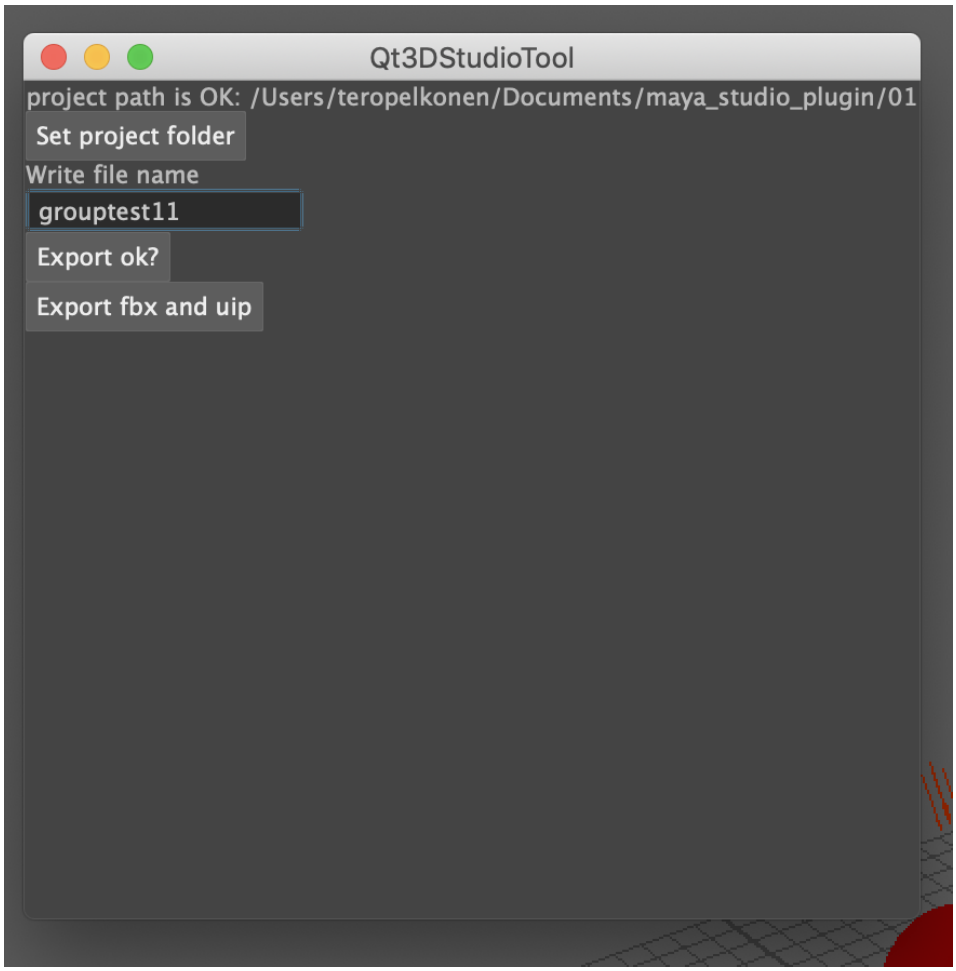
```
def write_file():
    global destinationFolder
    global systemEncoding

    print type(destinationFolder)
    print destinationFolder

    #xml_file = open(os.path.expanduser(destinationFolder), "w")
    xml_file = codecs.open(os.path.expanduser(destinationFolder), "w", encoding=systemEncoding)
    xml_file.write(doc.toprettyxml())
    xml_file.close()
```

KUVIO 30. Uipmaker proseduraalisen koodin merkistön päivitetty muuttuja

Tänään käsittelin paljon uutta tietoa ja sain korjattua ongelmat, joihin törmäsin. En ymmärtänyt kaikkea lukemaani, mutta tunnistan varmasti tulevat merkistöihin liittyvät ongelmat nyt paremmin.

Torstai 13.6.2019

Aamupäivästä testasin vielä edellisenä päivänä tehtyä työtä kokeilemalla eri töiden tallennuksia erilaisilla skandinaavisilla merkistöillä, ja en saanut vikailmoituksia. Päätin siirtyä seuraavaan aiheeseen, joka oli käyttöliittymä. Työkalussa oli ollut todella yksinkertainen käyttöliittymä, jollaista ei voida mahdollisilla asiakkailla käyttää, joten olin pyytänyt tiimimme käyttöliittymäsuunnittelijaa tekemään minulle alustavan mallin siitä, miltä se voisi näyttää. Kuviossa 31 näkyy alkuperäinen versio käyttöliittymästä.



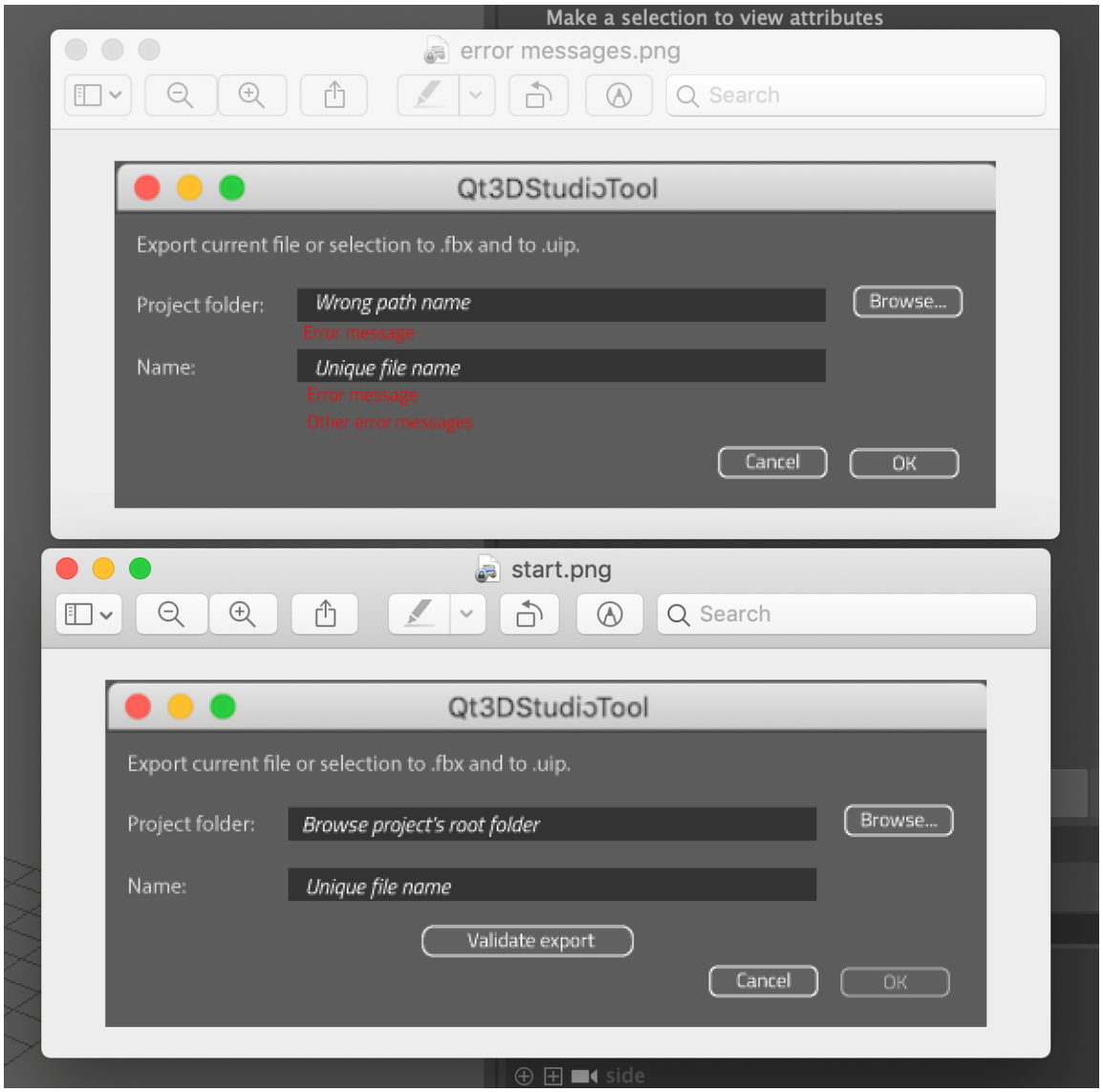
KUVIO 31. Alkuperäinen käyttöliittymä

Hyödynnän käyttöliittymän tekemiseen Mayan Commands-rajapintaa, joka sisältää useita tähän liittyviä komentoja kuten Windows, jolla voidaan tehdä ikkuna, sekä ColumnLayout ja RowLayout, joilla voidaan ohjata käyttöliittymään tulevien elementtien asettelua (engl. Layout). Nämä elementit

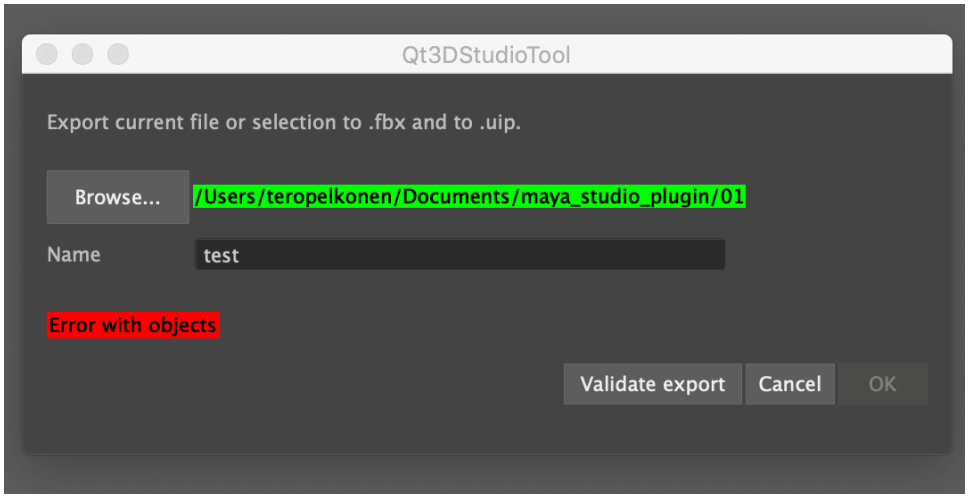
ovat tyypillisesti tekstiä, tekstinsyöttökenttiä, nappeja, listoja ja muita hyvin tyypillisiä käyttöliittymäsisältöjä.

Olen ollut tekemisissä jonkin verran verkko-ohjelmoinnissa käyttöliittymien tekemisen kanssa, ja siellä tätä työskentelyä on kehitetty mielestäni todella paljon. Mayalla käyttöliittymän teko on huomattavasti tönkömmän oloista, sillä elementtien ominaisuuksien säädöt kuten sisennykset, välit, värit ja muut ulkonäköön vaikuttavat asiat ovat joko poissa tai ne joudutaan tekemään soveltamalla muita tapoja.

Sain kuitenkin tehtyä suunnittelijamme tekemän käyttöliittymän, ja koin erään kentän siinä haitalliseksi, joten esitin tehdyn työn hänelle uudestaan, ja suunnittelimme yhdessä hieman muunnellun käyttöliittymän. Hän huomasi esimerkiksi, että nappi joka avaa uuden ikkunan pitää sisältää kolme pistettä tekstin lopussa. Kuviossa 32 näkyy hänen suunnittelemansa käyttöliittymät ja kuviossa 33 tekemäni versio.



KUVIO 32. Käyttöliittymäsuunnittelijan näkemyskäyttöliittymästä



KUVIO 33. Tekemäni yksinkertainen käyttöliittymä

Vanhan toiminnallisuuden siirtäminen uuteen käyttöliittymään sujui aika ongelmitta. Työpäivän aikana en saanut valmiiksi työkalun vahvistusviestiä, kun työkalua oli onnistuneesti käytetty. Aioin viimeistellä tämän seuraavana päivänä.

Perjantai 14.6.2019

Päivän tavoitteeksi asetin keskittymisen käyttöliittymään ja työstin ensimmäisenä siihen liittyvää koodia. Tarkistin edellispäivänä kesken jääneen vahvistusviestin luomisen. Tässä kohdassa koodia käytin ensimmäistä kertaa itse tehtyjen moduulien välistä palautusarvoa. Window-olion luodessa fbx-tiedoston se komentaa uipmaker moduulia luomaan uip-tiedoston annetuilla parametreillä. Lisäsin koodiin pätkän, joka palauttaa totuusarvon window-oliolle, kun uip-tiedosto on luotu. Olio tuo näytille onnistumisilmoituksen, jos tuo arvo on totta. Jos uipmaker-koodissa tapahtuu virhe, se palauttaa epätosiarvon, ja toisenlainen viesti tulee näkyviin.

Seuraavaksi testasin vielä työkalua, ja päätin kokeilla erilaisten tiedostonimien luomista. Huomasin, että lisäämäni nimentarkistusfunktiot eivät toimineet kuten piti, vaan ne päästivät mahdollisesti haitallisia merkkejä läpi. Etsin tapoja lisätä tarkistuksia string-muuttujiin, mutta suurin osa esimerkeistä muutti stringit täysin Ascii-merkistöön, jossa ei hyväksytä skandinaavisia kirjaimia.

Päädyin selvittämään tätä ongelmaa suurimman osan päivästä. Ratkaisuni oli funktio, jossa string-muuttujan jokainen merkki tarkistetaan. Koodi tarkistaa, onko merkki Ascii-kirjain, numero tai jokin merkki listasta, johon on määritelty hyväksytyt merkit. Tämä ratkaisu toimi ainakin kyseisellä hetkellä halutulla tavalla.

Tekstisyötön tarkistuskoodin muutos aiheutti uusia merkistöongelmia, ja selvitin niitä lopun työaikaa. Vika vaikutti johtuvan siitä, että koodinmuutos toi uusia string-muuttujia koodiin, ja osa niistä oli väärässä merkistömuodossa. Tein muuttujiin merkistömuutokset, ja koodi alkoi taas toimimaan ennen kuin lähdin töistä.

Viikkoanalyysi

Tällä viikolla jouduin ensimmäistä kertaa ohjelmoijana miettimään merkistön käyttämistä kunnolla. Kouluopinnoissani on keskitytty verkkopuolen ohjelmointiin, ja siellä merkistön käytöllä on jo tukevat standardit etenkin utf-8 käytössä. Vaikuttaa siltä, että ohjelmistopuolella on yleisempää, esimerkiksi rajoittaa käyttäjiä merkkien käytössä kuin muuttaa koodia sellaiseksi, että se hyväksyy muita kuin ascii-merkkejä. Maya 2019 ei esimerkiksi hyväksy ääkkösiä objektien nimiin, vaikka jotkin sen aikaisemmat versiot hyväksyvät.

Hyvä merkistön käsittelytapa on tietää, mitä merkistöä mikäkin teksti käyttää, ja se teksti tulkitaan sillä merkistöllä. Jos tehdään ohjelmaa, jossa käyttäjältä otetaan syöte, on määriteltävä, mitä merkistöä käyttäjältä hyväksytään. Jos merkistöä pitää kääntää toiseen merkistöön, se on tehtävä puhtaasti sitä varten olevilla työkaluilla. On pyrittävä käyttämään mahdollisimman paljon Unicodea. (Zentgraf 2015, Viitattu 16.7.2019.)

Työkalun toimivuuteen lisätyt käyttöliittymäparannukset olivat myös asia, johon en ollut keskittynyt aikaisemmin. Nyt mielestäni minulla on parempi käsitys, kuinka Pythonilla ja Maya Commands-rajapinnalla saadaan kirjoitettua halutunlaista käyttöliittymää.

Mielestäni oli erikoista, että törmäsin merkistöongelmiin vasta kuudennella viikolla enkä aikaisemmin. Tämä saattaa johtua siitä, että yleensä vältän ääkkösten käyttöä tietokoneympäristöissä. Kun työkalu menee asiakkaille, olen aivan varma, että joudun palaamaan merkistöasian selvittämiseen.

3.7 Seitsemäs viikko: UV-kartoitus ja käyttöliittymäparannukset

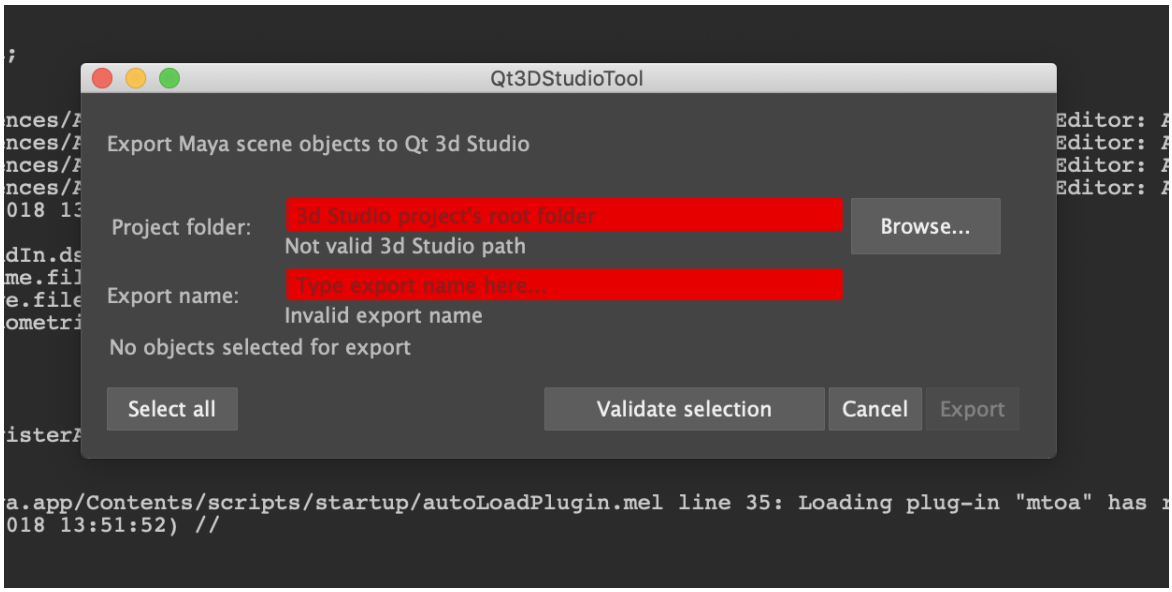
Maanantai 17.6.2019

Viime viikolla olin saanut käyttöliittymää kehitettyä oikeaan suuntaan, ja tänä päivänä aioin keskittyä sen parantamiseen. Aamulla törmäsin esimieheeni ja pyysin häntä vilkaisemaan senhetkistä työtilannetta. Sain häneltä seuraavat kehittämistehtävät, jotka olivat kansiopolun manuaalisen kirjoittamisen mahdollistaminen, vikatilanneilmoituksien siirtäminen syötekenttiin tai niiden läheisyyteen sekä käyttökieltä pitää parantaa.

Käyttökielen parantaminen oli helpoin tehtävä. Vaihdoin Ok-napin Export-napiksi, joka kuvaa paremmin napin viemistoimintoa. Lisäsin myös Browse…-napin riville Select folder:-tekstin, joka kertoo mitä pitää etsiä.

Manuaalisen kansiopolun kirjoittaminen oli hieman haasteellisempi tehtävä, sillä tehty koodi oli suunniteltu pelkälle Browse…-napin toiminnoille, ja se palauttaa aina oikean muotoisen kansiopolun. Käsin kirjoitettu polku voi mennä niin pahasti pieleen, että koodissa olevan Os moduulin käyttöjärjestelmään liittyvät toiminnot lakkaavat toimimasta oikein. Näin kävikin, ja jouduin lisäämään ensimmäisen kerran koodiin Try Except-toiminnon. Tämä toiminto tarkoittaa yksinkertaisimmillaan kahta kokeilua. Try-toiminto yrittää sen sisälle kirjoitettua koodia, jos tämä ei onnistu niin se suorittaa Except-toiminnon sisällä olevan koodin. Tuohon toimintoon voisi lisätä vielä muita variaatioita eri tapauksille, mutten kokenut tarvitsevani sellaista tällä hetkellä.

Vielä oli jäljellä vikailmoituksien siirtäminen syötekenttiin. Loin tarvittavat päivitysfunktiot, ja lisäsin taustavärin muuttumisen kenttiin tilanteen mukaisesti. Jos kaikki oli hyvin, niin kenttä oli vihreä ja jos jotain oli pielessä, niin kenttä oli punainen. Mieleeni tuli myös, että värisokeat käyttäjät eivät voisi käyttää tällaista, joten lisäsin kenttien alle vielä rivit, joihin tilanteesta tulee lisätietoa. Kuviossa 34 näkyy kentän värin muuttuminen sekä tämänhetkinen ulkoasu.

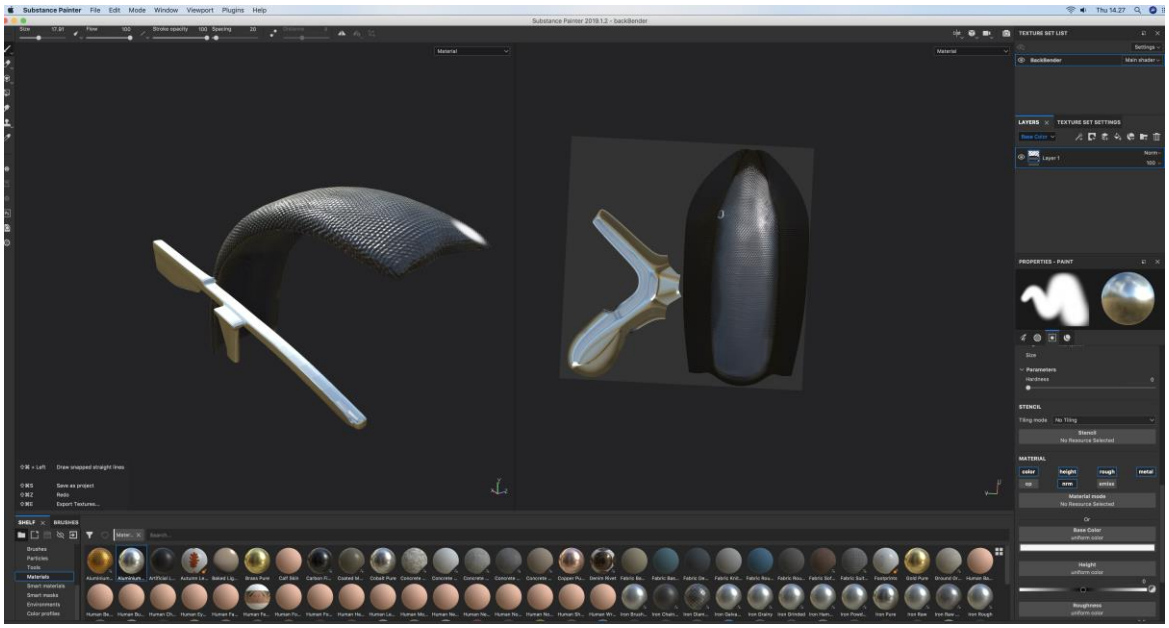


KUVIO 34. Värisokeaystävällisempi käyttöliittymä

Tiistai 18.6.2019

Tälle päivää en ollut asettanut itselleni mitään tavoitteita, mutta törmäsin aamulla Buddyyn ja päätimme käydä läpi työkalun tätä kehitystilannetta. Eniten minua mietitytti Freeze Transforms-toiminnon käyttö, joka nollaa polygonimallien sijainti-, rotaatio- sekä skaalatiedot. Buddylla oli idea, mistä polygonien väärä asettelu voisi johtua. Eri 3D-mallinnusohjelmissa on käytössä eri tavalla toimivat ulottuvuuskoordinaatistot. Joissakin ohjelmissa kasvava Y-arvo voi viedä polygonia käyttäjästä kauemmas ja toisessa ohjelmassa tuoda lähemmäs. Sovimme, että poistan Freeze Transforms-toiminnon koodista ja ohjelmoin sijaintien Y-arvot kertomalla ne miinus yhdellä. Näin niistä saa heti 3D Studiota vastaavat. Tämä ratkaisu toimi oikein, ja toinen parannus oli, että objektit eivät enää menettäneet tietojaan.

Lopun päivää opiskelin 3D-mallinnuksen teksturointia varten minulle hankittua Substance Painter-ohjelmaa. Toin ohjelmaan Blenderillä tekemiäni moottoripyöränosia, ja tein niihin yksinkertaiset tekstuurit. Sain valmiiksi etu- ja takarenkaihin liittyvien osien värilliset tekstuurit, joihin liitin myös normaalikartan. Kuviossa 35 näkyy työkalun työnäkymä.



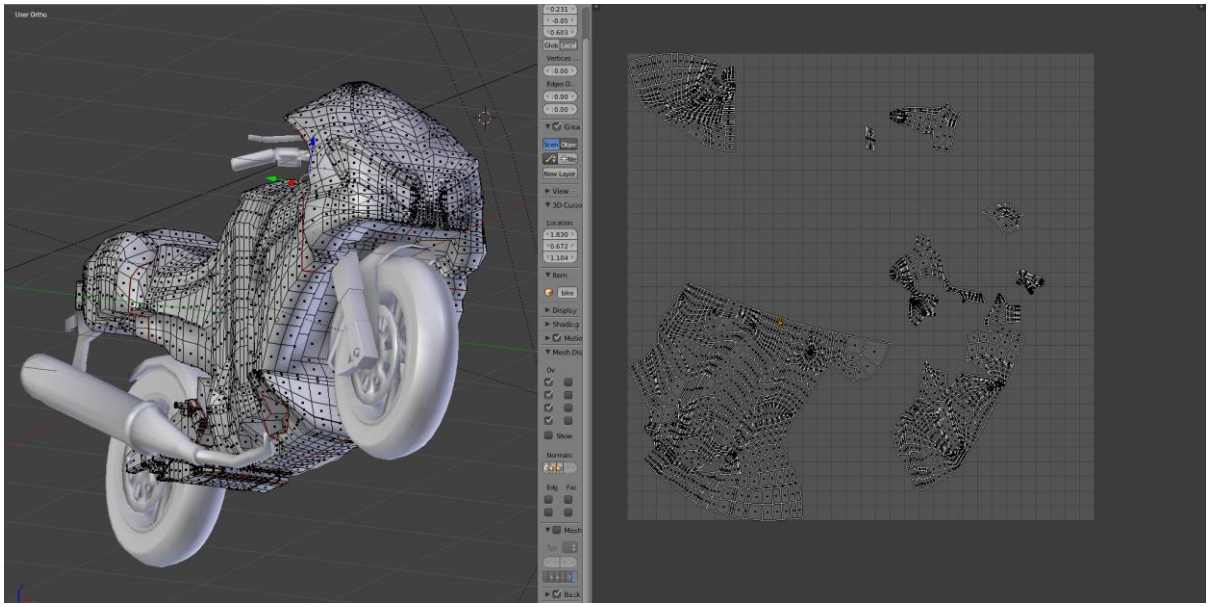
KUVIO 35. Substance Painter

Keskiviikko 19.6.2019

Edellisenä päivänä minulla ei ollut suurempia suunnitelmia työkalun kehittämistä varten, vaan keskityin enemmän 3D-työskentelytaitojen kehittämiseen. Tälle päivää olin suunnitellut samanlaista.

Kello kahdeltatoista tiimillämme oli tiimikokous, jossa kerroimme taas työtilanteemme. Sain omalla vuorollani vielä yhden idean, mitä voisin lisätä työkaluun. Työkalun käyttämisessä oleellista on, että käyttäjä valitsee objektien hierarkioiden ylimmät solmut, jotka ovat muuttujasolmuja (engl. transform node). Jos käyttäjä valitsee yhden portaan alempaa hierarkiasta olevan solmun, niin se on yleensä muotosolmu (engl. shape node), joka sisältää vääräntyyppistä tietoa, ja työkalu ei toimi oikein. Lisäsin työkalun koodiin toimintoja, jotka poistavat joko näiden muoto solmujen valinnat tai käyttäjälle tulee viesti käyttöliittymään vääränlaisesta valinnasta.

Loppupäivän työstin tekemäni moottoripyörän 3D-mallin runkoa ja aloitin tekstuurien piirtämisen siihen Substance Painter-ohjelmalla. Runkoa UV-kartoittaessa huomasin, että sen polygonimallissa on useita paikkoja, joita en ollut osannut ajatella teksturoinnin kannalta. Kuviossa 36 näkyy UV-kartoituksessa käytettyjä saumoja sekä levitetty tekstuuri.



KUVIO 36. Moottoripyörä mallin UV-kartoitus

En ole UV-kartoittanut vähään aikaan, ja oli todella mielenkiintoista tehdä sitä pitkästä aikaa. UV-kartoituksessa luotu 3D-malli avataan saumoilla kaksiulotteiseksi. UV:lla tarkoitetaan horisontaalista ulottuvuutta U sekä vertikaalia ulottuvuutta V. Näillä ulottuvuuksilla on arvot nollasta yhteen. Kun molemmat akselit ovat nollissa, niin koordinaatti piste on vasemmassa alakulmassa. Kun arvot olisivat molemmat yhden, niin koordinaatti piste olisi oikeassa yläkulmassa. Avattu 3D-malli yritetään saada aukaistua tasaisesti tälle koordinaatistolle, jotta sille voidaan määrittää esimerkiksi tekstuuri. Huonosti UV-kartoitetussa mallissa tekstuuri venähtää huonon näköiseksi.

Torstai 20.6.2019

Tämän päivän tavoitteeksi asetin sen, että saisin moottoripyörän kaikki osat UV-kartoitettua sekä tehtyä niille tekstuurit Substance Painter-ohjelmalla. Olin jo saanut ison osan moottoripyörämallin osista teksturoitua, mutta uskoin, että loppujen osien viimeistely veisi koko päivän, ja niin se veikin.

Kävin aamupäivästä keskustelemassa esimieheni kanssa työtehtävä tilanteesta, ja kerroin että tarkoitukseni on luoda tämä malli tekstuureineen ja viedä se Mayaan. Sieltä oli tarkoitus viedä sen tekemälläni työkalulla 3D Studioon, jolloin voisin löytää mahdollisia vikoja tekemästäni koodista.

Aamupäivästä otin hetken opiskelua varten ja luin verkosta kirjoituksia 3D-mallintamisen eri vaiheisiin liittyen. Teksteistä kävi ilmi, että eri 3D-tuotannoissa työtekijät erikoistuvat yleensä yhteen tai useampaan työvaiheeseen. Minua kiinnostaa osata hieman kaikkea, ja useaan eri ohjelmaan tutustuminen olisi myös mukavaa.

Sain moottoripyörän kaikki osat teksturoitua, ja olin lopputulokseen ihan tyytyväinen, sillä mallinnus- ja teksturointityöskentelystä minulla on ollut opiskeluiden aikana pitkä tauko. Seuraavana päivänä on juhannus, ja pääsisin seuraavalla viikolla siirtämään työn Mayaan.

Viikkoanalyysi

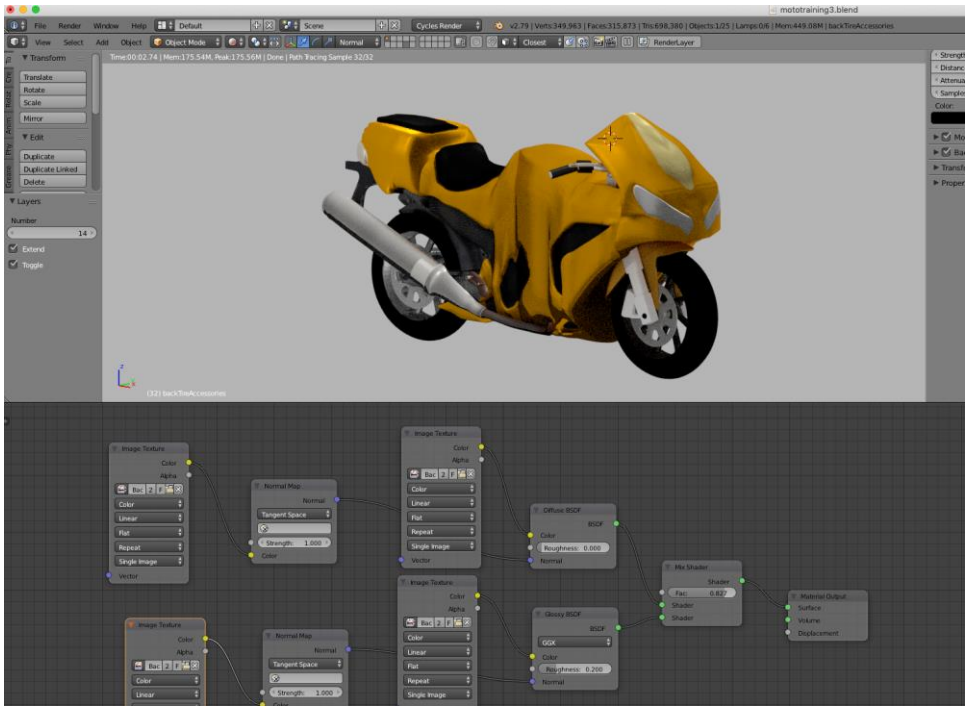
Tämä viikon teemoja olivat käyttöliittymän kehittäminen Python-koodauksen puolella ja UV-kartoitus 3D-mallintamisessa. 3D-työskentely oli todella mukavaa vaihtelua Python-koodin tekemiseen. Samalla tulin ajatelleeksi tekemäni viemistyökalun käyttöä hieman mallintajan näkökulmasta, esimerkiksi sitä, kuinka käyttäjä valitsee haluamansa objektit vietäväksi. Tekemälläni moottoripyörämallin esityksessä oli jossakin työvaiheessa useampia eri objekteja piilotettuna. Tämä on sen vuoksi, että mallintaja saattaa tarvita jotain objektia myöhemmin, mutta piilottaa ne toistaiseksi. Mielestäni viemiseen voisi lisätä valinnan: ”Vie kaikki näkyvät”.

Python-koodauksessa paras hoksaus oli huomata, että eri 3D-ohjelmissa X-, Y-, ja Z-akselien arvot avaruudessa voivat olla ohjelmakohtaisesti täysin erit. Tästä johtui ilmeisesti myös aikaisemmin 3D Studioon tuotujen esityksien peilikuvaefekti. Mayassa ja 3D Studiossa Z-akselin positiiviset arvot vievät eri suuntiin.

Kuvioissa 37, 38 ja 39 näkyy tekemäni malli tekstuureineen. Olen ihan tyytyväinen tekemääni malliin, sillä minulla oli ollut pitkä tauko 3D-mallintamisesta. Tämä malli oli myös sopivan haastava muodoiltaan.

KUVIO 37. Moottoripyörä piirustus + malliupotettuna

KUVIO 38. Moottoripyörämalli sivusta



KUVIO 39. Moottoripyörämalli sekä Blender-ohjelman materiaalijärjestelmä

3.8 Kahdeksas viikko: Työkalun päivittämistä ja testaamista

Maanantai 24.6.2019

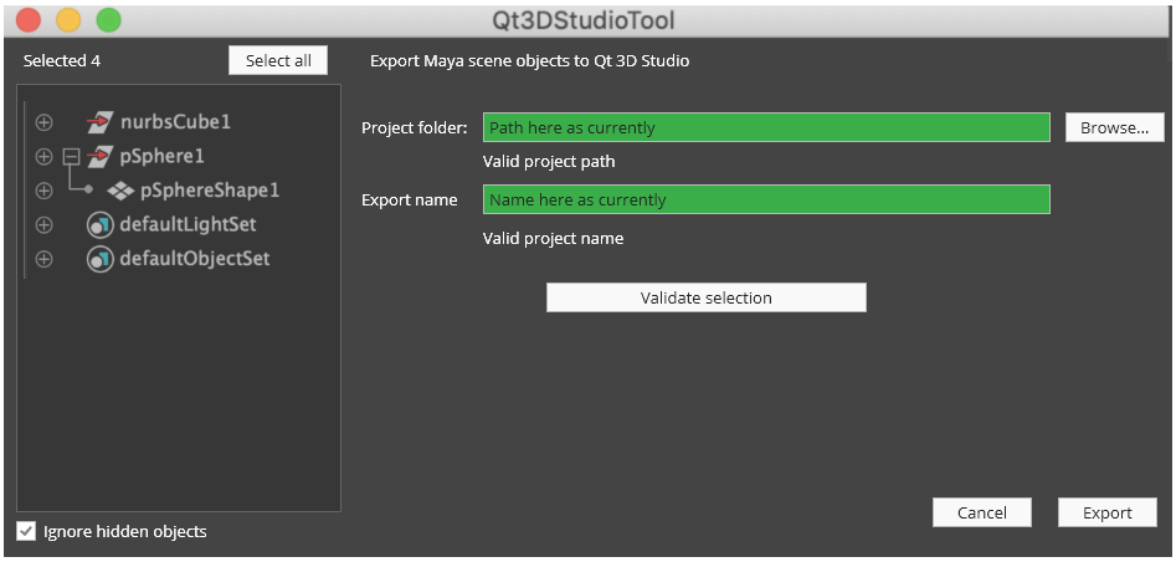
Viime viikolla olin keskittynyt paljon 3D-mallintamiseen, ja tänä päivänä halusin kehittää Maya-työkalun koodia. Latasin verkosta muutamia monimutkaisempia malleja ja siirsin ne Maya-esitykseen. Siirsin myös Blender-ohjelmalla tekemäni moottoripyörämallin ja aloitin työkalun testaamisen. Asettelin mallit eri järjestyksiin ja siirsin ne työkalulla 3D Studioon. Osassa malleista huomasin, että tiedoston siirtäminen kesti jopa minuutteja, mikä pitää ottaa huomioon käyttöliittymää suunniteltaessa. Käyttäjä voi luulla, että ohjelma on kaatunut, jos hän ei saa minkäänlaista ilmoitusta työkalun eri vaiheista.

Toinen työkalun käyttöön vaikuttava tärkeä huomio oli Validate Selection-napin vääränlainen toiminen. Sen pitäisi tarkistaa valitut objektit ja lukita ne listaan objekteista, jotka viedään ulos Mayasta. Ennen korjausta se tarkisti valitut objektit ja aktivoi viemisnapin, mutta valitut objektit eivät jääneet mihinkään muistiin. Tästä johtuen käyttäjä voisi valita tarkistuksen jälkeen muita objekteja, jotka eivät läpäisisi tarkistusta ja työkalu voisi kaatua. Korjasin tilanteen tekemällä listan, johon lisätään objekteja vain painamalla Validate Selection-nappia, ja pelkästään tämän listan objektit viedään ulos Mayasta työkalulla.

Lopun päivää keskityin käyttöliittymän parantamiseen. Tein yhteistyötä tiimimme käyttöliittymäasiantuntijan kanssa, ja sain häneltä uuden käyttöliittymän suunnitelman.

Tiistai 25.6.2019

Tänään tavoitteeni oli saada luotua käyttöliittymään uudet ominaisuudet, jotka työkaverini oli minulle hahmotellut. Käyttöliittymän ohjelmoinnin lisäksi minun piti ohjelmoida uutta logiikkaa piilotettujen objektien valitsemiseen. Merkittävin päivitys uudessa käyttöliittymäpohjassa on tekstikenttä, jonne ilmestyy rivi riviltä valituiksi hyväksytyt objektit. Kuvioissa 40, 41 ja 42 näkyvät uudet käyttöliittymäsuunnitelmat.



KUVIO 40. Käyttöliittymän aloitustilanne

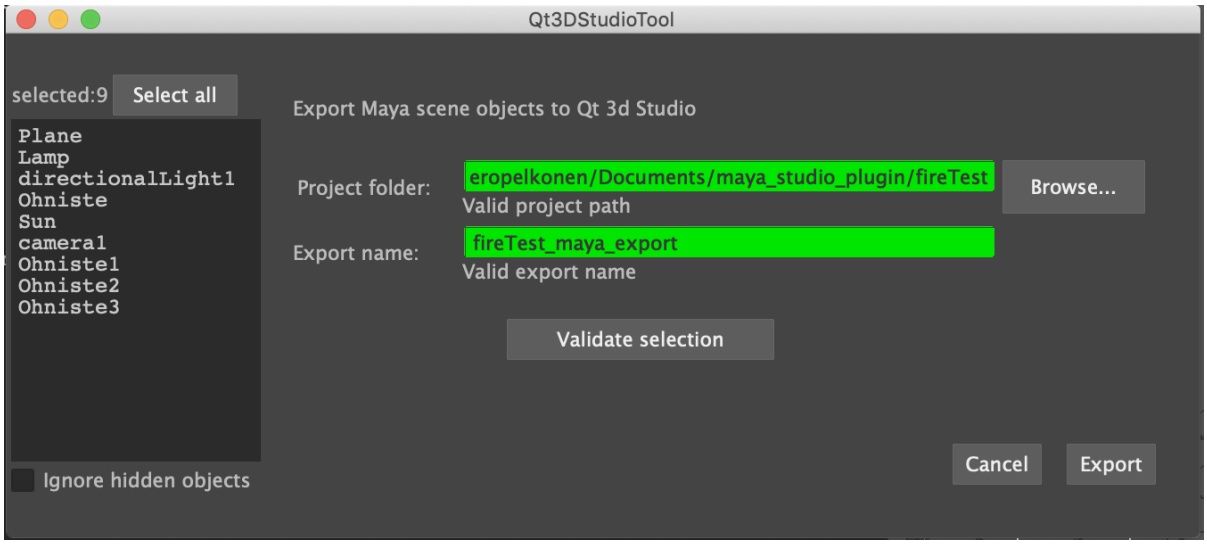
KUVIO 41. Käyttöliittymän vientitilanne

KUVIO 42. Käyttöliittymän vikatilanne

Kopioin nykyisen käyttöliittymän ulkoasuun liittyvän koodiosion erilliselle skriptisivulle, ja muokkasin sen proseduraaliseen muotoon. Tällä tavoin pystyin kutsumaan koodia nopeasti ja tekemään siihen pieniä muutoksia kerrallaan. Sain käyttöliittymäkoodista haluamani näköisen, ja aloin kopioimaan sen runkoa itse pääkoodiin. Hyvin nopeasti ulkoasu oli mielestäni kunnossa.

Seuraavaksi mietin, kuinka objektit saataisiin siirrettyä uuteen tekstikenttään. Verkko-ohjelmoinnissa olen ollut tekemissä useiden listojen kanssa ja tätä kokemusta pystyin hyödyntämään tekstikentän kanssa. Sain luotua kentän niin, että se näyttää usealta eri tekstiosalta, vaikka se on oikeasti yksi pitkä tekstipätkä, joka on paloiteltu rivivaihdoilla.

Päivän viimeinen haaste oli lisätä logiikka piilotettujen objektien osalta. Uudessa käyttöliittymässä on valintalaatikko (engl. checkbox), jolla voidaan ohjata, halutaanko vientiin valita myös piilotetut objektit. Tämän toiminnallisuuden lisääminen tarkoitti ikkunaskriptissä kahden uuden funktion luomista sekä muutaman edellisen muokkaamista. Tein myös muutoksia uip-tiedoston luomiskoodiin, jotta viedyt piilotetut objektit näkyisivät myös piilotettuina 3D Studiossa. Kuviossa 43 näkyy päivittämäni käyttöliittymä.



KUVIO 43. Päivitetty käyttöliittymä

Keskiviikko 26.6.2019

Esitin tällä viikolla tehtyjä muutoksia esimiehelleni sekä käyttöliittymävastaavalle. Sain heiltä uusia ideoita, joiden pohjalta kehitin uuden viestikentän, joka ilmoittaa kaikki tilanteeseen liittyvät viestit sekä ohjekentän, jossa mainitaan viennissä sallitut objektit.

Näiden kenttien luominen oli tänään todella helppoa, sillä minusta tuntuu, että olen oppinut hyviä käytäntöjä Mayan käyttöliittymäkielen kanssa. Kenttien luomisen jälkeen isompi ongelma oli, että kokonaisuus ei enää tuntunut toimivan hyvin ja jouduin siirtelemään eri kenttiä loogisesti järkevämpään järjestykseen. Tällaisia muutoksia oli esimerkiksi siirtää ohjeet lähemmäksi ikkunan alkua ja viestit lähemmäksi toimintanappeja.

Käyttöliittymän jälkeen testasin monimutkaisemman esityksen ulosvientiä työkalulla ja otin siitä aikaa. Kesti kolme minuuttia, että tiedostot olivat luotuja ja valmista-viesti ilmestyi työkaluun. Kokeilin fbx-tiedoston ulosvientiä samasta esityksestä Mayan omalla vientitoiminnolla, ja se oli valmis välittömästi. Selvitin asiaa, ja ongelma ratkesi sillä, että ulkoistin tiedoston viemisen erilliselle moduulille. Kolme minuuttia lyheni 15 sekuntiin tällä muutoksella ja olin siihen tyytyväinen.

Torstai 27.6.2019

Työkalu alkaa tuntua valmiilta, ja tänään oli hankala keksiä, miten sitä voisi kehittää paremmaksi. Yksi merkittävä asia käyttäjien kannalta on työkalun itsetekemä suodatus vietäville objekteille. Päätin keskittyä tuohon.

Ensimmäisenä otin tarkistukseen Nurbs-objektit, koska uskoin, että ne ovat ehkä eri valolähteiden jälkeen yleisin objektimuoto, jota käyttäjät voisivat yrittää viedä Qt 3D Studioon. Tarkistin tekemästäni koodista pätkiä, joissa käydään läpi eri objektityyppien tarkistamista. Huomasin, että kun olin keskittynyt muihin osa-alueisiin, niin tiettyjen asioiden tekeminen koodissa vaati taas totuttelua. Sain kuitenkin nopeasti luotua funktiot, joilla voidaan tarkistaa, onko valittu objekti Nurbs-tyyppinen. Nurbs-objekti on vektorikurveja käyttävä objekti, jollaista 3D Studio ei tällä hetkellä tue.

Laajensin tätä funktiota koskemaan kaikkia muita objektityyppejä kuin mitkä ovat sallittuja. Tämän koodipätkä lisääminen ei sujunut niin yksinkertaisesti, sillä objektien iteroinnissa tapahtui ongelmia. Vika vaikutti johtuvan objektien solmujen käsittelystä. Joudun jatkamaan tämän parissa huomenna.

Tiimissämme juhlittiin tänään uuden 3D Studion uuden version julkaisua, ja kävimme yhdessä syömässä.

Perjantai 28.6.2019

Tänä päivänä jatkoin edellispäivänä aloitettua eri objektien suodattamista sekä testaamista. Käytin tähän koko päivän, enkä saanut silti valmista. Edellispäivänä minulla oli ongelmia eri objektityyppien iteroinnissa, mutta sain tämän korjattua heti aamusta.

Uusia ongelmatilanteita koodissa alkoi ilmetä, kun testasin erilaisia hyväksyttyjä objektimuotoja, mutta ne sisälsivät ryhmähierarkioita ja osa objekteista oli piilotettuja. Selvisi, että koodi ei pystynyt käsittelemään näitä tilanteita, joten loin yksi kerrallaan erilaisen ongelman ja aloitin sen selvittämisen koodissa.
Tämä testaus oli pitkästä aikaa raskasta ajatustyötä, sillä näin pitkälle viedyssä koodissa yhden asian muuttaminen voi vaikuttaa useaan toimintoon. Tein pieniä muutoksia, jotka testasin heti.

Samalla huomasin, että minun kannattaa parantaa koodin konsoliin kirjoittamia viestejä, joilla voi tulkita paremmin ongelmatilanteita.

Olin kuitenkin tyytyväinen, että löysin itse nämä ongelmakohdat koodista, ja pääsin työstämään niitä heti, ettei käynyt niin, että työkalu menee mahdolliselle asiakkaalle käyttöön ja kuulen ongelmista vasta heiltä.

Viikkoanalyysi

Tällä viikolla tein paljon testaustyötä sekä kehitin käyttöliittymää Maya Commands moduulia käyttäen. Käyttöliittymän muokkaamisessa hienoa oli huomata, kuinka paljon oma käsitys siitä on parantunut. Myös uusiin käyttöliittymäelementteihin tutustuminen opetti paljon. Testaaminen oli välillä todella haastavaa, ja saatoin uppoutua tietyn ongelman selvittämiseen tunneiksi.

Lähes kaikki löytämäni ongelmat olivat sellaisia, joita en osannut odottaa. Tärkein työkalu ongelman selvittämisessä oli ongelmakohdan löytäminen tulkitsemalla koodia ja jättämällä tarvittavia ilmoituksia konsoliin. Tällä tavoin sain selvitettyä hyvin nopeasti, jos jotain arvoa koitettiin esimerkiksi iteroida listana, vaikka se on tyhjä. Selvitystyötä auttoi myös pelkkä hyvän tauon pitäminen sekä ulkona käyminen.

Ratkaisuni ongelmiin oli lisätä konsoliin ilmestyvien viestien määrää ja parantaa niiden laatua. Tämä auttaa myös tulevaisuudessa, sillä näen selkeämmin, missä kohtaa koodia ohjelma kaatuu.

Testaaminen voi olla likaista. Sillä ei ole selkeää alkua, keskivaihetta tai loppua. Testauksessa joudutaan ottamaan usea riskitekijä huomioon. Näitä riskitekijöitä on esimerkiksi toiminnallinen-, vakaus-, suoritus-, käytettävyys-, testattavuus-, tietosuoja-, ymmärrettävyys-, skaalautuvuus- ja helppokäyttöisyysongelmat. Testaaja pyrkii löytämään virheitä kaikkialta ohjelmasta ennen kuin se menee käyttäjälle. (Tricentis 2019, viitattu 16.7.2019.)

Testausta on ainakin kahdenlaista. On muodollista testaamista (engl. Formal Testing), joka on tarkistamista sekä tutkimustestaamista (engl. Exploratory Testing), joka on tutkivaa. Muodollisessa testaamisessa voidaan käyttää automaatiota, jossa ennalta määritelty tieto syötetään ohjelmalle

ennalta määritetyissä toiminnoissa. Se on kaikki, mitä testiautomaatio voi tehdä. Tutkimustestaamisessa tutustutaan ohjelmaan, suunnitellaan ja suoritetaan testejä. Tuloksia tulkitaan sitä mukaa, kun niitä saadaan. Saatujen tuloksien pitäisi vaikuttaa seuraaviin testeihin. Muodollisella testaamisella selvitetään tunnettuja riskejä ja tutkimustestaamisella tuntemattomia mahdollisia riskejä. (Tricentis 2019, viitattu 16.7.2019.)

Testaaminen on toinen ala, joka kiinnostaa minua. Tekemäni testaukset työkalulle ovat tutkimustestauksen tyyppisiä, sillä yritän löytää sellaisia ääritilanteita, joissa työkalun koodi hidastuu tai ei toimi oikein. Testiautomaatio ei työkalulle ole vielä suoritettu, mutta ehkä sekin on vielä edessä, sillä toimistollamme on testaamisista vastaava yksikkö.

3.9 Yhdeksäs viikko: Animaatioiden siirtäminen

Maanantai 1.7.2019

Tänään oli ensimmäinen päivä, kun suurin osa yrityksen työntekijöistä oli aloittanut kesäloman ja toimisto oli lähes tyhjänä. Olin edellisellä viikolla tehnyt paljon erilaista testaamista, ja tänään halusin saada nuo testit valmiiksi sekä aloittaa animaatioiden harjoittelun työkalua varten.

Testaamisessa edellisviikolla minulla oli jäänyt kesken valittujen objektien validointi, myös silloin kun valittujen listalla oli hyväksyttäviä, piilotettuja ja kiellettyjä objekteja. Näiden validointi pitäisi toimia, vaikka ne olisivat minkälaisessa järjestyksessä. Eri listoilla lukitaan mitä tarkistettuja objekteja viedään lopullisesti 3D Studioon.

Korjasin edellisviikon koodia niin, että koodi valitsee aina myös piilotettujen objektien aliobjektit myös piilotettujen listalle ja piilottaa objektit Mayassa. Näin saan yksinkertaiset listat luotua. Listoja tulee kolme, jotka ovat valitut-, piilotetut- ja hylätyt objektit. Valitut objektit listasta suodatetaan piilotetut ja hylätyt objektit. Hylätyt objektit poistetaan valitut listalta ja piilotetut pysyvät tai poistuvat käyttäjän valinnan mukaan. Uusi koodi toimi oikein Mac-työkoneellani sekä Windows-testikoneella.

Seuraavaksi aloitin animaatioharjoitukset. Sillä hetkellä työkalu siirsi pelkästään staattisia objekteja, ja tulevaisuudessa se voisi myös ehkä siirtää Mayassa animoituja tapahtumia 3D Studioon. Tämän vuoksi minun pitää ensin käydä Mayasta läpi sen animaatioperusteita.

Tiistai 2.7.2019

Tälle päivälle tavoitteenani oli tutustua Pluralsight-verkkosivulta Mayan animoinnin perusteisiin. Videoita oli yli kolme tuntia, ja kokemuksesta tiesin, että niiden läpi käymiseen menee aina enemmän aikaa, sillä asioiden testaaminen vie aikaa ja katselu keskeytyy.

Olen työskennellyt aikaisemmin jonkin verran videoiden parissa, ja 3D puolella olen enimmäkseen keskittynyt mallintamiseen. Animaatioiden tekemiseen en ole vielä ehtinyt hirveästi perehtyä, mutta pystyn hyvin soveltamaan aikajanan ja 3D-mallien ymmärtämistä animoinnissa.

Yksinkertaisuudessaan se on vain sitä, että määritellään avainpisteitä (engl. keyframe) aikajanalle (engl. timeline). Ne pisteet sisältävät sijainti-, rotaatio-, skaala- tai jotain muita ominaisuustietoja ja aikajanapisteiden muuttuessa nuo tiedot muuttuvat myös.

Valitsemani kurssi oli minulle juuri sopiva, sillä siinä keskityttiin pelkästään animoinnin tekemiseen vaikuttaviin työkaluihin. Se ei sisältänyt animoinnin valmistelua kuten takilointi (engl. rigging), jossa luodaan animointia varten liikuteltavat luut, luurangot sekä ohjaustyökalut. Toinen yleinen aihe on painotus (engl. weighting), jossa polygonimallin muodot liitetään luotuun luurankoon kertomalla-kuinka paljon polygonin muodot ottavat vaikutteita luurangosta.

Tein verkkokurssin ja opin Mayan perustyökaluja. Seuraavaksi jatkoin 3D Studion puolella ja loin siellä yksinkertaisen animaation. Tämän projektitiedoston tulkitseminen jäi minulla kesken, joten jatkan sitä huomenna.

Keskiviikko 3.7.2019

Tälle päivää asetin tavoitteeksi saada animaation siirrosta parempaa ymmärrystä. Aloitin selvittämisen tarkkailemalla molempien ohjelmien animaatioihin liittyviä asetuksia ja työkaluja.

Ensimmäiseksi huomasin, että aikajana toimii 3D Studiossa sekuntiarvoissa, kun taas Mayassa aikajana toimii kuvataajuus periaatteella. Mayan oletusasetuksena kuvataajuus on 24 kuvaa sekunnissa. Päättelin, että tämä on tärkeä ottaa huomioon tekemässäni työkalussa. Jos Mayassa jokin avainpiste on kohdassa 12 ja kuvataajuus on 24, niin 3D Studiossa tuo kohta on 0,5 sekuntia alusta.

Seuraavaksi aloitin Pythonilla uuden moduulin luomisen, joka keskittyisi animaatioiden siirtämiseen. Jos koen myöhemmin, että koodi ei ole hyvä pitää moduulina, niin voin siirtää sen työkalun uipmaker moduuliin.

Löysin verkosta hyviä koodinpätkiä animaatioiden avainpisteiden selvittämiseksi, ja valitsin yhden moduulille pohjaksi. Tulkitsin koodin, ja sen kautta opin nopeasti uusia animaatioihin liittyviä funktioita Mayan Commands moduulista.

Torstai 4.7.2019

Tällä viikolla animaatioiden selvittäminen on edennyt hyvin, ja tavoitteeni oli saada ensimmäinen yksinkertainen animaatio siirrettyä Mayasta 3D Studioon.

Edellispäivänä olin saanut kirjoitettua jonkinlaista moduulinrunkoa animaatiokoodia varten. Testasin tämän käyttämistä työkalun uipmaker koodista, ja moduuli ei lähtenyt toimimaan niin kuin sen olin ajatellut. Tein vähän aikaa vianetsintää, kunnes päätin lopettaa moduulin käytön ja kirjoittaa animaatioihin liittyvän koodin työkalun uipmaker moduuliin.

Suunnittelin uipmaker koodiin tarkistuksia, joissa koodi haistelee, onko siirrettävässä esityksessä animaatioiden avainpisteitä. Jos sellainen havaitaan, niin koodi suorittaa uutta animaatioihin liittyvää logiikkaa, ja jos ei, niin koodi toimii kuten aikaisemminkin.
Juuri ennen työpäivän loppumista sain siirrettyä ensimmäisen animaation, niin että se näyttää aikajanalla oikealta, mutta 3D-mallit eivät yhdisty oikein. Yritän selvittää tuota vikaa huomenna.

Perjantai 5.7.2019

Tänään tavoitteeni oli saada animaatiot siirtymään ja toimimaan tekemälläni työkalulla. Aamulla työstetyssä versiossa animaatioiden avainpisteet siirtyivät Mayasta 3D Studioon, mutta mallit eivät tulleet näkyviin.

Aloitin vian selvityksen luomalla toimivan 3D Studio-animaation, jossa malli näkyy oikein. Vertasin tämän animaation sekä työkalun tekemää animaatiota. Huomasin pari kohtaa, joilla saattaisi olla vaikutusta mallien näkymiseen. Korjasin kohdat koodissa, ja mallit alkoivat tulla oikein näkyviin seuraavissa siirroissa.

Seuraavaksi tein paljon monimutkaisemman animaation Mayassa, jossa kaikki objektit liikkuvat. Tämänkin siirtäminen onnistui, ja kaikki objektit liikkuivat samalla tavalla 3D Studiossa. Ainoa virhe

oli, että z-akselin arvot eivät kertaantuneet miinus yhdellä. Korjasin tämänkin, ja siirsin uusimman version verkkoon.

Viikkoanalyysi

Tällä viikolla oli ensimmäistä kertaa tekemissä animaatioiden kanssa Mayassa, ja mielestäni opin todella hyvin, kuinka käyttää niitä työkaluja, jotka ovat minulle tällä hetkellä tärkeitä. Maya on tunnettu animaatiotyökalu, josta löytyy kaikki ominaisuudet, joita elokuva- ja peliteollisuus käyttävät. Voisin kuluttaa vaikka kuinka paljon aikaa opettelemalla sen eri animaatio-ominaisuuksia.

Tein myös paljon Python-ohjelmointia, ja se sujui nyt todella vaivattomasti. Työkalun objektien valinta- ja validointipäivitykset sain lisättyä koodiin ilman suurempia mietintöjä. Tätä auttaa varmasti se, että tunnen työkalun metodit ja funktiot.

Uuden animaatiomoduulin kanssa oli hieman vaikeuksia, eikä minulle selvinnyt, miksi se ei alkanut toimimaan. Keksin kuitenkin toisen tavan tuoda animaatiot koodissa, niin se riittää minulle.

3D Studion animaatio-ominaisuudet ovat hyvin yksinkertaiset, joten jouduin esimerkiksi suodattamaan Mayasta tulevaa animaatiotietoa. Uskon, että tutustun tulevaisuudessa lisää Mayan animaatiomahdollisuuksiin, sillä näitä taitoja saattaa tarvita tulevissa työtehtävissä.

3.10 Kymmenes viikko: Uusia testejä ja materiaaliongelmia

Maanantai 8.7.2019

Työkalun tekeminen on edennyt tähän mennessä todella hyvin, ja tarkistin verkossa olevaa työtehtävälistaa. Siinä ei ollut tekemättä merkittäviä avonaisia tehtäviä, joten päätin esitellä työkalun tilannetta työtoverilleni. Hän on ollut pitempään 3D Studion tekijätiimissä, ja sain häneltä heti hyviä huomioita koskien siirrettyjä valoja. Kirjasin näistä huomioista itselleni kolme uutta tehtävää.

Ensimmäinen huomio oli, että valot olivat liian isoja kooltaan. Olin pienentänyt kokoa staattisten esityksien siirrossa, mutta nyt uusissa animaatioita sisältävissä esityksissä valojen kokotieto tuli taas liian isona. Korjasin kokotiedon animaatioon liittyvässä koodiosuudessa.

Toinen huomio oli, että valojen voimakkuutta ei otettu mitenkään huomioon siirrossa, vaan kaikki valot saivat saman voimakkuusarvon. Selvitin että Mayassa valon tehoa määrittää Intensity-arvo, joka voi maksimissaan olla 1 ja että vastaava arvo 3D Studiossa on Brightness-arvo, jonka yleinen maksimi on 100. Lisäsin Mayan valo-objektien arvon kerrottuna sadalla siirtymään 3D Studioon.

Viimeinen huomio oli valojen värit, joihin en ollut keskittänyt yhtään huomiota aikaisemmin. Lähdin selvittämään, kuinka kahden järjestelmän valotyökalut eroavat toisistaan. Mayassa pystyi valitsemaan usean eri värimäärittelyn väliltä, kun taas 3D Studiossa käytetään pelkästään RGB-värimääritystä, jossa väri saadaan sekoittamalla punaista, vihreää ja sinistä. Myöhemmin huomasin, että Buddy-työkaverini olikin jo ottanut värit valoissa huomioon, kun hän oli tehnyt uip-tiedoston luojan runkoa, eikä minun tarvitse muuttaa näitä asetuksia tällä hetkellä.

Tiistai 9.6.2019

Eilen olin löytänyt uusia tehtäviä työkalua varten, ja tänään päätin kokeilla työkalua eri tavoin löytääkseni uusia vikatilanteita ja jotain parannettavaa. Loin uuden Maya esityksen, johon liitin objekteja, joissa on eri materiaaleja, tekstuureita ja animaatioita.

Tämän esityksen siirtäminen sujui vaivatta 3D Studioon ja objektit näkyivät työnäkymässä oikein. 3D Studiossa on myös muita näkymiä, joilla voi katsella lopullista 3D esitystä. Yksi näistä näkymistä on OpenGL Runtime Wiewer. Käynnistin esityksen siellä, ja kaikki näkyi mustavalkoisena. Aloitin työn selvittääkseni, mihin värit ovat kadonneet.

Objektit ja niiden materiaalit näyttivät oikeilta uip-tiedostossa. Se, että ne näkyvät myös työskentelynäkymässä oikein, pitäisi tarkoittaa, että kaikki on kunnossa. Kysyin työkavereideni mielipidettä, ja yksi heistä ehdotti, että tarkistaisin Qt Creator –ohjelmalla, saanko mitään vikailmoituksia. Qt Creator on Qt:n oma monialustainen integroitu ohjelmointiympäristö (engl. IDE = integrated development enviroment). En ollut käyttänyt tätä vielä kertaakaan, joten asensimme sen yhdessä. Työkaverini näytti, miten 3D Studio voidaan sillä käynnistää, ja saada sen kautta tarkempia ohjelmistolokitietoja. Valitettavasti näistä ei ollut apua, ja aioin jatkaa asian selvittämistä seuraavana päivänä.

Keskiviikko 10.6.2019

Tämän päivän tavoitteeni oli selvittää, miksi 3D Studiossa lopullisen työn katsominen OpenGL Runtime Viewer-työkalulla näyttää mustavalkoiselta. Objektit, niille annetut värit ja tekstuurit näkyvät oikein työnäkymässä siirron jälkeen, mutta tuolla työkalulla eivät.

Aloitin selvittämisen luomalla yksinkertaisen esityksen Mayasta, jossa oli pelkkä pallo yhden värin materiaalilla. Se näkyi 3D Studiossa siirron jälkeen Runtime Viewer-työkalussa mustavalkoisena. Tarkistin kaikki aikaisemmat testiesitykseni, ja niissäkin OpenGL Runtime Viewer-työkalunäkymät näkyivät mustavalkoisina, paitsi valotestissä, jossa testasin valojen eri värejä. Jossakin tilanteessa kokeilin uuden kameran lisäämistä, ja värit tulivat oikein näkyviin tämän jälkeen.

Jatkettuani testaamista ilmeni, että uuden kameran lisääminen ei ollutkaan pakollista tilanteen korjaamiseksi vaan riitti että, kameran piilotti ja palautti näkyviin. Aina, kun käyttäjä muokkaa esitystä, niin 3D Studio tallentaa siitä väliversion. Tuo väliversio toimii oikein, sillä se sisältää 3D Studion itse tekemää tietoa esimerkiksi väreistä ja tekstuureista, jotka se hakee fbx-tiedosta sekä projektikansioista.

Torstai 11.7.2019

Tänä päivänä tarkoitukseni oli jatkaa siitä, mihin edellispäivänä jäin. Olin alkanut ymmärtää, kuinka värit saadaan palautettua OpenGL Runtime Viewer-näkymään, ja aioin seuraavaksi kokeilla materiaalien värien määrittämistä Mayassa ja tuon tiedon siirtämistä 3D Studioon.

Aikaisemmin tämä väritieto on tullut 3D Studion kautta, kun se tulkitsee fbx-tiedostoa. Heti ensimmäisissä testauksissa huomasin, että materiaalin diffuse-värin määrittäminen uip-tiedostoon tuo väriä OpenGl runtime viewer-näkymään.

Tarkastelin aikaisemmin tekemääni koodia ja huomasin, että olin käsitellyt paljon materiaaleja, mutten ollut hakenut niistä ikinä väritietoja. Nopeiden hakukone-etsintöjen jälkeen löysin getAttribute-toiminnon, jolla sain väritiedon kaivettua materiaalikohtaisesti.

Lisäsin tämän tiedon iteroitumaan koodiin, ja aluksi kaikki näytti hyvältä. Pian huomasin, että hakemani väritieto ei ollut sama, jonka 3D Studio määrittää. Sävyt olivat huomattavasti tummempia, ja 3D Studion määrittämät värit olivat myös mielestäni silmää miellyttävämmät.

Loin tarvittavat funktiot tietojen lisäämiseen, mutta jätin ne kommentoiduiksi toistaiseksi. Tämä tarkoittaa sitä, että koodi on edelleen olemassa, muttei vaikuta mihinkään.

Viikkoanalyysi

Tämän viikon aikana keskityin ensimmäistä kertaa pelkästään koodin testaamiseen ja parantamiseen. En tehnyt mitään isoja muutoksia koodiin, mutta löysin paljon uutta työkalun toimivuudesta. Esimerkiksi tuo OpenGL Runtime Viewer-näkymän värien katoamisen ymmärtäminen oli tärkeä havainto, sillä opin sitä kautta myös uutta 3D Studion toiminnasta. Olen muutenkin keskittynyt enemmän Mayan käyttöön kuin 3D Studioon, joten on hyvä, että löydän siitä uusia puolia.

Minulle selvisi, että vaikka 3D Studion editori ja OpenGL Runtime Viewer-näkymä toimivat saman renderöinnin kautta, ne hakevat silti tietonsa eri kautta. Uip-tiedosto on, mikä määrittää esityksen,

ja jos editorissa näkyvät värit, niin se pitäisi näkyä myös muissa näkymissä. Tulen varmasti selvittämään tätä tarkemmin tulevina viikkoina työkalun kehitystyössä.

Päivityksiä tehdessä mietin ensimmäistä kertaa, että nämä saattavat olla ylimääräisiä ominaisuuksia, ja että koodin pitää olla sen tyyppistä, että päivitykset saadaan helposti kumottua. 3D Studion tekemä värimäärittely esimerkiksi on mielestäni paljon parempi, joten joudun tiedustelemaan esimieheltäni sekä Buddylta, mikä on heidän kantansa asiaan. Jätetäänkö värien määrittely 3D Studiolle vai haetaanko ne itse Mayasta?

4 POHDINTA

Aloittaessani työt Qt:lla minulla ei ollut selkeää kuvaa tulevasta työtehtävästä, ja mitä se pitäisi sisällään. Mielestäni olen sisäistänyt työtehtävät lyhyessä ajassa, ja nyt minulla on varma olo ohjelmoidessa Pythonilla ja käyttäessä Mayaa. Suurin kehitys on tapahtunut siinä, että osaan aloittaa ongelmien ratkaisun, kun minulle annetaan uusia työtehtäviä. Ohjelmoinnissa olen oppinut tulkitsemaan Mayan ohjekirjastoja, joista löydän tiedot, kuinka käsitellä objekteja koodilla Mayan sisällä.

3D-työskentelyssä olen päässyt tutustumaan yhteen alan käytetyimmistä työkaluista ja oppinut sen käytön perusteita. Tutustuminen myös 3D-puolen ohjelmointiin on ollut hyvin mielenkiintoista. Aikaisemmin minulle ei ollut tullut mieleenkään, että voisin itse ohjelmoida työkaluja 3D-mallintamista varten. Jos saan tällaisen työtehtävän, niin osaan mielestäni jäsentää tarvittavat ominaisuudet uudelle työkalulle.

Parasta on ollut huomata, että samat metodit, joita olen aikaisemmin käyttänyt asiakaspalvelutyössä ongelmien ratkaisemiseksi, toimivat myös ohjelmointityössä. Joskus kannattaa puhua vieruskaverin kanssa ääneen, miksi jokin ei toimi ja ratkaisu voi löytyä nopeasti yhdessä. Tällä työtavalla pääsee myös rikkomaan tutustumismuureja uusien työkavereiden kanssa ja saattaa päätyä tekemään töitä henkilöiden kanssa, joihin ei muuten tutustuisi helposti.

Toinen työmenetelmä oli löytää jotain virkistävää työhön liittyvää tekemistä, kun mieli alkaa turtua jonkin asian tekemiseen. Näin minulla kävi muutaman kerran Python-ohjelmoinnin kanssa, ja kun happihyppy ulkona ei enää auttanut, tästä oli hyötyä. Tähän työtehtävään kuuluu paljon minulle mielenkiintoisia aiheita jo pelkästään 3D-malleja koskien, joten näiden opettelu välillä on hyödyllistä ja virkistävää. Pelkästään puolen tunnin eri asian käsittely saattaa riittää siihen, että näkee tuoreilla silmillä ratkaisuja aikaisempaan ongelmaan.

Työkalua ohjelmoidessa mietin paljon, että pitääkö tämän koodin pätkän olla oliomuodossa vai proseduraalisessa, pitäisikö koodit yhdistää yhdeksi isoksi koodinpätkäksi vai pidetäänkö ne erillään osissa. Mielestäni tekemäni ratkaisu, jossa työkalun ikkuna on ainut oliomuotoinen koodi ja muut ovat proseduraalisia moduuleita, on ollut hyvä ja helposti päivitettävä. Osasyy tähän ratkaisuun oli se, että ikkunan pitää pystyä tallentamaan itseensä tietoja, ja se tavallaan hallitsee muita

moduuleita. Tämän kokemuksen ansiosta, kun jatkossa teen jotain tämäntyyppistä, osaan varmasti paremmin suunnitella ohjelmarunkoa.

Päiväkirjamuotoisen opinnäytetyön suurin etu oli se, että iltaisin tuli kerrattua tehtyjä työtehtäviä ja näitä varten tulee opiskeltua niihin liittyvää asioita. Tämä lisäteorian opiskelu on hyödyntänyt eniten, sillä olen nyt oppinut joitain 3D-mallinnukseen liittyviä asioita, vaikka olen käyttänyt näitä monesti aikaisemmissa töissä. Esimerkki tästä on vaikka valojen toimintaperiaatteet 3D-mallintamisessa. Näiden 10 viikon aikana on myös yleisesti tullut opittua todella paljon uudesta työtehtävästä, joista kaikista olen päiväkirjaosiossa maininnut.

Etsiessäni koodiongelmiin ratkaisuja Technical Artist -verkkofoorumeilta olen saanut myös paremman kuvan tästä työroolista. Technical Artistin kuuluu osata 3D-työnteon perusteet sekä osata ohjelmoida hyvin. Vaikuttaa siltä, että Technical Artistin rooli on tehdä työkaluja 3D-artisteille sekä suorittaa muita teknisempiä tehtäviä, joita pelkästään 3D tai ohjelmointiin erikoistuneen ei kannata tehdä. Suurin osa ammattilaisista, joita löysin, oli ensin työskennellyt pitkään perinteisellä ohjelmistoalalla ja siirtynyt sieltä peli-, elokuvateollisuuteen tai muille 3D:tä sisältäville aloille.

Tärkein huomio oli, että tätä tehtävää varten minulla on vielä paljon opeteltavaa. Tämän vuoksi minun on ylläpidettävä uusien 3D-työhön liittyvien asioiden opettelua myös jatkossa. Huomasin myös, että kykenen näihin tehtäviin, kun teen tarpeeksi taustatutkimusta ennen kuin aloitan täysin uuden haasteen. 10 viikon aikana tuli monta itselle täysin tuntematonta käsitettä ja teknologiaa, ja pystyin käyttämään näistä jokaista, kunhan tein aluksi tarpeeksi taustatyötä.

Peleihin liittyvässä ohjelmoinnissa olen aikaisemmin käyttänyt API-ohjeiden lukemista jonkin verran, mutta en tuolloin ollut oppinut tulkitsemaan niitä kovin hyvin. Tässä työroolissa Mayan ohjeiden lukeminen on ollut lähes päivittäistä, ja uskon että tästä on hyötyä, jos nyt aloittaisin esimerkiksi uuden Unity-peliprojektin. 3D-työkalujen käyttäminen on nyt luontevampaa, ja tunnistan paremmin eri työkalujen toiminnallisuudet ja mihin niitä käytetään.

Työni analysointi on nopeuttanut oppimistani tähän rooliin. Viikoittaisten analyysien aikana olen kerrannut tekemiäni uusia työtehtäviä sekä etsinyt niihin liittyvää aineistoa. Aineistoista en yleensä löytänyt niinkään uusia työtapoja, mutta opin, miksi tekemäni muutokset vaikuttavat. Tekemäni työkalu menee yritysasiakkaiden käyttöön syksyllä ja sain Qt:lta vakituisen työpaikan tämän työnäytteen perusteella.

LÄHTEET

Autodesk Maya 2019. Learn. Viitattu 2.6.2019, https://knowledge.autodesk.com/support/maya/learn?sort=score.

Autodesk Maya 2015. Nodes and attributes. Viitattu 2.6.2019, http://help.autodesk.com/view/MAYAUL/2015/ENU/guid=Nodes_and_attributes_Nodes_and_attributes_overview.

Byl, P, 2019. Shader Development deom Scratch for Unity with Cg. Sisäinen lähde. Viitattu 15.7.2019, https://www.udemy.com/unity-shaders/.

Crease, A 2019. 3D Modeling Basics. Instructables. Viitattu 2.6.2019, https://www.instructables.com/id/Intro-to-3D-Modeling/.

Pierce, R 2019. Vertices, Edges and Faces. Viitattu 2.6.2019, https://www.mathsisfun.com/geometry/vertices-faces-edges.html.

Python 2019. What is Python? Executive Summary. Viitattu 2.6.2019, https://www.python.org/doc/essays/blurb/.

Python 2019. Comparing Python to Other Languages. Viitattu 2.6.2019, https://www.python.org/doc/essays/comparisons/.

Python 28.3.2019. 19.9. xml.dom.minidom – Minimal DOM implementation. Viitattu 2.6.2019, https://docs.python.org/2/library/xml.dom.minidom.html.

Qt Company 2019. What is qt. Viitattu 15.7.2019, https://www.qt.io/what-is-qt/.

Sorva, J 2017. Verkkokurssi:CS-A1110 Luku 7.5: Ohjelmointiparadigmoista. Aalto University, Viitattu 16.7.2019, https://plus.cs.hut.fi/o1/2017/k07/osa05/.

Tricentis 2019. Software Testing Explained, Viitattu 16.7.2019, https://www.tricentis.com/soft-waretesting/software-testing-explained.

Zentgraf, D 27.04.2015. What Every Programmer Absolutely, Positively Needs To Know About Encodings And Character Set To Work With Text, Viitattu 16.7.2019, http://kunststube.net/encod-ing/.