
Jimi Engholm

# Piirretyn näköinen 3D-pelihahmo

## Millä tekniikoilla 3D-hahmo saadaan näyttämään 2D:nä tuotetulta Unreal Engine 4:ssä?

Metropolia Ammattikorkeakoulu

Medianomi

3D-animaatio ja -visualisointi

Opinnäytetyö

20.5.2021

## Tiivistelmä


| | |
|---|---|
| Tekijä(t): | Jimi Engholm |
| Otsikko: | Piirretyn näköinen 3D pelihahmo |
| Sivumäärä: | 51 sivua + 2 liitettä |
| Aika: | 20.5.2021 |
| Tutkinto: | Viestintä |
| Tutkinto-ohjelma: | Medianomi |
| Suuntautumisvaihtoehto: | 3D-animointi ja -viestintä |
| Ohjaaja(t): | Lehtori Jaro Lehtonen |


---


Opinnäytetyössä käsitellään projektiosuutta 3D-hahmon luomisprosessista tyylillä, jolla sen saa näyttämään 2D-hahmolta pelimoottorissa, sekä teoriaa kolmi- ja kaksiulotteisen grafiikan luomisen tekniikoiden heikkouksista, vahvuuksista ja yhdistelmistä.

Opinnäytetyö olettaa lukijan tietävän perusasiat 3D-työskentelystä mallinnuksen, teksturoinnin, rigaamisen ja animoinnin alueilla. Pelimoottorin ja pelisuunnittelun teknisiä asioita opinnäytetyössä ei juuri käydä konkreettisesti läpi, mutta niitä pohditaan suhteessa aiheena olevan visuaalisen tyylin luomiseen.

Tavoitteena on saada ainakin yksi toimiva työprosessi selville projektiosuuden aiheena olevan tyylin aikaansaamiseksi sekä ymmärtää, mitä aihealueita tulisi tutkia vielä lisää lopputuloksen parantamiseksi.

Työn lopputuloksena on onnistunut pelihahmo, jota voisi animoida ja käyttää pelaajahahmona Unreal Engine -pelimoottorissa. Työn aikana selviää myös, kuinka tärkeää artistisen näkemyksen toteutuksen kannalta on tietää, miten suuri osa toivotusta kuvasta voidaan tuottaa tietokoneen laskelmilla ja miten suuri osa on pakko tehdä taiteilijan käsityöllä, riippuen projektin tarpeista.



Avainsanat: 2D-animaatio, 3D-animaatio, videopeli, pelihahmo, Unreal, pelimoottori, anime, cel-shading

## Abstract


| | |
|---|---|
| Author(s): | Jimi Engholm |
| Title: | A 3D game character in a 2D style |
| Number of Pages: | 51 pages + 2 appendices |
| Date: | 20 May 2021 |
| Degree: | Bachelor of Culture |
| Degree Programme: | Media |
| Specialisation option: | 3D Animation and Visualisation |
| Instructor(s): | Jaro Lehtonen, Senior Lecturer |


---


This thesis describes creating a 3D character in a style that looks like 2D in a game engine and the strengths, weaknesses, and combinations of different techniques to create 3D and 2D graphics.

The thesis assumes that the reader knows the basics of 3D modelling, texturing, rigging, and animating. The technical side of working with game engines and designing games are not dealt with in detail, but they are discussed in relation to creating the wanted style of the 3D character created in the project.

The goal of this thesis is to find at least one working pipeline for creating the wanted style and to understand what should be researched more to improve and develop the techniques further.

In conclusion, a complete game character, which could be animated and used as a playable character in Unreal Engine, has been designed. During the process it was discovered just how important it is to know how much of the wanted final image can be computer-generated and what needs to be the responsibility of the artist.



Keywords: 2D-animation, 3D-animation, video game, game character, Unreal, game engine, anime, cel-shading

**Sisällys**

# 1 Johdanto

Animaatio on kiinnostanut minua lapsuudesta asti. Aloitin animoinnin tekemällä stop motion -animaatioita Legoilla, paperilla ja muovailuvahalla. Piirrettyä animaatiota en juurikaan ole opetellut, mutta nyt 3D-tietokoneanimaatiota opiskelleena sen mahdollisuudet ja vahvuudet tietokoneanimaatioon verrattuna kiinnostavat ja olen miettinyt, kuinka paljon asioita 2D-grafiikkaa käyttävästä animaatiosta voisi tuoda videopeleihin. Tarkoitan nimenomaan 3D-peleihin, sillä kaksiulotteiset pelithän voivat tehdä periaatteessa mitä vain 2D-animaatiotkin, mutta tietyt asiat eivät toimi 3D:ssä yhtä hyvin.

Suurin ero 3D- ja 2D-grafiikan tuottamisessa on se, että tietokoneen laskelmat tekevät 3D-kuvan tekemisestä tietyssä mielessä nopeampaa ja tehokkaampaa, mutta toisaalta matematiikka vie artistista vapautta kylmällä logiikallaan. Näiden kahden, ihmisen ja koneen piirtämien kuvien, välinen tasapaino löytyy eri projekteista siitä, kuinka paljon kontrollia halutaan ottaa kuvan eri aspekteista. Esimerkiksi valaistus voi ihmisen piirtämänä olla mitä hän haluaa, mutta kone laskee sen juuri niin kuin se on ohjelmoitu valo laskemaan.

Videopelien kuvat lasketaan reaaliaikaisesti, joten ihmisen piirtämää jälkeä ei aina voi käyttää, etenkään kolmiulotteisissa peleissä, joissa pelaaja olettaa voivansa katsoa kaikkea pelin sisältöä mistä tahansa kuvakulmasta. Videopeleillä ja taiteella muutenkin on tosin se ässä hihassa, että aina voi huijata. Pelien tekijät ovat aina etsineet tapoja tehdä jokin asia helpommin, halvemmin ja nopeammin kuin se tehtäisiin mm. elokuvissa. Veden realistinen simuloiminen voi elokuville kestää suurillakin renderöintifarmeilla (monta tietokonetta laskemassa kuvia samaan videoon) kauan, mutta videopeleihin käytetään metodeja, jotka näyttävät lähes yhtä hyviltä, mutta ovat paljon kevyempiä laskea reaaliajassa. Esimerkiksi vesiefektien piirtäminen levyille, jotka liikkuvat 3D-tilassa niin nopeasti ja tiheässä rykelmässä, ettei niitä erota simuloiduista vesiefekteistä, on peleissä usein käytössä oleva tapa.

Tässä opinnäytetyössä halusin tutkia eroja ja raja-aluetta 2D:n ja 3D:n välillä pelimoottorissa, jotta osaisin käyttää oppimaani omissa projekteissani. Mitä kaikkea piirtämällä animoitavaa voisi matkia tai huijata jollain 3D-tekniikalla? Mikä taas ei onnistu pelimoottorissa? Opinnäytetyöni olettaa lukijan tietävän perusasiat 3D-hahmon luomisesta ja animoimisesta eikä mene pintaa syvemmälle asioihin, joita en projektiosuutta varten tutkinut. Alkuperäinen suunnitelmani sisälsi pelihahmon liikkumiseen ja ohjaamiseen liittyviä asioita, mutta en päässyt niin pitkälle, koska jo hahmon luominen käyttövalmiiksi oli tarpeeksi iso aihe yhdelle opinnäytetyölle.

Tavoitteeni projektiosuudessa oli saada selville, kuinka lähelle Promare-elokuvan tyylistä hahmoa pääsisin Unreal Enginellä (Pelimoottori) enimmäkseen Arc System Works -pelistudion tekniikoita käyttäen (sillä he ovat tässä tyylissä johtava pelistudio tällä hetkellä) ja miten tämä eroaisi Blenderillä (3D-animaatio-ohjelma) renderöidystä tavasta.

Ensimmäisessä luvussa johdannon jälkeen käyn läpi eroja 3D- ja 2D-animaatioiden välillä ja mainitsen esimerkkejä siitä, miten animaatioelokuvat ja videopelit käsittelevät näiden erojen heikkouksia ja vahvuuksia tyyleissään. Sen jälkeen kerron työprosessistani projektiosuudessa pelihahmon luomisesta ja lopuksi yhteenvedossa kerron, mitä itse opin ja mitä jatkossa kannattaisi tutkia lisää.

# 2 Teoriaa 3D:n ja 2D:n eroista animaatiossa

Videopelit ja visuaalinen viihde ylipäänsä ovat edistäneet teknologiaa koko olemassaolonsa ajan. Kolmiulotteinen animaatio toi monia mahdollisuuksia, jotka olivat vaikeita 2D:llä tai vain veivät aikaa. Nykyään 3D on käytössä lähes kaikessa 2D:nkin tuottamisessa. 2D-grafiikkaakin silti käytetään, koska sillä on artistinen etulyöntiasema 3D-animaatioon kontrollin määrän takia. 3D-tuotokset ovat usein, printtaamista ja hologrammeja lukuun ottamatta, loppujen lopuksi kaksiulotteisia kuvia joko yksittäisinä tai videomuodossa. Kaiken, mitä 3D:llä voi kaksiulotteiseen kuvaan tehdä, voi tehdä myös suoraan 2D:nä, vaikkakin hitaammin ja se vaatii mahdollisesti enemmän osaamista. Mutta kaikkea mitä 2D:llä voi tehdä, ei voi tehdä 3D:llä, ainakaan vielä.

Kaksiulotteisessa grafiikassa ainoastaan katsojan ymmärryksellä kuvasta on väliä. Niin kauan kuin katsojan kokemus ei siitä kärsi, kuvassa saa tapahtua mitä vain, vaikka ne tapahtumat eivät noudattaisikaan fysiikan lakeja. Deformaatio eli muodonmuutos on asia, joka on 3D-tietokonegrafiikalla laskiessa todella vaikeaa, ja yksi ratkaisu ei usein toimi muissa tilanteissa. Artisti voi kaksiulotteisessa animaatiossa vain piirtää välimuotoja kahden kuvan väliin, mutta miten tietokone laskisi kahden täysin erilaisen objektin välimuodot? 3D-laskelmat perustuvat loogiseen luonnon lakeja (ainakin enimmäkseen) noudattavaan matematiikkaan. Tämä matematiikka ei aina päde piirrettyihin, ja siksi 2D-asioiden tuottaminen 3D-muodossa on hyvin vaikeaa. Animaatioelokuvissa ja muissa ”valmiissa kuvapaketeissa” on se hyvä puoli, että tekniikoita voi yhdistellä niin paljon kuin haluaa ja sen voi tehdä niin hitaasti kuin haluaa.

## 2.1 Animaatioesimerkit

### 2.1.1 Spider-Man: Into the Spider-Verse (Sony 2018)

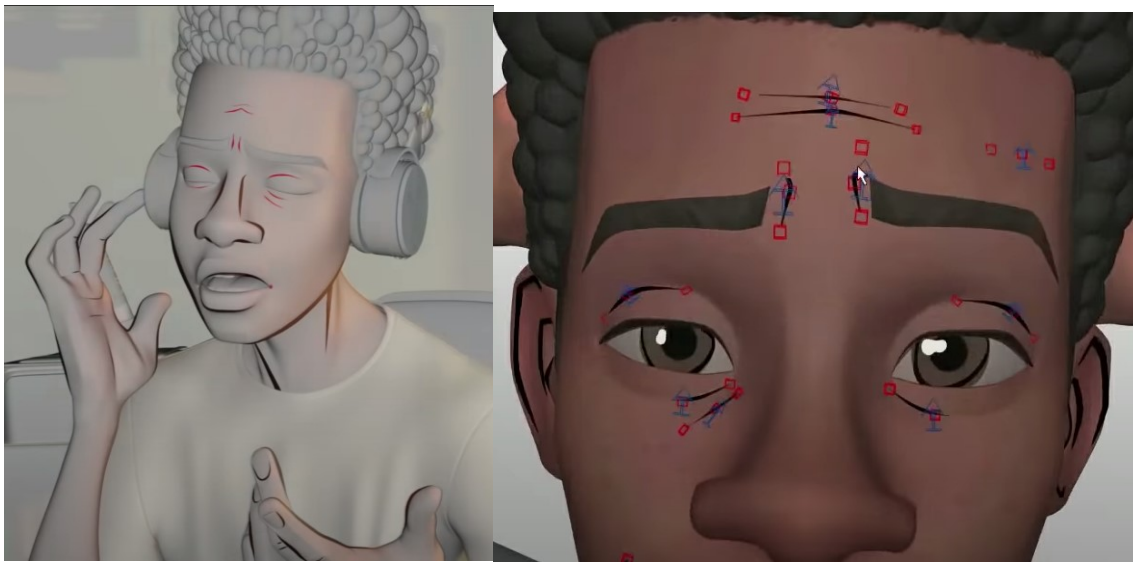
Sonyn Spider-Man: Into the Spider-Verse -elokuva (2018) on 3D-animaatio, jossa on käytetty 2D-elementtejä sarjakuvamaisen lopputuloksen aikaansaamiseksi. Spider-Versen tekijät halusivat elokuvan visuaalisen ilmeen olevan ainutlaatuinen. Tyylillä lähdettiin hakemaan ilmettä, jossa yhdistyisi parhaita puolia 2D- ja 3D-animaatiosta. Vuoden kestäneen testaamisen jälkeen studio sai 10 sekuntia animaatiota, johon he olivat tyytyväisiä. Haluttuun lopputulokseen pääseminen ei siis ollut helppoa.

”Don’t emulate reality, and don’t make it a cartoon.” (Beveridge 2019 Snyderin 2019 mukaan)

Spider-Verse-tyyliin kuului sekunnissa näytettävien kuvien määrän laskeminen 3D-animaatiolle (ja länsimaalaiselle 2D-animaatiollekin suurimmaksi osaksi) tyypillisestä kahdestakymmenestäneljästä kahteentoista. ”On ones” -animaatiotyyli (24 kuvaa sekunnissa) tarkoittaa, että videossa, joka näyttää 24 kuvaa sekunnissa, jokainen kuva on erilainen. ”On twos” (12 kuvaa sekunnissa) tarkoittaa, että vain joka toinen kuva liikkuu. Kuvien määrä voi olla eri hahmoille ja taustoille, tai vaihdella kesken animaation. Esimerkiksi kohtauksissa, joissa elokuvan hahmot liikkuvat sulavasti ja/tai itsevarmasti, ne animoidaan suuremmalla kuvamäärällä, kun taas kömpelöt liikkeet näytetään vähemmällä.

Motion blur, eli liikkeen hämärtyminen, on yleensä realismiin liitetty efekti, jolla nimensä mukaan liikkeessä olevat asiat sumennetaan. Piirretyssä animaatiossa käytetään yleensä tämän sijaan smear-tekniikkaa (suom. tahra). Siinä liikkeen välikuvia tyylitellään mm. vauhtiviivoilla ja ”lisäraajoilla”, jotka parantavat liikkeen tuntua. Motion blur- ja on twos -animaatio sopivat hyvin yhteen, sillä kun kuva vaihtuu vain joka toinen kerta normaaliin verrattuna, on silmällä aikaa katsoa kuvia kauemmin. Jos kuvat olisivat liikkeessä sumeita, niin liikkeet eivät olisi yhtä selkeitä kuin ne ovat näiden lisättyjen efektien kanssa ilman sumennusta.

Hahmojen 3D-mallien rigeihin, eli animaation ohjaamisen työkaluihin, on Spider-Versessä lisätty kaksiulotteisia ääriviivoja 2D-animaattorien ohjeistuksen ja tekoälyn laskelmien mukaan. Näillä viivoilla hahmoista saatiin selkeästi ilmeikkäitä piirretyn animaation tapaan. Kaikkea ei kuitenkaan saatu tehtyä itse 3D-animaatio-ohjelmien sisällä. Jotkut asiat, esimerkiksi smear-efektit voitiin piirtää 3D-renderöityjen kuvien päälle. (CGSociety 2019).



Kuva 1. Miles-hahmon kasvojen viiva-rigi. CGSociety (2019)

Kuva 2. Smear-efekti liikkeen välikuvissa. CGSociety (2019)

### 2.1.2 Promare (Studio Trigger 2019)

Studio Triggerin anime-elokuva Promare on 2D-animaatio, jossa on käytetty apuna 3D-elementtejä tilan tunnun ja kamera-ajojen parantamiseksi. 3D-osuudet on tehty sopivaksi 2D-tyylin kanssa. Promaren trailerissa (IGN 2019) voi nähdä, kuinka kamera-ajot on tehty sulaviksi käyttämällä 3D-versiota hahmoista ja tiloista esimerkiksi kohdassa 1:06–1:09. Tämä olisi piirtämällä vaatinut liikaa

työtä näin yksinkertaiseen kohtaukseen. 3D-malleja käytettiin myös suurimpaan osaan kohtauksista, joissa näkyy robotteja, autoja tai rakennuksia, sillä geometrisia muotoja on helpompi tehdä perspektiivissä 3D:llä kuin piirtämällä. Trailerin kohdassa 0:44–0:48 näkyvä hahmo robotin sisällä on myös tehty 3D:llä, mutta tyyli ei juuri eroa 2D-versiosta.

Kohtauksissa, joissa on käytetty 3D:tä, ei juuri näy deformaatiota. Kun hahmot liikkuvat tyylitellymmin, animaattorit vaihtavat 2D-animaatioon. Tämä vaihtelu on ollut aikaisemmin hyvin vaikeaa tehdä vakuuttavasti, mutta nykyään siinä ollaan jo niin taitavia, ettei etenkään nopeissa kohtauksissa sitä välttämättä huomaa. 3D-animaation suhteellinen animointinopeus kohtauksiin, joissa perspektiivi on tärkeä eikä tyyliteltyä liikettä käytetä, on houkutellut studiot kehittämään siihen vaihtamista mahdollisimman sulavaksi.

## 2.2 Peliesimerkit

### 2.2.1 Resident Evil 2 (Capcom 1998)

Ongelmaksi 2D:ltä näyttävää 3D-animaatiota tavoitellessa nousevat reaaliaikaiset renderöinnit, eli kuvat, jotka pitäisi laskea katsomishetkellä. Videopelit suurimmaksi osaksi renderöidään juuri sillä hetkellä, kun niitä pelataan. Sellaisiakin tapauksia löytyy, joissa suurikin osa pelistä voi olla valmiiksi renderöityä. Esimerkiksi Resident Evil 2 -peli, jossa taustat olivat valmiiksi 3D:n avulla renderöityjä 2D kuvia kolmiulotteisen ympäristön sijaan. Hahmot renderöitiin 3D:nä reaaliaikaisesti taustakuvien päälle varjoineen. Näin peli näytti paremmalta vähemmillä renderöintilaskelmilla. Etukäteen renderöinti vähentää reaaliaikaisten laskutoimitusten määrää tietokoneelta ja antaa artistille vapauksia. Mutta tässä esimerkissä on otettu täysi kontrolli kamerasta, eli paljon kontrollia pelaajalta, jotta taustat sopisivat peliin. Kamerakulmia kontrolloidaan niin, että taustat näkyvät vain juuri niistä kulmista, mistä artisti on tarkoittanut niiden näkyvän, Vapaa kamera paljastaisi, ettei pelaaja ole oikeassa kolmiulotteisessa tilassa.


Kuva 3. Capcom. Resident Evil 2 (1998)

Pelaaja ja zombiet ovat 3D-hahmoja, mutta tausta on kaksiulotteiseksi renderöity kuva, jonka päälle on pistetty 2D-animoidut liekkiefektit.

Entä jos haluaa 3D-peliinsä samanlaisia oikeasti liikkuvia ja muotoaan muuttavia asioita kuin 2D-animaatioissa, mutta kolmiulotteisena ja reaaliaikaisesti renderöitynä?

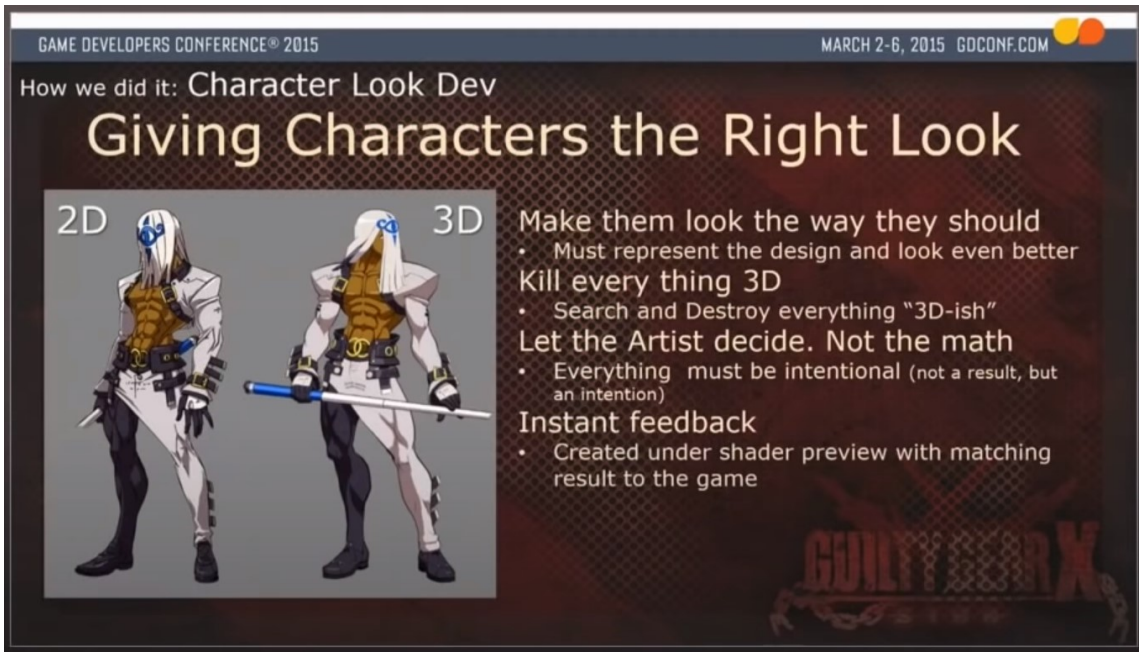
### 2.2.2 Guilty Gear Xrd (Arc System Works 2014)

Arc System Works on pelistudio, jolla on paljon 2D-pelien tuottamisen historiaa. Studio päätti kuitenkin tehdä uusista Guilty Gear -sarjan peleistä 3D-renderöityjä. Tyylin he halusivat Xrd-osassa pitää japanilaisen animaation näköisenä, niin kuin sarjan edeltävissä 2D-grafiikkaa käyttävissä osissa. Tähän työhön toi helpotusta se, että Guilty Gear on sivultapäin kuvattu kaksintaistelupeli, eli kameran liikettä voidaan hallita täysin, eikä kaiken tarvitse näyttää hyvältä jokaisesta kulmasta. Pelissä kuitenkin käytetään tietyissä kohtauksissa kameraajoja, joten grafiikkaa ei voitu renderöidä etukäteen.


Kuva 4. Arc System Works. Guilty Gear Xrd (2014)


Kuva 5. Arc System Works. Guilty Gear (1998)

Tulen tässä opinnäytetyössä viittaamaan usein Game Developers Conferencen, eli GDC:n vuoden 2015 esitykseen, jossa tekninen artisti Junya C Motomura kertoo Guilty Gear Xrd pelin luomisprosessista ja sen renderöintitekniikoista, joita itse sovelsin tämän opinnäytetyön projektiosuuteen (GDC 2015). Koko konferenssivideo antaa heidän tekniikoistaan paremman kuvan kuin itse voisin tässä kirjallisessa muodossa nopeasti käydä läpi, joten jos aiot tätä renderöinti-tyyliä enemmän tutkia yleisellä tasolla, niin kannattaa kyseinen video katsoa. Tässä opinnäytetyössä käyn läpi oman työprosessini yhtä tiettyä lopputulosta hakien ja kerron ongelmista ja ratkaisuista, joista voi olla apua vastaavanlaista projektia tehdessä.




Kuva 6. Guilty Gear -pelin hahmojen tyyli. GDC (2015)

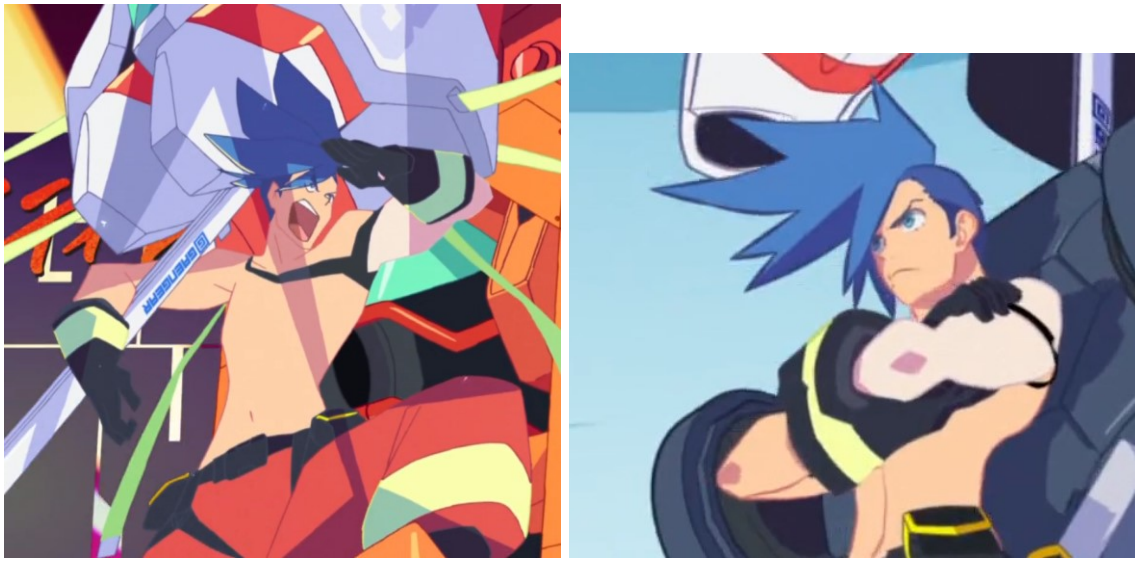
”Let the Artist decide. Not the math.” (Motomura 2015) 3D-renderöinnin vahvuus käyttää matematiikkaa tarkkaan laskemiseen on myös sen heikkous, kun halutaan artistista kontrollia.

# 3 Prosessi

## 3.1 Design

Otin projektini tyylin malliksi Studio Triggerin tekemän Promare-nimisen elokuvan. Aloitin tutkimalla Studio Triggerin tyyliä piirtää hahmoja ja suunnittelin oman hahmoni tiettyjä piirteitä Triggeriltä lainaten. Tyylini tärkeimmät ominaisuudet olivat (japanilaiselle animaatiolle ominaisen tyylittelyn lisäksi) yksinkertaiset muodot, varjot ja värit, värilliset ääriviivat ja kasvonpiirteiden läpinäkyminen hiuksista. Tutkin myös ilmeitä ja sitä, miten kasvojen 3D-mallin tulisi pystyä muuttumaan ilmeiden mukaan.

Mitä en projektin alkuvaiheessa tajunnut tehdä oli kartoittaa tarkasti, minne varjot tulisivat eri kulmista. Tämä olisi pitänyt tehdä ennen kuin mallinsin hahmoni saadakseni aikaiseksi Arc System Worksin tasoista varjostusta, sillä kuten tulen selittämään myöhemmässä renderöintiluvussa, varjojen paikka riippuu mallin geometriasta.



Kuva 7. Galo Thymos 2D (vasen) ja 3D (oikea). Studio Trigger. Promare (2019)

Promaren päähenkilö Galo on suunniteltu sopivan yksinkertaiseksi 3D-mallinnettavaksi, niin kuin hän joissain kohtauksissa näkyykin.


Kuva 8. Gueira. Studio Trigger. Promare (2019)

Otin hahmoni hiuksien inspiraatioksi toisen Promare hahmon, Gueiran.

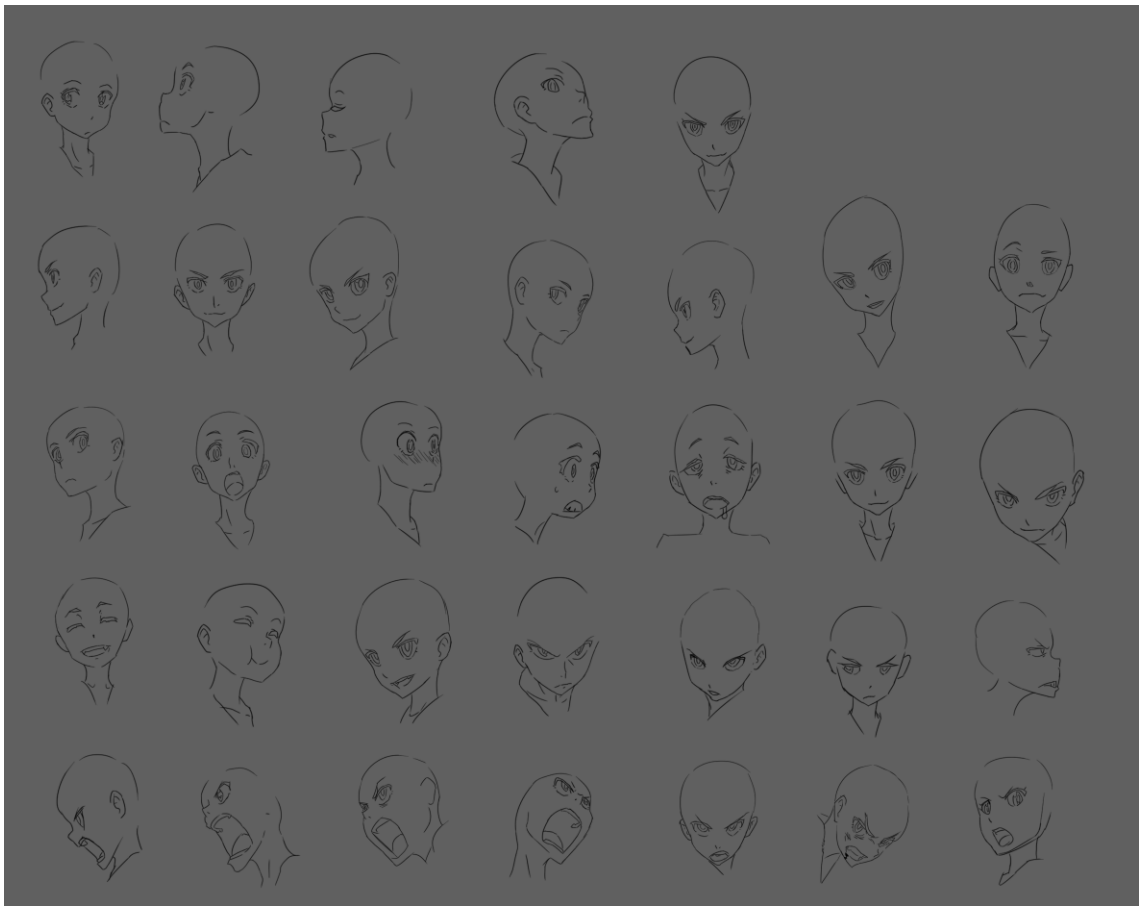
Kuva 9. Valmis hahmodesign

Kuva 10. Galon silmien läpinäkyvyys hiuksista. Studio Trigger. Promare (2019)

Kuva 11. Hahmon pään design

Kaksiulotteisten hahmojen hiukset usein piirretään eri tavalla eri kulmista. 3D-mallissa tällaista perspektiivin kanssa leikkimistä on vaikea harrastaa, koska jos haluaa yhden hiusten osan näkyvän vain tietystä kulmasta, täytyy se fyysisesti piilottaa jonnekin muista kulmista katsottaessa.

Tämä designvaikeus kannattaa ottaa huomioon jo piirtäessä hahmoa, ettei hiustyylin mahdottomuus tai tietyistä kulmista huonolta näyttäminen tule vastaan vasta kesken mallintamisen, niin kuin minulle kävi.



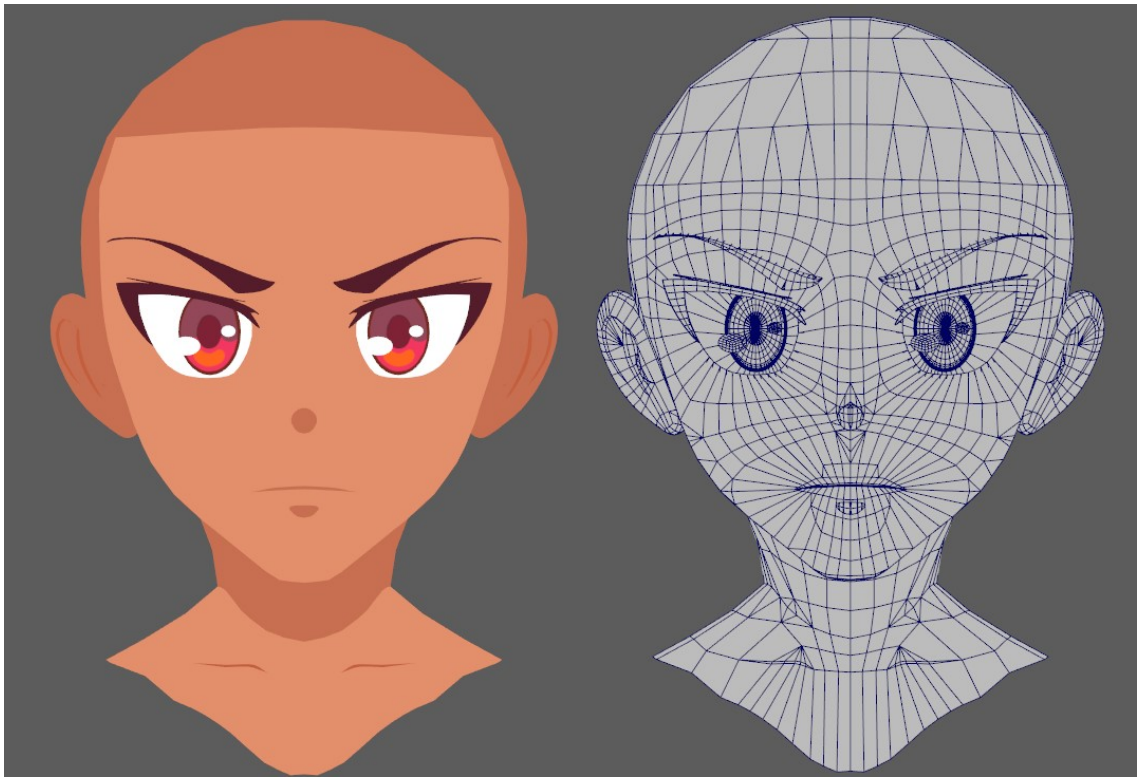
Kuva 12. Studio Triggerin Kill la Kill -sarjan (2013) Ryuko-hahmon ilmeitä erotettuna muusta hahmosta.

Alun perin suunnittelin hahmolleni monimutkaista kasvojen rigiä, jolla voisi matkia 2D-animaation ilmeitä. Se osoittautui tarpeeksi suureksi projektiksi omaksi opinnäytetyökseen. Tutkin Studio Triggerin hahmojen ilmeitä ja kasasin niitä kirjastoksi, jota minun oli tarkoitus käyttää rigin suunnitteluun. Yllä olevassa kuvassa 12 Ryuko-nimisen hahmon ilmeistä näkee, kuinka paljon 2D-animaatio venyttää kasvoja ja sen osia. Silmät ja suu muuttavat muotoaan pyöreiden ja terävien muotojen välillä kallistuen suuntaan tai toiseen korostaen tunnetilaa, joka hahmolla sillä hetkellä on. Valmis rigi käyttäisi suurta määrää blendshapeja (eli animoitavia geometrian muutoksia), jotka pitäisi suunnitella toisiinsa sopiviksi. Muun muassa leuan ja suun muotojen muutosten kanssa voi tulla vaikeuksia. Aion koettaa tehdä tämän rigin vielä opinnäytetyön jälkeen.

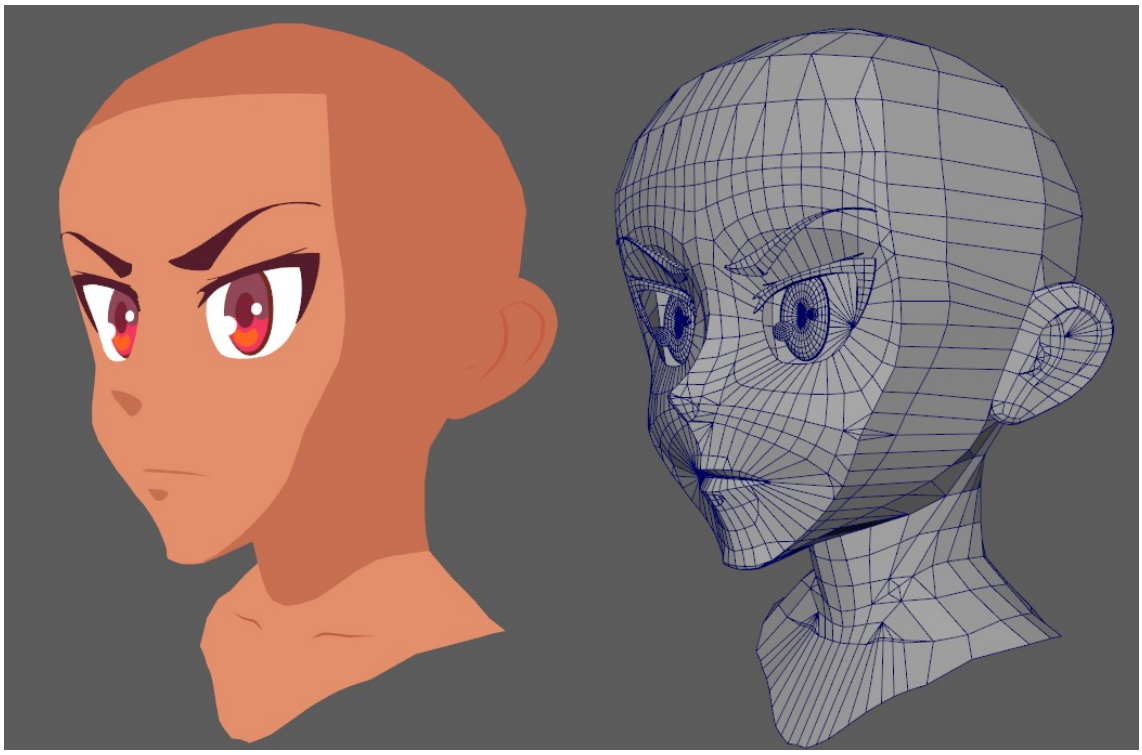
## 3.2 Mallintaminen

### 3.2.1 Pää

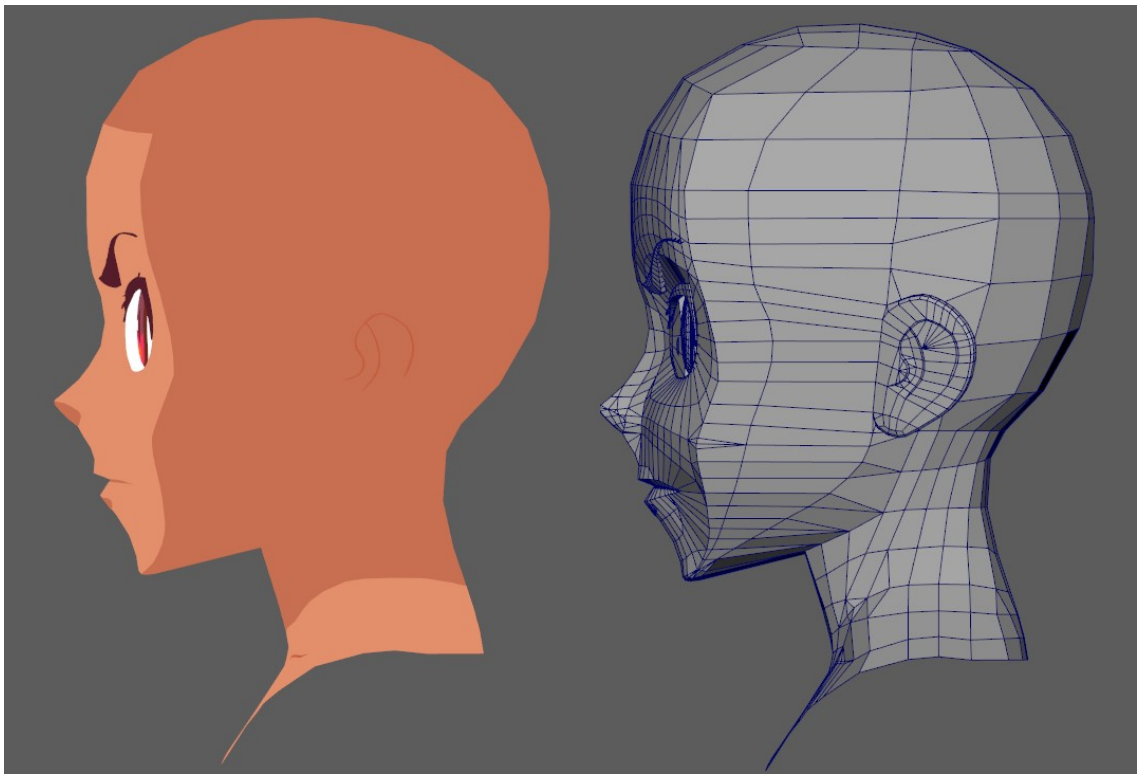
Aloitin mallintamisen hahmon päästä. Anime-tyylisten kasvojen muoto vaatii tietynlaista geometriaa näyttääkseen 2D-animaatiolta joka kuvakulmasta, ja koska käytin terävää varjostusta, ei geometrian tarkkuutta tarvita kuin siluetin muodostamiseen tai jonkin kasvojen piirteen tai varjojen näyttämiseen. Esimerkiksi silmien ympärillä ei ole juurikaan edgelooppeja (eli täyden ympyrän kulkevia geometriaviivoja), koska ne eivät silmien syvennyksestä näkyisi mihinkään suuntaan. Sama tilanne suun kanssa, paitsi että sivulta katsottuna leuka tarvitsi lisää kaarevuutta. Tällaisia geometrian tarkkuutta lisääviä kohtia ovat myös posket, nenä ja leuka. Vastakohtana ovat otsa ja kaula, joissa taas vähensin geometriaa, koska sitä ei kaulassa ja pään päällä tarvita niin paljon kuin kasvoissa.



Kuva 13. Suoraan edestä kasvot saa kohtalaisen helposti näyttämään samalta kuin piirustusversio.

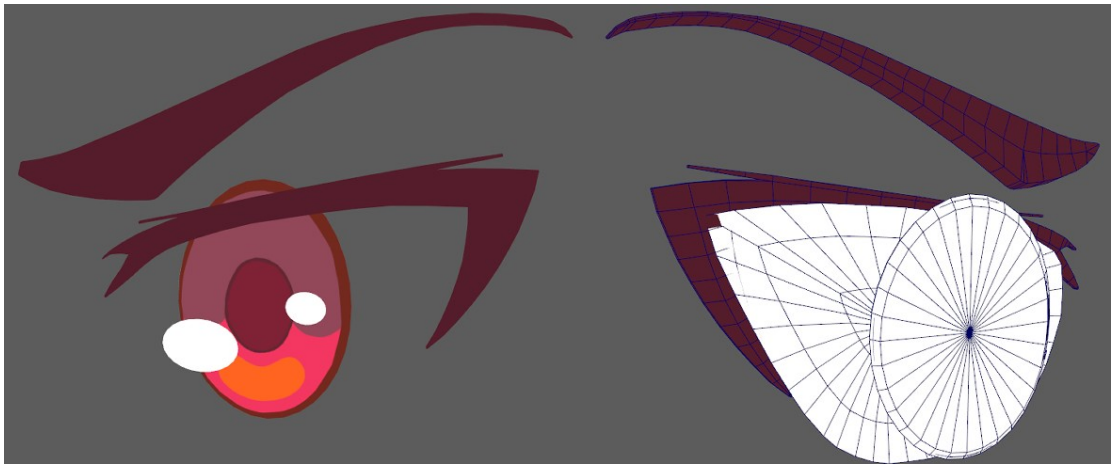


Kuva 14. Kasvot ovat kohtalaisen litteät, jotta ne näyttäisivät ¾ kuvakulmasta anime-tyylin mukaiselta.



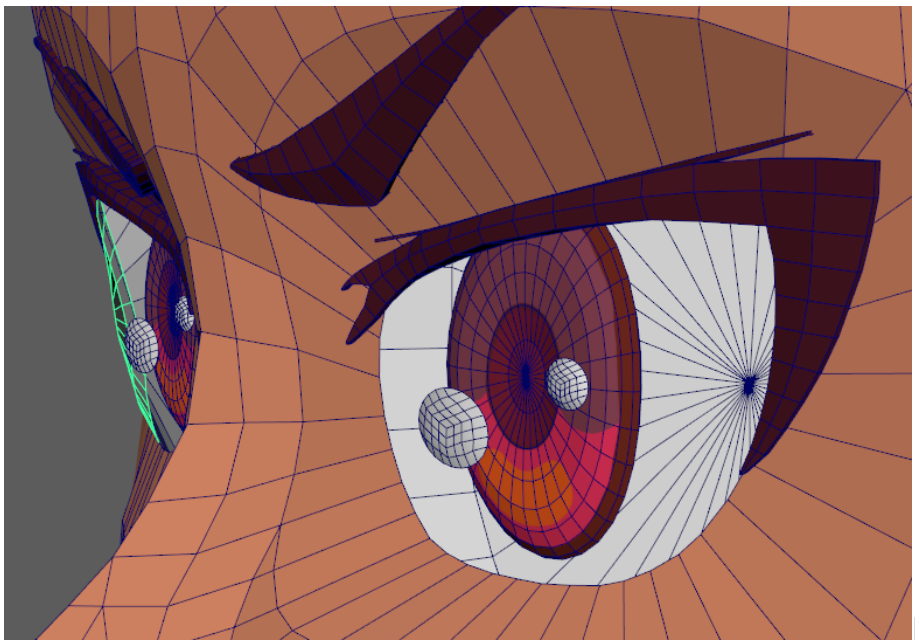
Kuva 15. Syvyys piti saada näyttämään hyvältä myös sivulta katsottuna. Silmien 3D-mallit tekevät tämän vaikeaksi, koska ne ovat pyöristetyt levyt ”mahdotto-man” muotoisessa kolossa normaalin pallon muotoisen silmän sijaan.

### 3.2.2 Silmät

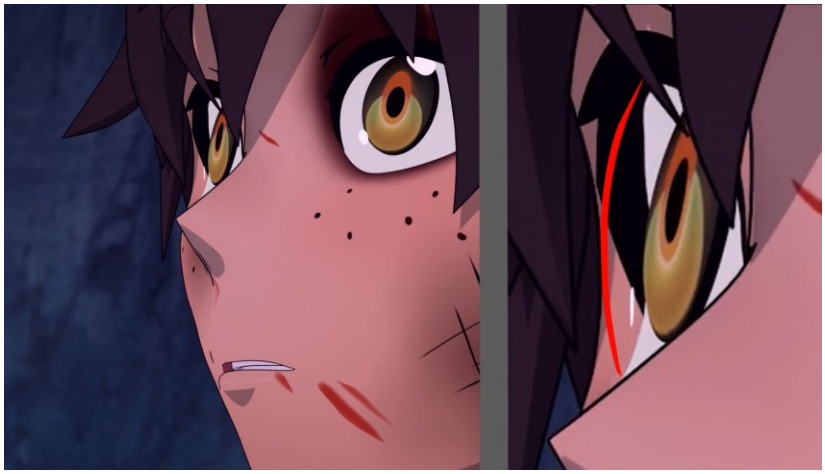


Kuva 16. Silmien osien 3D-mallit.

Hahmon silmät muodostuvat pään silmäkuoppien ympärille mallinnetuista ripsistä, kuoppien sisälle laitetuista teksturoiduista levyistä, silmien hohtopisteistä ja silmien valkuaisten ulkopintaa vastaavista levyistä. Tämä tekniikka on yleisessä käytössä hahmoilla, joiden silmät eivät ole palloja. Pallon muotoisia silmiä voi vain pyörittää, ja ne näyttävät siltä miltä pitääkin, mutta kun silmät eivät ole pyöreät, ei niitä voi järkevästi pyörittääkään. Siksi silmä on jaettu levyyn, jossa on pupilli ja iiris, ja valkuaiseen, joka on oikeasti vain valkoiseksi värjätty kuoppa päässä. Silmälevy on hieman pyöristetty, joten kun sitä liikuttaa kuopassa, niin silmä näyttää pyörivän.



Kuva 17. Silmät



Kuva 18. Oscarin silmän kaarevuuden puuttuminen sarjassa RWBY. Rooster Teeth. RWBY (2020)

Silmän valkuaisten ulkopintaa vastaava yksipuolinen levy ei ole kaikilla studioilla käytössä. En tiedä miksi, sillä se on kuitenkin yksinkertainen tapa välttää silmän hassun muodon näkyminen tietyistä kuvakulmista, niin kuin yllä olevassa kuvassa 18 RWBY sarjassa. Edellisellä sivulla olevassa kuvassa 17 valkuaislevy näkyy tätä kuvaa vastaavassa kulmassa vihreällä korostettuna silmän pyöreyttä matkimassa.

Varjojen saaminen silmän pinnalle osoittautui hankalaksi, mutta ei mahdottomaksi. Testasin valkuaisten levyn toiselle puolelle läpinäkyvän materiaalin laittamista, mutta koska Unreal renderöi saman objektin molemmat puolet samaan aikaan, se ei osannut päättää, kumman puolen levystä pitäisi tulla päälle. Tästä ongelmasta pääsee, kun käyttää kopiota valkuaislevystä, kääntää sen facenormalin ulospäin ja laittaa siihen läpinäkyvän varjotekstuurin ja alkuperäiseen levyyn valkoisen. Tämä varjo on vain yksinkertainen puoliksi musta ja puoliksi valkoinen tekstuuri, jota voi liikuttaa ylös ja alas. Ympäristön valot eivät siis ohjaa sitä, mutta sille voi tehdä animaatiorigin.

Kuva 19. Silmän yksinkertainen varjo.

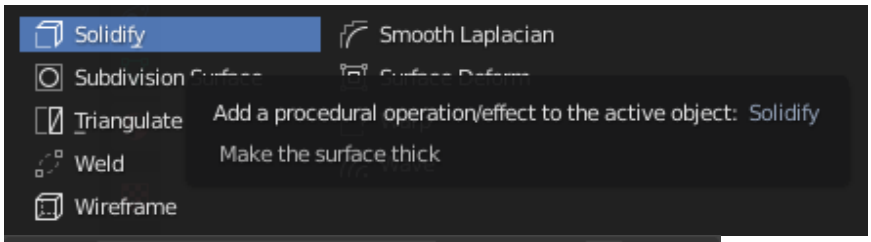
### 3.2.3 Ääriviivat

Hahmon ”ulkoiset” eli mallin muotojen ulkorajojen ääriviivat tein käyttäen ”inversed hull” -tekniikkaa. Siinä tehdään mallista hieman suurempi kopio, jonka normaalit käännetään sisäänpäin (tästä enemmän normalien editointi-luvussa).

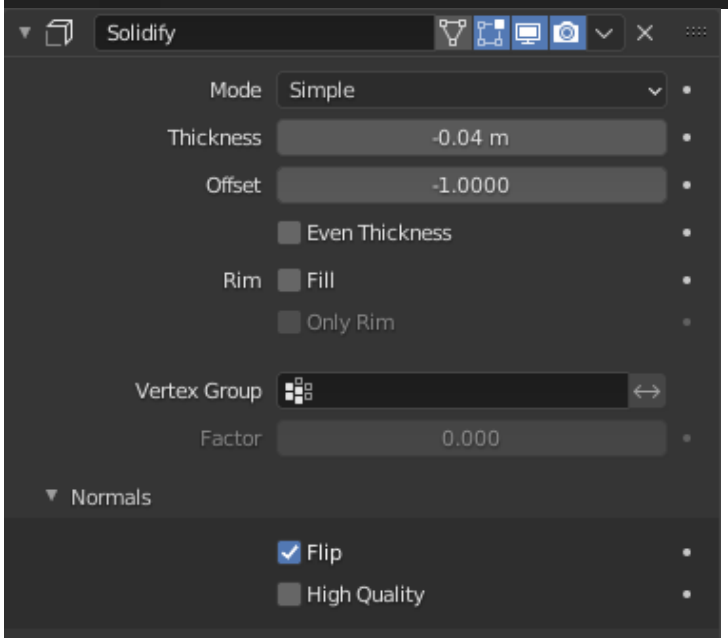
Kuva 20. Hahmon pää ja hiukset ennen ääriviivoja. Tein inversed hull -mallit Blenderin Solidify modifier -työkalulla. Sillä voi valita 3D-mallin ja antaa sille toisen kerroksen tietyn matkan päähän joko sisälle tai ulospäin skaalaten.

Vaikkei se ole välttämätöntä, otin itse kopion alkuperäisistä malleista ja tein siitä ääriviivamallin erillään alkuperäisestä mallista. Tämä oli mielestäni selkeämpi tapa muokata niitä, koska niin kuin seuraavan sivun kuvasta 23 näkee, kahden päälekkäisen mallin geometriat voivat helposti mennä sekaisin, jos toista kerrosta ei saa piilotettua kun editoi toista.

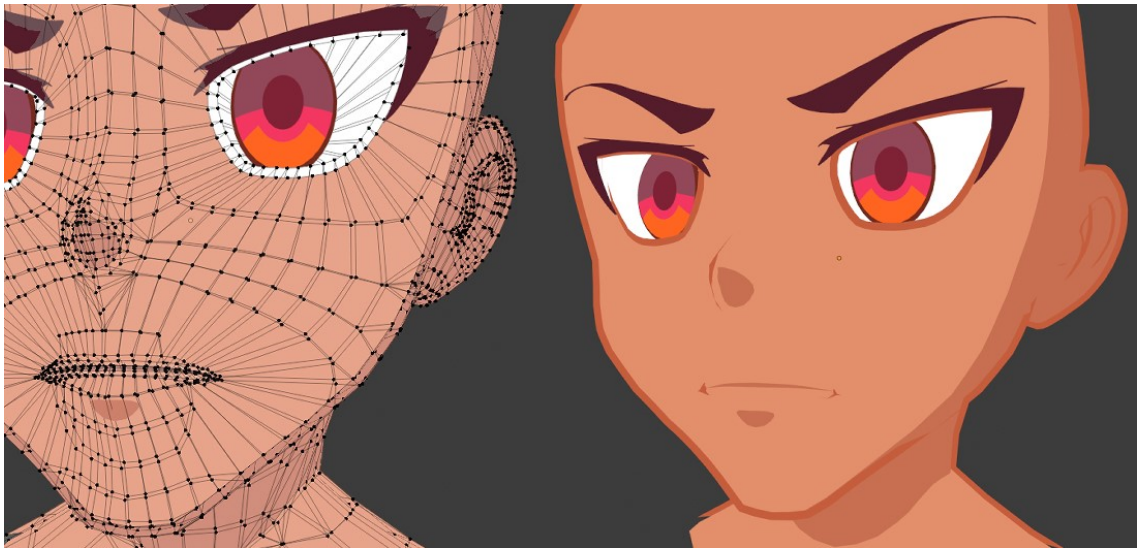
Kuvassa Solidify multiplierin asetuksista tärkeimmät asetukset ovat Thickness, joka säätää etäisyyttä johon uusi kerros mallia tehdään, Rim, joka yhdistää vanhan ja uuden mallin (tämä kannattaa ottaa pois päältä, koska tulemme poistamaan vanhan mallin ja pitämään vain uuden kerroksen) ja Normals: Flip, jolla uuden kerroksen face normal käännetään sisäänpäin.




Kuva 21. Blenderin Solidify modifier



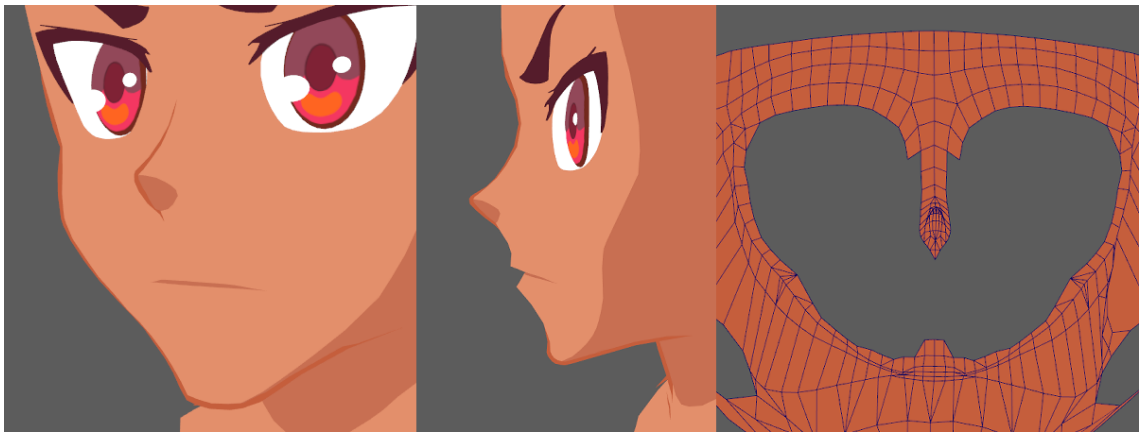

Kuva 22. Solidify multiplierin asetukset



Kuva 23. Muokkaamaton hull-malli.

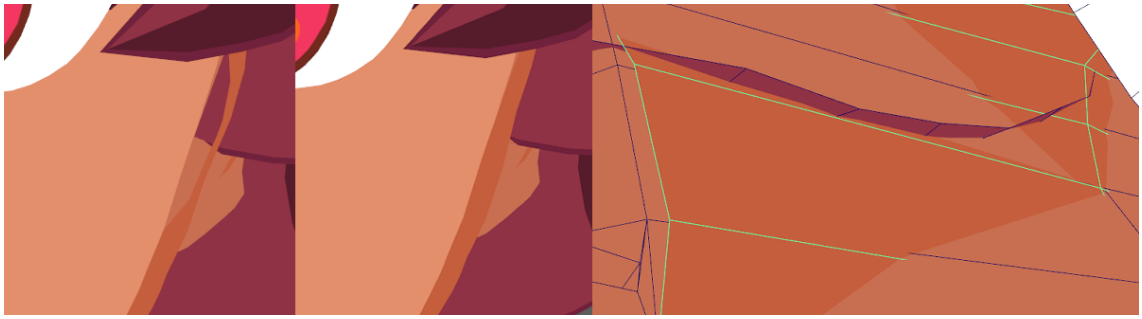
Kun modifier on hyväksytty malliin, siihen kannattaa laittaa heti väri, jonka haluaa ääriviivoille, niin näkee mitä kohtia pitää korjata. Sitten poistetaan mallin alkuperäinen sisempi kerros. Tässä kohtaa on siis alkuperäinen kasvojen malli ja siitä suurennettu hull-versio, jonka facenormal on sisäänpäin ja joka on kokonaan ääriviivojen värinen. Niin kuin kuvassa 23 näkyy, laajennettu malli

käyttäytyy oudosti silmien, nenän ja huulten kohdilla. Poistin hull-mallin kasvojen keskiosan lähes kokonaan, koska en halunnut ääriviivoja näkymään muualle kuin kasvojen rajoille, leukaan ja nenään.



Kuva 24. Ääriviivat kun hull-mallia on muokattu ja suurin osa kasvojen alueesta poistettu.

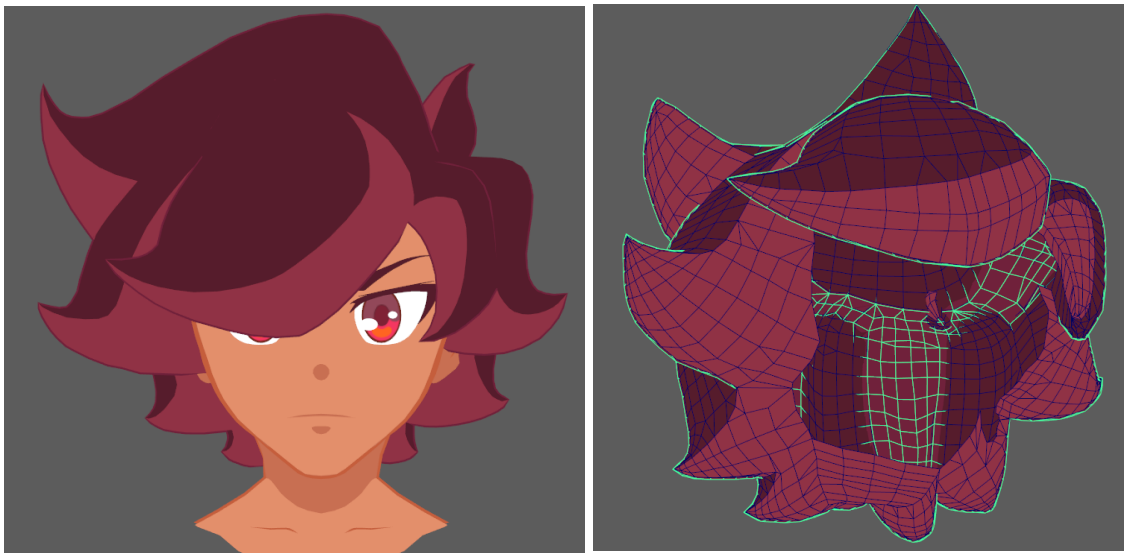
Poistin myös korvat, koska tiesin niiden jäävän lähes kokonaan hiusten alle. Kasvojen ja hiusten raja-alueen kanssa sai olla varovainen, koska inversed hull -tekniikalla tehdyt ääriviivat näyttävät huonolta jos ne loppuvat kuin seinään.



Kuva 25. Vasemmalta oikealle: 1) Editoimaton laajennettu pään malli menee suoraan läpi korvista ja hiuksista. 2) Editoitu malli näyttää hyvältä posken ääriviivalta. 3) Korjattu ääriviivojen malli korostettuna vihreillä viivoilla tulee pään mallista läpi sen sisäpuolelle, ennen kuin se menee läpi hiuksista tai korvista.

Tällaisia manuaalisia korjauksia pääsee tekemään lähes kaikkiin kohtiin, joissa ääriviivojen tulisi olla kahden asian rajalla, sillä automaattinen 3D-mallin laajentaminen siirtää sen kokonaan irti alkuperäisestä luoden rakoja, joista näkee läpi.

### 3.2.4 Hiukset



Kuva 26. Hiusten 3D-malli.

Niin kuin mainitsin designluvussa, hiusten mallintamisen aikana minulle selvisi, kuinka hankalaa ne on saada näyttämään hyvältä joka suunnasta. Tein artistisen päätöksen jättää varjot hiuksista pois, sillä tasaisen värisiin hiuksiin on helppo piilottaa pinnan muotoja niissä kulmissa, joissa niiden ei tahdo näkyvän. Hiusten ääriviivojen muokkaus sujui enimmäkseen siirtämällä kohtia hull-mallista hiusten sisälle pois näkyvistä (vihreällä korostetut alueet yllä olevassa oikean puoleisessa kuvassa). Minimaaliset ääriviivat paljastivat mahdollisimman vähän hiusten kolmiulotteisuudesta ja saivat ne näyttämään niin 2D:ltä kuin ensimmäisellä yrityksellläni osasin tehdä.

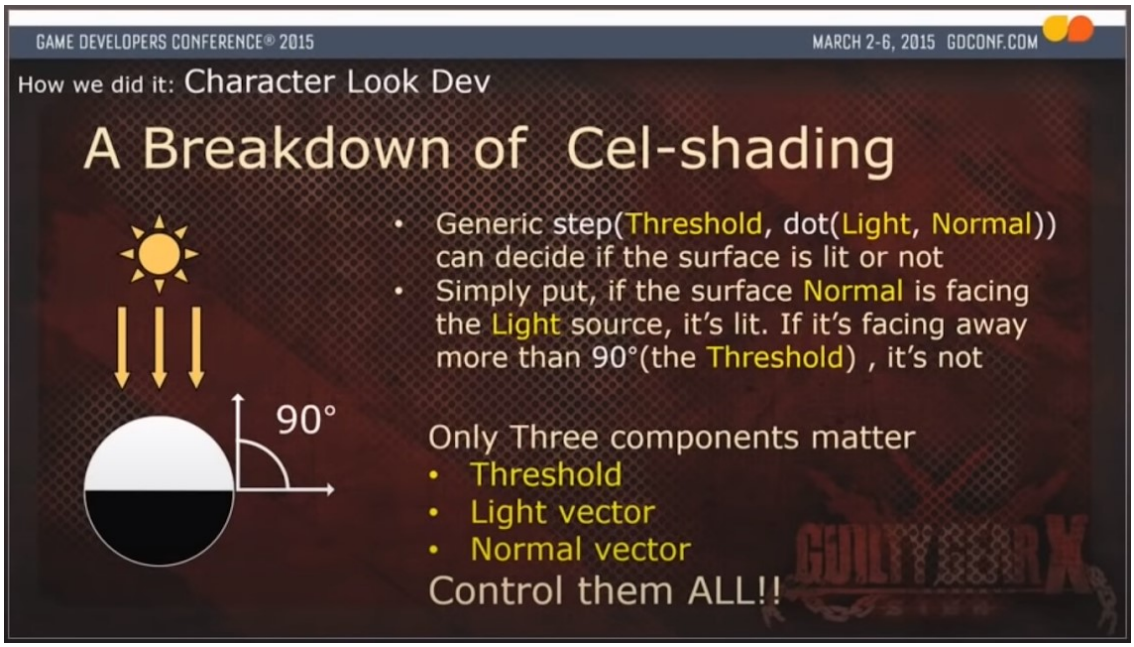
Tein suunnilleen saman prosessin kaikille hahmon malleille, joille ääriviivat halusin. Kerron seuraavassa luvussa vielä normaalien editointiin liittyvistä mallintamistekniikoista, joilla saa ääriviivat käyttäytymään niin kuin tekijä haluaa. Raskaaksi käsityöksi tämä prosessi tulee, mutta se on hinta, jonka tarkasta haluamastaan tyylillisestä lopputuloksesta joutuu usein maksamaan. Ääriviivat voi tehdä nopeammallakin tavalla renderöintivaiheessa, mutta siitä, ja syistä miksi en sitä tapaa käyttänyt, kirjoitan lisää renderöintiluvussa.

Kuva 27. Hahmon valmis malli inversed hull -ääriviivoineen Blenderissä.

## 3.3 Normaalien muokkaus

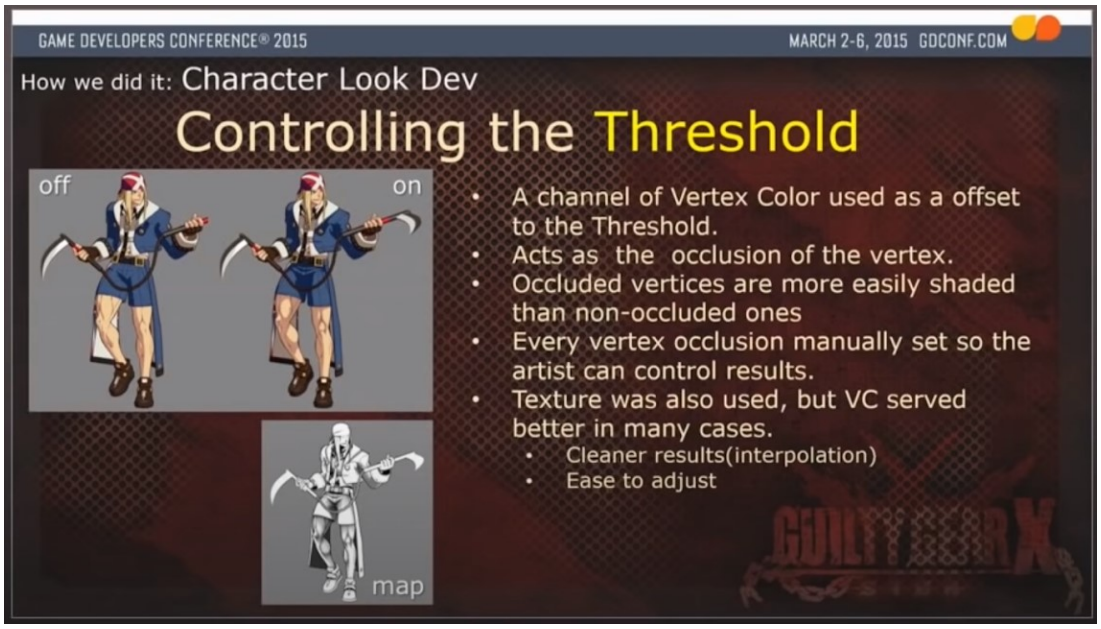
### 3.3.1 ”A Breakdown of Cel-shading”

Vuoden 2015 GDC tapahtumassa Arc System Works -pelistudion tekninen artisti Junya C Motomura kertoi pelin Guilty Gear Xrd tekniikoista saada aikaan 2D-anime-tyylinen renderöinti 3D-taistelupelissä. Motomura puhui valon vektorista (Light vector), eli suunnasta johon valo osoittaa, normaalien vektorista (Normal vector), eli suunnasta johon mallin pinta osoittaa, ja threshold-arvosta, jolla säädetään kuinka suuri aste ero näiden vektorien välillä voi olla niin, että se vielä lasketaan valaistuksi. (Motomura 2015)




Kuva 28. Junya C Motomuran selitys cel-shading tekniikasta. GDC (2015)

Mitä enemmän näihin kolmeen pystyy vaikuttamaan, sen paremman lopputuloksen cel-shading-varjostustyylillä saa. Paras tulos saadaan muokkaamalla jokaisen verteksin eli geometrian pisteen normaaleita ja threshold-arvoa. Arc System Works käytti vertex coloria eli geometrian pisteisiin tallennettua väriarvoa antamaan eri threshold-arvon jokaiselle verteksille. Näin jokainen pinta voidaan valaista juuri niin kuin 3D-artisti haluaa, vaikka valo ei realistisesti käyttäytyisikään niin kyseisen muodon kanssa.




Kuva 29. Guilty Gear -hahmo esimerkkinä threshold-arvon käytöstä. GDC (2015)

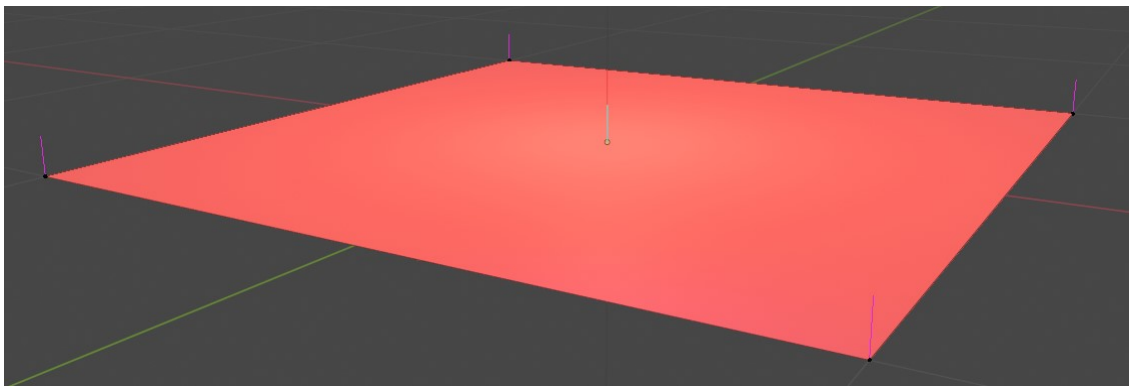
Motomura näytti esimerkkinä vertex colorilla muokatuista varjoista, kuinka paikat, joihin valo ei loista yhtä helposti kuin muualle, voidaan säätää alhaisella threshold-arvolla olemaan lähes aina varjossa. Kuten jo viime luvussa mainitsin, varjojen paikan voi säätää normalien avulla ainoastaan jos, verteksit on mallinnettu sinne minne varjot tahtoo.

Kuva 30. Arc System Worksin pelisarjan Guilty Gear hahmo Sol Badguy ilman vertex colorin muokkaamia vector threshold -arvoja (vasen) ja niiden kanssa (oikea). GDC (2015)

### 3.3.2 Face- ja vertex-normaalit

Näitä asioita testaillessani Blenderillä tajusin ensimmäistä kertaa, että normaaleitahan on kahta eri tyyppiä, on face normal, eli mihin suuntaan taso osoittaa ja vertex normal, eli mihin kulma/reuna osoittaa. Yleensä kun haluaa muokata 3D-mallin normaaleita, se tehdään normal mapillä, eli kuvatiedostolla, joka kertoo pikselien väreillä, mihin suuntaan minkäkin kohdan pinnasta tulee heijastaa valoa. Verteksien normaaleilla ei saa yhtä paljon yksityiskohtia aikaan, mutta ne tekevät saman asian. Normal mapit usein beikataan yksityiskohtaisesta mallista yksinkertaisempaan. Näin saadaan lisää yksityiskohtia kohtalaisen automaattisesti kevyempään versioon alkuperäisestä mallista. Siksi en ole perehtynyt verteksien normaalien muokkaamiseen aikaisemmin. Sitä ei Mayassa juurikaan voi tehdä, mutta Blenderissä työkaluja tähän löytyy paljonkin. Voit valita tason ja kääntää sen toiseen suuntaan (tämä onnistuu Mayassakin) tai valita verteksin ja kääntää sitä mihin suuntaan vain. ”Mihin suuntaan vain? Mitä jos käännän sen 180 astetta eli mallin sisäpuolelle?” Löysin puolivahingossa tavan tehdä valaistuja ääriviivoja.

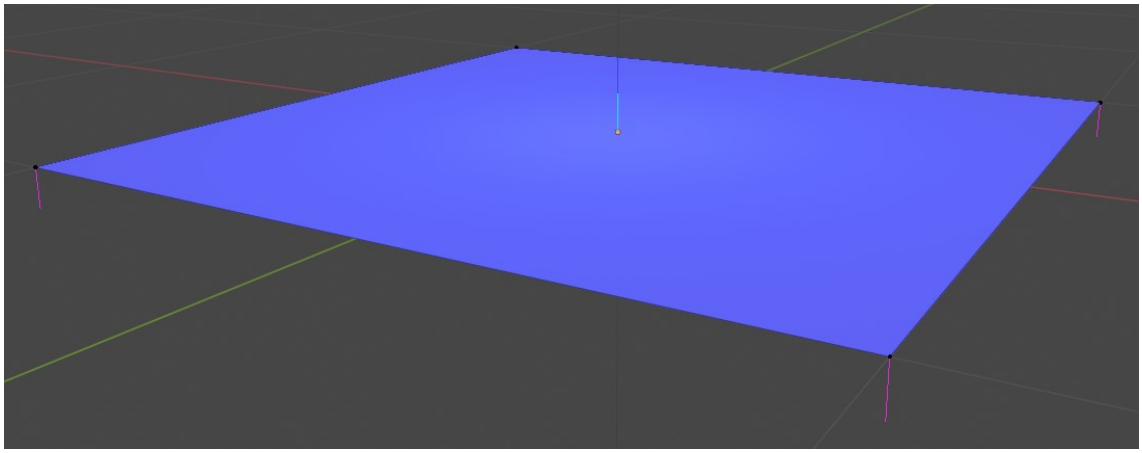
Innostuin tästä löydöstä paljon, koska en muista ikinä nähneeni missään videopelissä valaistuja ääriviivoja. Selitän löytämäni tekniikan muutamalla kuvalla käyttäen kahta levy-3D-mallia ja kahta eriväristä lamppua.



Kuva 31. Muokkaamaton levy.

Kuvassa 31 näkyy levyn face normal sinisellä viivalla sen keskellä ja vertex normal liiloilla viivoilla, kaikki osoittamassa ylöspäin. Levy on punainen, koska

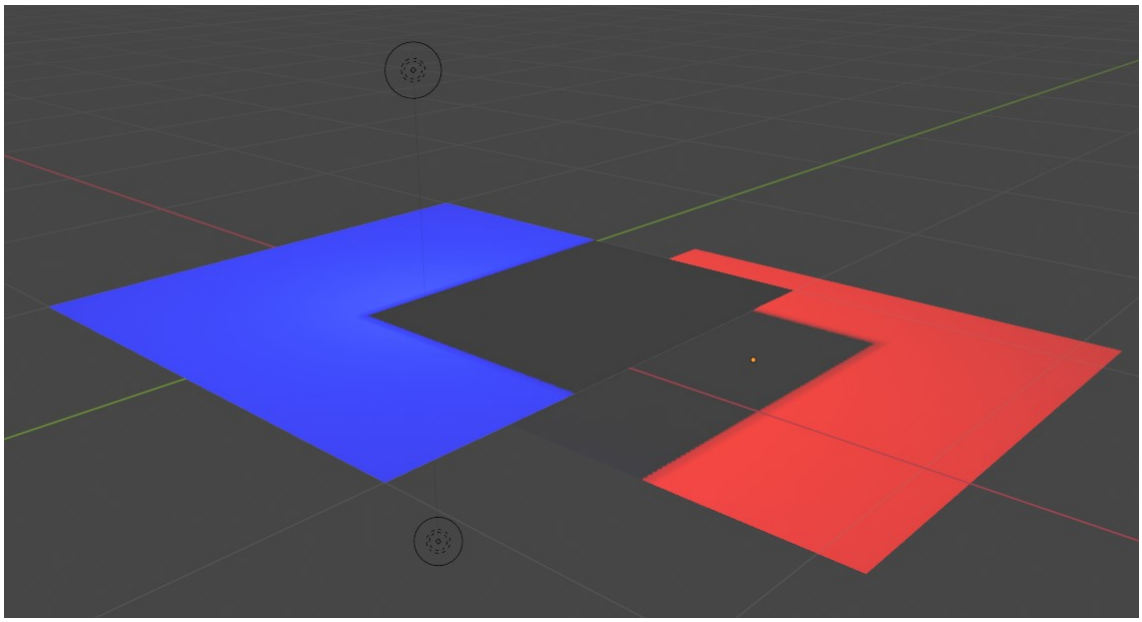
sen yläpuolella oleva punainen lamppu valaisee sen. Levyn alapuolella oleva lamppu valaisee sen alapuolen.



Kuva 32. Muokattu levy.

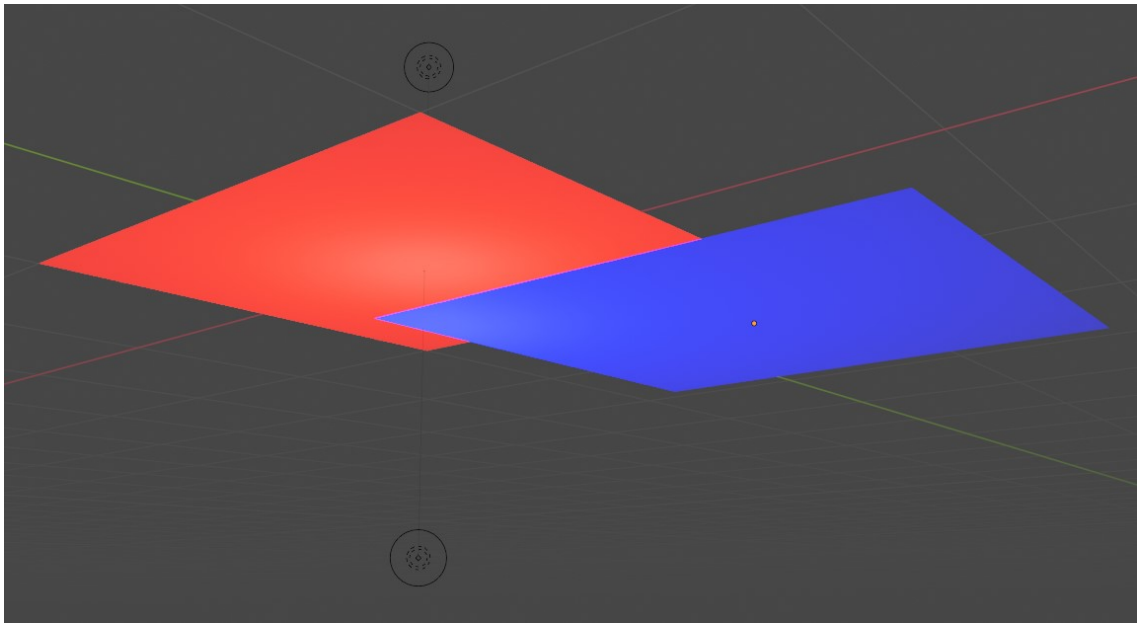
Käänsin liilat vertex normalit alaspäin face normalin pysyessä ylöspäin. Nyt levy on sininen päällipuolelta mutta punainen alapuolelta, vaikken ole koskenut lamppuihin.

**Face normal tarkoittaa siis suuntaa, johon valo heijastetaan ja vertex normal suuntaa, josta valo vastaanotetaan.**



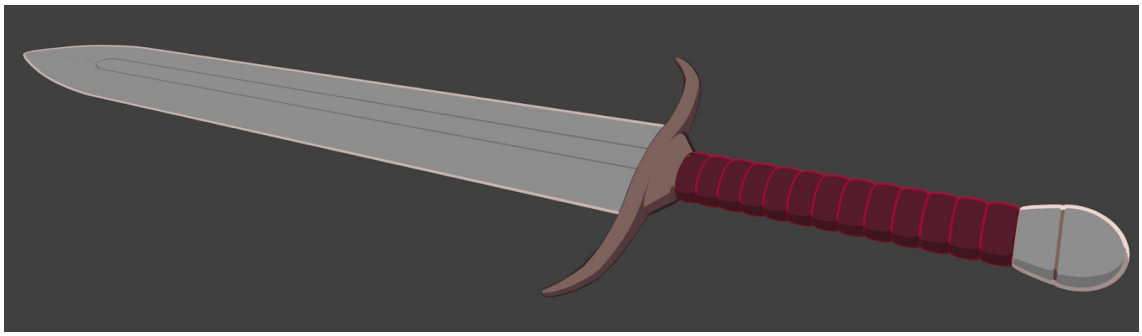
Kuva 33. Molemmat levyt yläpuolelta.

Lisäsin muokatun levyn alapuolelle muokkaamattoman levyn. Kuvassa 33 ensimmäinen levy varjostaa ylhäältä loistavaa punaista lamppua jättäen varjon alapuolella olevaan levyyn, joka taas varjostaa alapuolelta loistavaa sinistä lamppua jättäen varjon yllä olevaan levyyn. Molemmat varjot näkyvät päällipuolella, mutta eivät alapuolella

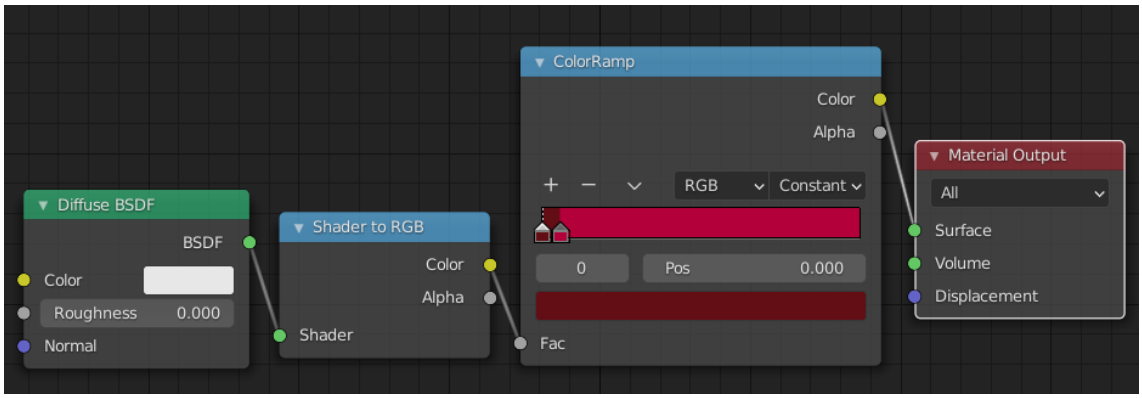


Kuva 34. Molemmat levyt alapuolelta.

Hämmentävä, mutta täydellinen tekniikka haluamaani tarkoitukseen, testasin sitä ensin hahmoni miekkaan.

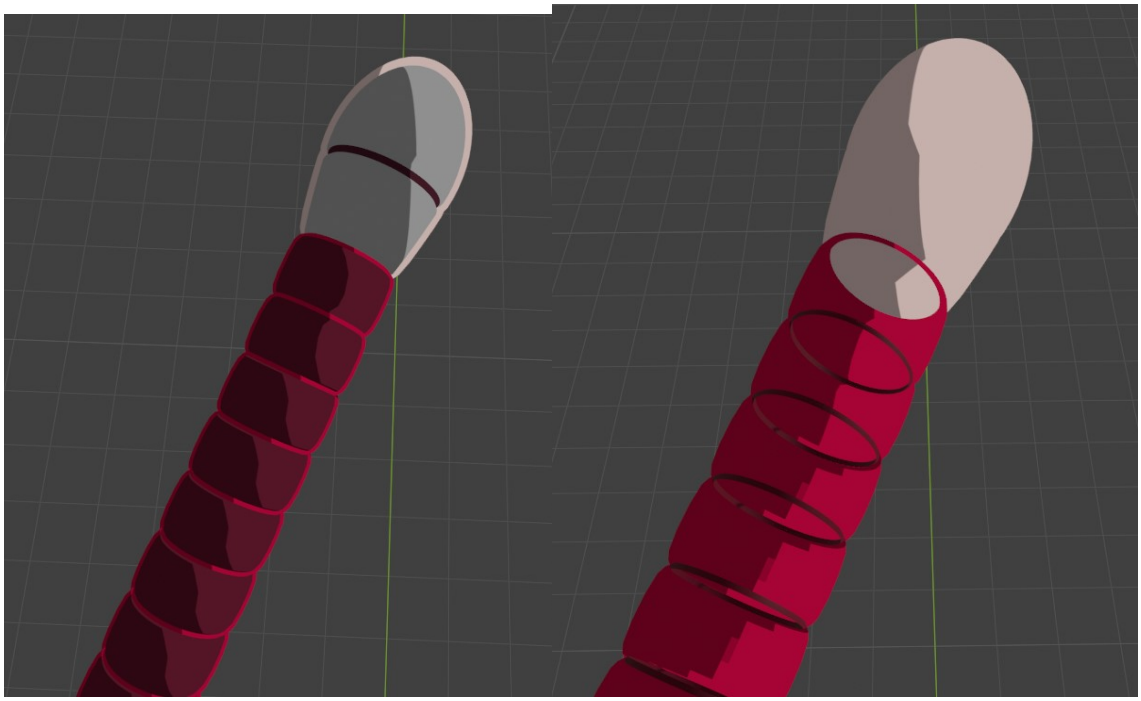


Kuva 35. Valmis miekan malli. Ääriviivojen valaisu näkyy tästä kulmasta vain kahvassa.




Kuva 36. Cel-shaded materiaali Blenderissä.

Käytin Blenderissä yksinkertaista materiaalia, joka muuttaa Diffuse shaderin pehmeät varjot ColorRamp nodella teräviksi. ColorRamp ajaa tässä tapauksessa saman asian kuin thresholdin muokkaaminen, paitsi että se on sama arvo koko mallille, jossa tämä materiaali on. Sama matematiikka vähän eri näköisillä noodeilla toimi myös Unrealissa, siitä lisää renderöintiluvussa.



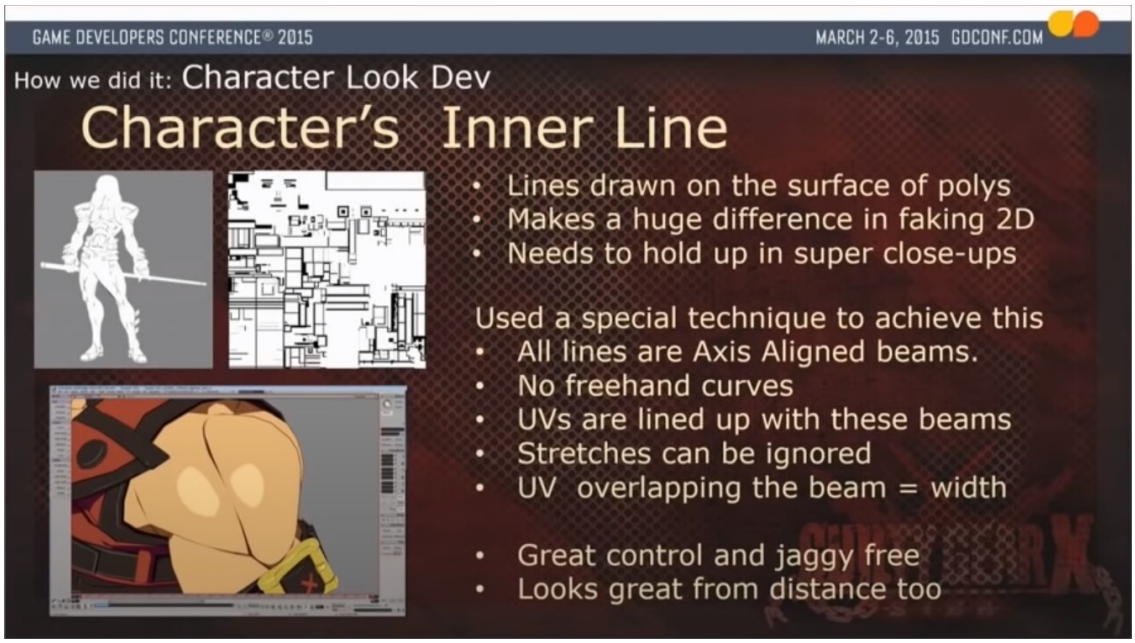
Kuva 37. Vasemmalla on miekan kahvan alkuperäinen malli sen hull-mallin sisällä. Oikealla pelkkä hull-malli vertex normalien muokkaamisen alkupuolella (osa varjoista on vielä palikkamaisia). Vaikka kuvassa näkyykin hull-mallin sisäpuoli, se ottaa valoa mallin ulkopuolelta. Kyseinen muokkaus on sama prosessi kuin sille levylle, jonka vertex normalit käänsin.

Niin kuin mainitsin mallintamista käsittelevässä luvussa, en ottanut varjojen kartoittamista huomioon ennen hahmon mallintamista, joten en lähtenyt yrittämään koko hahmon verteksien thresholdien muokkaamista. Seuraavalla hahmon mallintamiskerralla uusi yritys!

## 3.4 UV ja teksturointi

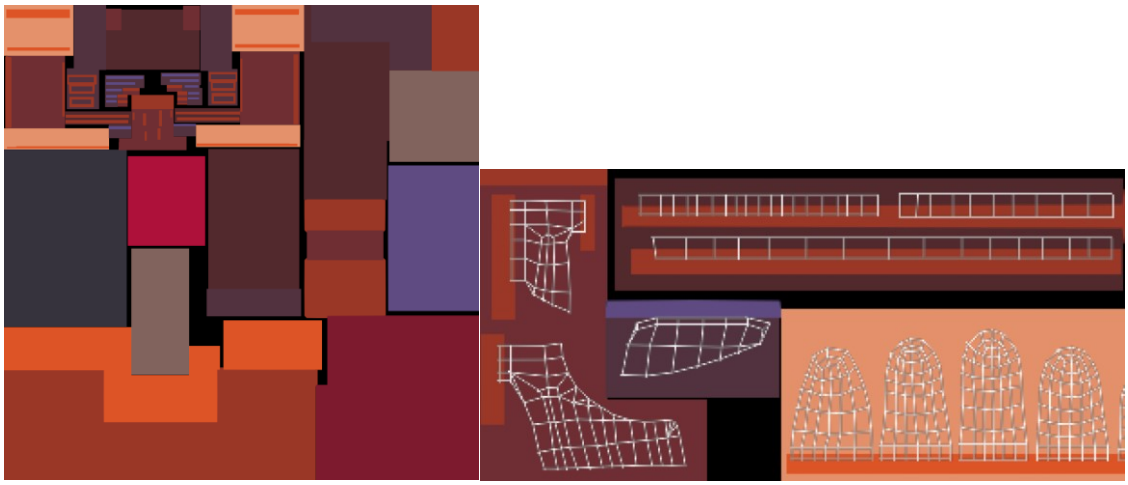
Projektin helpoin osuus oli teksturointi, sillä valitsemani tyylisuunta käyttää yksinkertaista väritystä. Silmiä lukuun ottamatta kaikki hahmon tekstuurit ovat vain väripalikoita, joiden sisään hahmon uv alueet on laitettu. Käytin tässäkin tapauksessa esimerkkinä GDC:ssä Arc System Worksin esittelemää tekniikkaa. He määrittelivät varjojen värit toisella tekstuurilla, jolla normaali väritekstuuri kerrotaan, jotta tummemmat varjovärit saadaan aikaan. (GDC 2015)


Kuva 38. Tint-tekstuurien käyttö. GDC (2015)

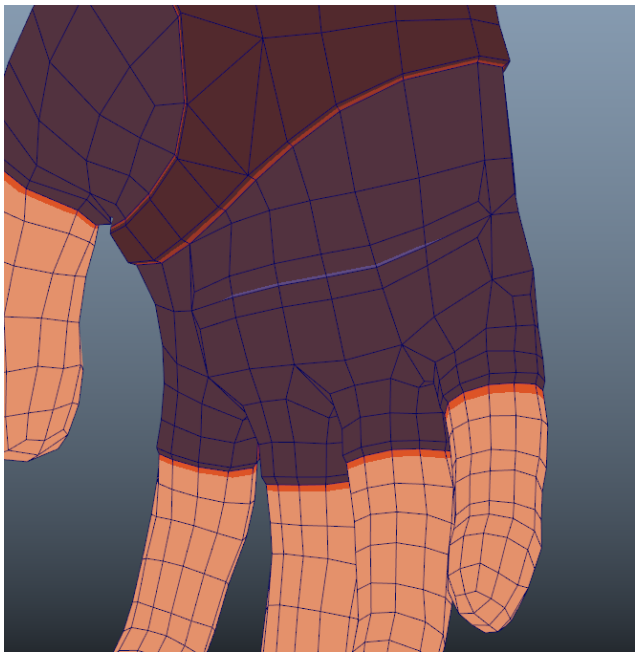



Kuva 39. Arc System Works teki hahmojensa mallien ”sisäiset” ääriviivat siirtämällä uv-alueiden reunojen verteksit tekstuurin värialueen yli mustalle alueelle. GDC (2015)

Näin he välttyivät pikselöityneiltä viivoilta, joita tekstuuri piirustukset liian pienillä resoluutioilla aiheuttavat. Kun tekstuurin viivat menevät suoraan pikselien mukaan viivat eivät sumennu.



Kuva 40. Vasemmalla kuva hahmoni kehon tekstuureista, oikealla sormien ja käden uv-alueita, jotka on venytetty ääriviivavärien alueille.



Kuva 41. Käden malli ääriviivoineen teksturoituna.

Kuva 42. Silmän tekstuuri, ainoa maalattu, ei palikoilla väritetty tekstuurialue tässä hahmossa.

Tein varjojen värit Unrealin materiaalissa vain kertomalla tekstuurit suoraan numero arvoilla Arc System Worksin käyttämän varjoväri tekstuurin sijaan, mutta aion sellaisenkin tämän projektin jälkeen tehdä.

## 3.5 Rigaaminen ja animaatio

Totesin aika nopeasti projektia tehdessäni, että 2D:ltä näyttävän 3D-hahmon rigaaminen ja animointi ovat molemmat tarpeeksi isoja aiheita omille opinnäytetöilleen, enkä siis mennyt kovinkaan pitkälle rigissäni, enkä juuri animoinut hahmoa tätä opinnäytetyötä varten. Kirjoitan nyt kuitenkin suunnista, joihin rigiin tai animointiin keskittyvä opinnäytetyö voisi lähteä tavoittelemaan samaa tyyliä, johon itse tällä projektilla tähtäsin.

Ensinnäkin rigin osalta tilanne on toisaalta yksinkertainen, mutta toisaalta haastava. Jo pariin otteeseen mainitsemani GDC 2015 Arc System Worksin esittely tekniikoista tehdä 2D:n näköistä 3D-pelianimaatiota kertoo Guilty Gear -hahmojen rigien käyttävän noin 400–600 luuta, joita kaikkia animoidaan käsin translaatiolla, rotaatiolla ja skaalaamalla. ”Freedom over automation” (Motomura 2019) on periaate, jolla vaikuttaa pääsevän lähimmäksi piirretyn näköistä lopputulosta. Artistin täytyy pystyä kontrolloimaan kaikkea. Hiuksia tai kangasta ei simuloida

heilumaan tuulessa, keyfreimien (pääasentojen) välistä interpolaatiota ei jätetä automaattiseksi, eikä framerate pysy tasaisena. Kaikki rigin osat tottelevat visiota, joka animaattorilla on liikkeestä.

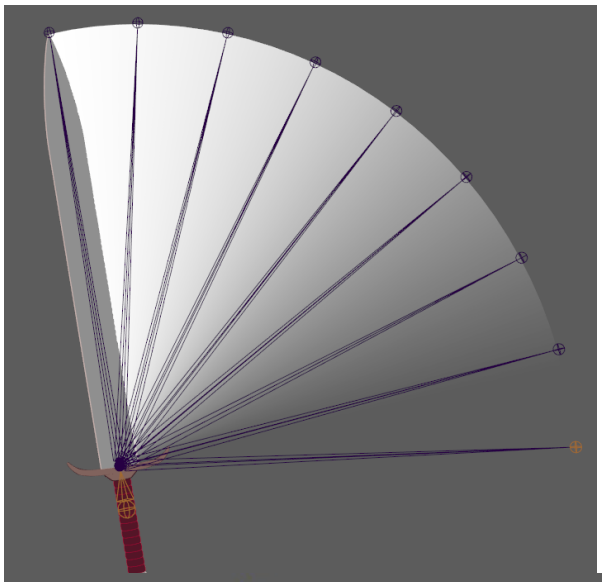

Kuva 43. Guilty Gear -hahmojen rigit. GDC (2015)

Toisaalta tämä tarkoittaa sitä, ettei hahmon osia tarvitse simuloida tai muuten tehdä monimutkaisia automaatioita rigiin, minkä luulisi helpottavan rigaajan työtä, mutta toisaalta se, että kaikkea pitää pystyä kontrolloimaan mielekkäästi ja hyvän näköisesti vaikeuttaa työtä. Mitä kaikkea sinun projektisi vaatii? Riittääkö se, että kankailla ja hiusmassoilla on omat luuketjut vai tarvitsetko jotain enemmän? Niin kuin opinnäytetyön alussa mainitsin, 2D-animaatiossa ei yleensä käytetä motion bluria, joka tekisi liikkeistä realistisen sulavia, koska 2D-animaatiosta halutaan napsakkaa ja selkeää. Sen sijaan 2D-animaatiossa käytetään smearing nimistä tekniikkaa, jossa esimerkiksi heiluvan käden jälkikuva seuraa sitä smear-efektien saattelemana. (Snyder 2019)

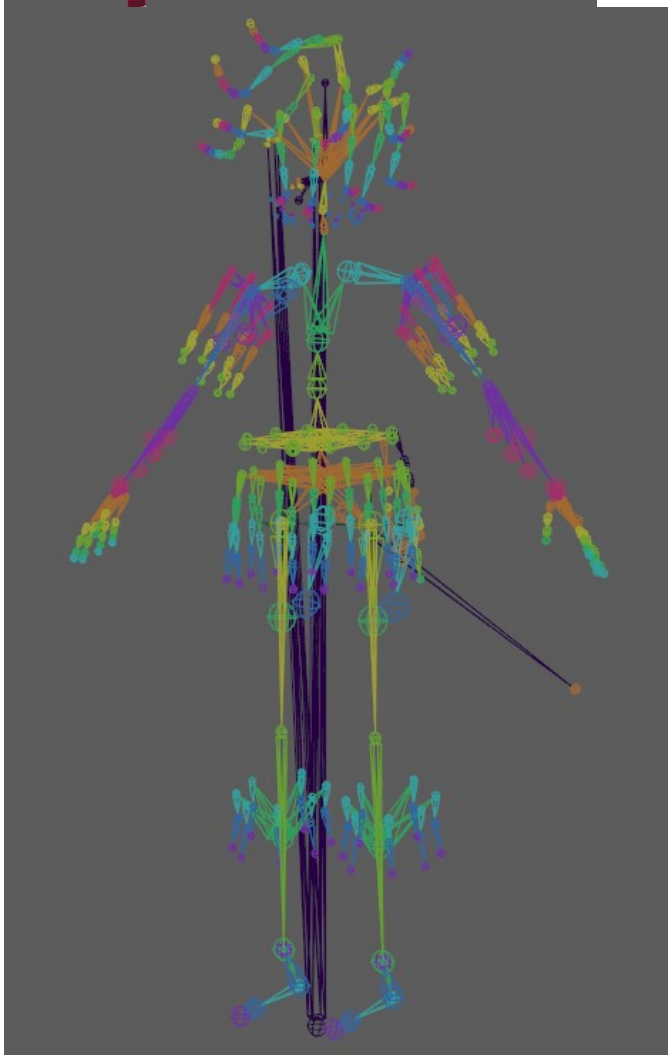
Kuva 44. Smear-efekti Spider-Verse -elokuvassa. (Snyder 2019)

Animaatioelokuvassa tämä voidaan renderöidä tai piirtää kokonaisen renderöidyn kuvan päälle, mutta 3D videopelissä smear-efektin osien, olivat ne sitten huitomis vauhtiviivoja tai lisäkäsiä, täytyy olla osa hahmon rigiä, jotta animaattori voi niitä ohjata.

Tähän projektiin tekemäni rigi ei ole aivan Guilty Gear Xrd:n tasolla, mutta siinäkin on kyllä 385 luuta (488 jos laskee myös luut, joita ei tuoda pelimoottoriin asti). Animaatiotestiä tehdessäni totesin, että määrää voisi vähentää etenkin vaatteiden luuketjuissa. Luita voisi myös lisätä jos tahtoisi lisää kontrollia mm. raajojen deformaatioon. Tyylistä riippuen animaattori voisi haluta taivuttaa käsivarsia pyöreästi, tai jopa lonkeromaisesti, ja sitä ei voi normaalilla luumäärällä tehdä.



Kuva 45. Tein hahmon miekalle simppelin motion blur -rigin. Tätäkin voisi viedä pidemmälle ja soveltaa smear-efektiksi asti miekalle ja myös kehon osille venyttäen osaa mallista viuhkamaisesti samoin kuin edellisen sivun Spider-Verse kuvassa 44.



Kuva 46. Hahmon rigin kaikki luut. Tein luultavasti liian monta kangas luuta ja hiuksistakin olisi voinut karsia muutaman.

Guilty Gear Xrd pelin rigit voivat olla raskaita useamman sadan luun rigejä, koska kyseessä on kahden pelaajan sivulta kuvattu taistelupeli, joten siinä ei ole ruudulla samanaikaisesti niin montaa hahmoa, että rigien raskaus haittaisi pelin pyörimistä.

Hyvä esimerkki 2D-animaatiota jäljittelevästä 3D-hahmoanimaatiosta, jota kannattaisi rigin kehitystä varten lähteä tutkimaan, on toinen Sonyn tuottama elokuva(sarja) Hotel Transylvania. Siinä hahmot venyvät ja vaihtavat muotoaan

erittäin ilmeikkäästi 3D- ja 2D-henkisten asentojen ja ilmeiden välillä.

Kuva 47. Sony. Hotel Transylvania 3: Summer Vacation (2018)

Kuva 48. Wit Studio. Attack on Titan (2019)

Hahmon liike ei ole ainoa asia, joka vaikuttaa siihen miltä animaatio näyttää. 2D-animaatiossa voidaan mm. vääristää kameran linssiä eri pituiseksi eri etäisyyksille. Esimerkiksi yllä olevassa kuvasarjassa 48 anime-sarjan Attack on Titan kohtauksessa hahmo liukuu kattoa pitkin alas, ja hänen jalkansa venyvät kuin kyseessä olisi lyhyt, melkein kalansilmälinssi. Samalla loput kehosta säilyy

venymättömänä, puhumattakaan taustasta, jonka linssi pysyy suunnilleen samana kohtauksen ajan.

3D-animaatiossa voisi toki renderöidä useamman kuvan ja kompositoida ne päällekkäin, mutta 2D:llä pystyy vääristelemään kuvaa kerralla niin paljon kuin haluaa. Tämän voisi kiertää 3D-hahmon rigillä niin, että hahmoa voi skaalata mielensä mukaan, jotta se näyttäisi vääristyvän kameran linssin mukaan. Tämä tosin vaatisi kameran vahvaa kontrollointia. Tästä päästään yhteen suurimmista ongelmista 3D-pelien 2D:n näköisiksi tekemisessä:

**kuinka paljon kameran kontrollia voidaan viedä pelaajalta artistisen näkemyksen mielekkääseen toteutukseen, ennen kuin pelaajasta alkaa tuntua enemmän siltä, että hän katsoo elokuvaa, kuin että hän pelaisi peliä?**

Jos haluaa saada samanlaisen katolla liukumisanimaation tehtyä pelimoottorissa kuin edellisen sivun kuvassa 48, se täytyisi tehdä juuri tietystä kamerakulmasta, sillä hahmon venymisen animaation säätäminen kameraan sopivaksi reaaliajassa mihin tahansa mahdolliseen kulmaan ei kuulosta järkevältä. Mitä tarkempaan taiteelliseen näkemykseen kohtauksella ja liikkeellä pyrkii, sen enemmän kontrollia on otettava kaikesta. ”Freedom over automation” (Motomura 2019) ajattelutapana pelin tyyliä luodessa muuttuu ”Freedom of the artist over freedom of the player” -ajattelutavaksi, kun lähtee viemään asiaa tarpeeksi pitkälle. Kysymyksiin ”onko tämä oikea tapa ajatella pelin tekemistä?” tai ”missä vaiheessa raja tulee vastaan?” ei löydy yhtä selkeää vastausta, koska erilaiset pelit rakennetaan erilaisten artististen näkemyksien pohjalta.

Kaikki tämä vaatisi siis animaatiotyylin tarkempaa tutkimista ja päätöksien tekemistä etenkin kameraan liittyvistä pelisuunnittelun puolista. Niin kuin sanoin, se olisi tarpeeksi kokonaan uuteen opinnäytetyöhön. Sama juttu kasvojen rigin kanssa, niin kuin mainitsin design-luvussa.

Testianimaatiota tehdessä paljastui, että olin tehnyt rigini olkapään niin, ettei se toimi Autodesk Mayan ulkopuolella. Ilmeisesti fbx tiedostomuoto ei osaa tallen-

taa eri rotation ordereita (järjestys, jossa x, y ja z akselien pyöriminen lasketaan) luille ja siksi hahmon olkapää menee rikki exportatessa, joten en tähän opinnäytetyöhön saanut valmista koko hahmon animaatiota pelimoottoriin asti. Pienellä rigin korjauksella sekin olisi mahdollista.

## 3.6 Renderöinti

### 3.6.1 Cel-shading ja automaattinen lineart

Ensimmäinen yritykseni Unreal Enginen piirretyn näköisen renderöinnin aikaansaamiseksi oli helpoin ja nopein tapa tehdä mistä tahansa objektista cel-varjostettu. Siihen käytettiin Unrealin post process -materiaalia.

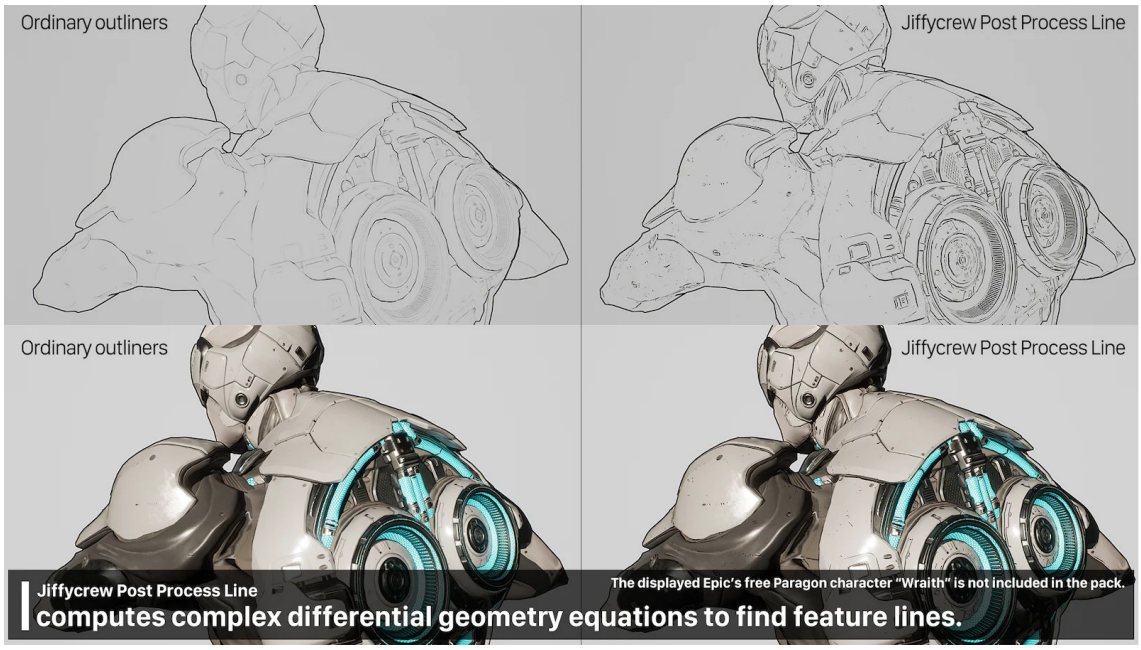
Post process eli jälkikäsittely tapahtuu, kun pelimoottori on jo renderöinyt kuvat kohtauksesta, mutta tekee vielä jotain muokkauksia niihin. Automaattiset ääriviivat tehdään käyttämällä depth pass nimistä renderöityä kuvaa, jossa valkoinen ja musta määrittävät kuinka kaukana kyseinen pikseli on kamerasta.

Kuva 49. Depth pass. Craig. (2018)

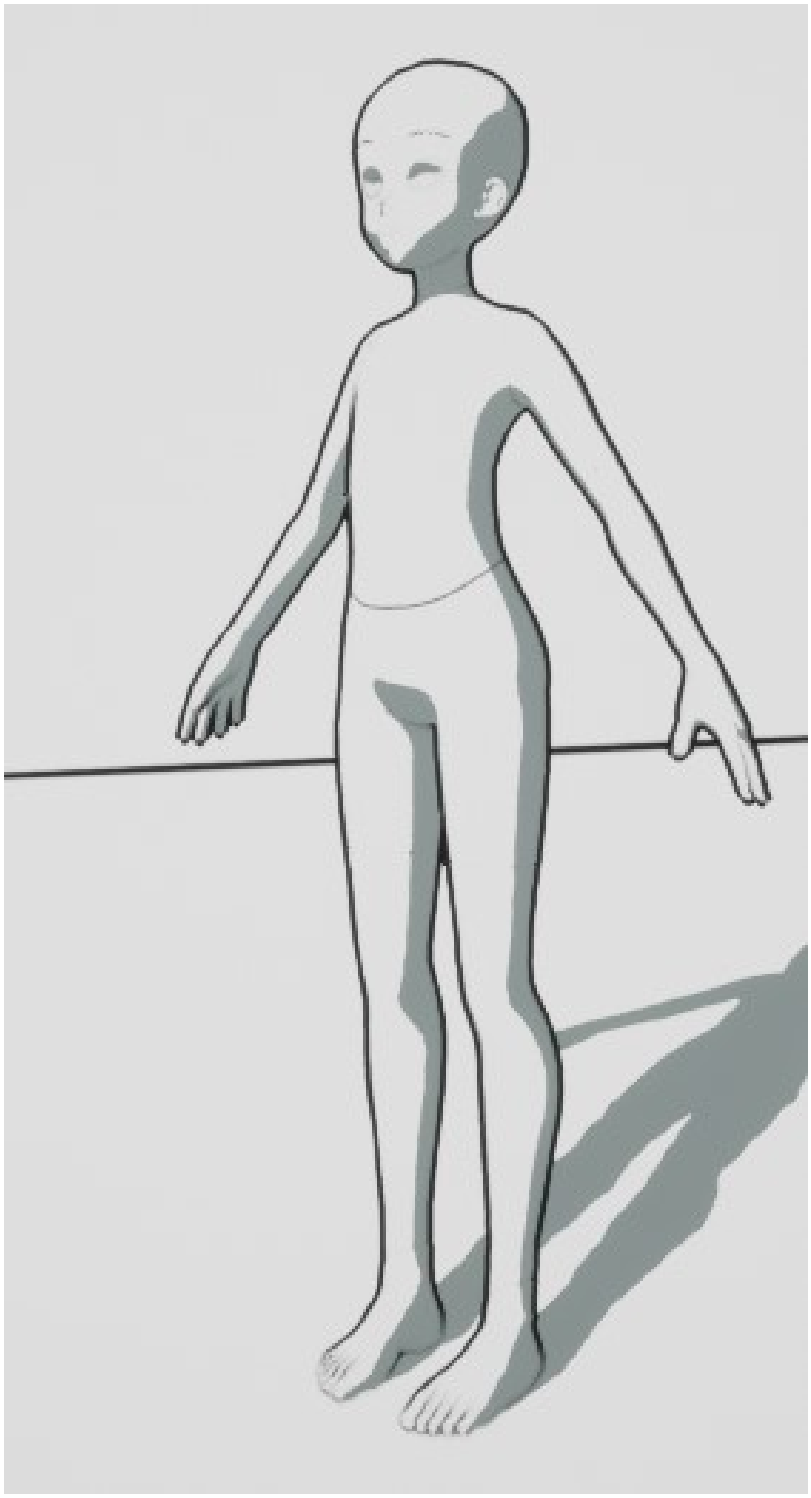
Ikkunoiden ulkopuolella on valkoista, koska se on kohtauksen kaukaisin paikka. Mitä lähempänä asiat ovat sen tummempina ne näkyvät depth pass kuvassa.

Jälkikäsittelyssä voidaan tehdä tälle syvyyskuvalle samankaltainen thresholdin rajaus kuin normaaleja käsittelevässä luvussa varjoille, näin saadaan rajat piirrettyä jokaiseen kohtaan, jossa on thresholdin mukainen ero tummemman ja vaaleamman alueen välillä.

Tämä tapa on yksinkertainen, kun sen yksinkertaisesti tekee, mutta kunnon koodaamisella siitä saa hyvinkin monimutkaisen. Jiffycrew Post Process Line on Unreal Enginen Marketplace kauppasivulla myynnissä oleva jälkikäsittely materiaali, jonka hinnaksi on laitettu 92 euroa (Jiffycrew 2019). Päättelin tästä asiaa tutkiessani, että tähän koodaamiseen lähteminen olisi minulle liikaa, joten en tutkinut tätä ääriviivojen piirtämistapaa sen enempää. Päätin myöhemmin siirtyä käyttämään mallintamista käsittelevässä luvussa mainittua inversed hull -tekniikkaa.



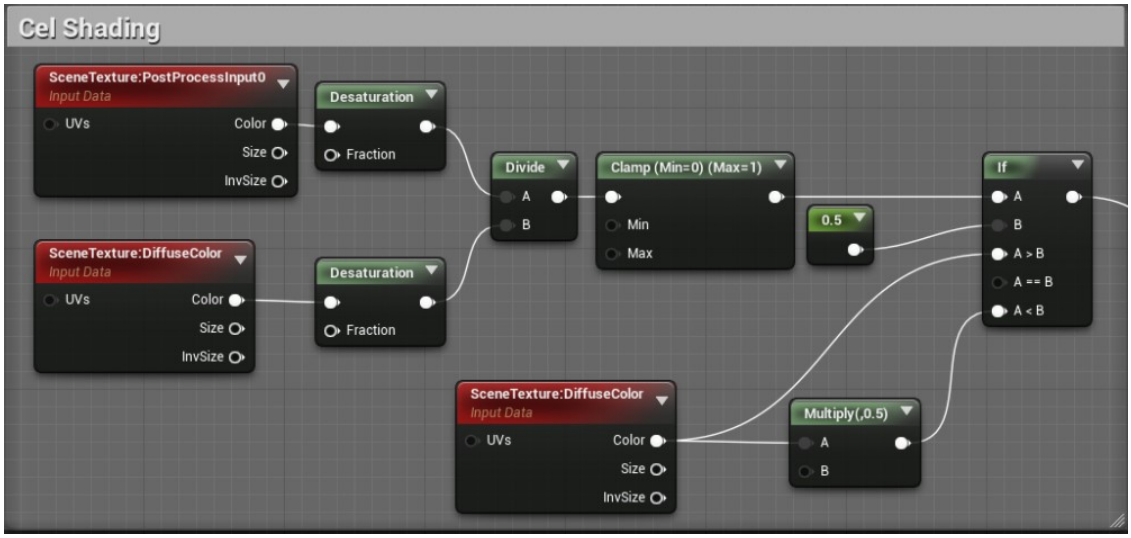

Kuva 50. Jiffycrew Post Process Line (2019) vertailu normaalin ääriviivojen piirtämisen kanssa.



Kuva 51. Ensimmäinen cel-shading hahmotesti.

Pääsin ensimmäisellä hahmotestilläni tähän asti. Jälkikäsittelyssä piirretyt ääriviivat näyttivät muuten hyviltä, mutta niiden paksuus määrittyi pikselipaksuuden mukaan, eli etäisyys ei vaikuttanut niihin. Kaukana ääriviivat peittivät lähes koko

hahmon. Niiden paksuutta ei myöskään saanut säädettyä erilaiseksi eri kohdille hahmoa.

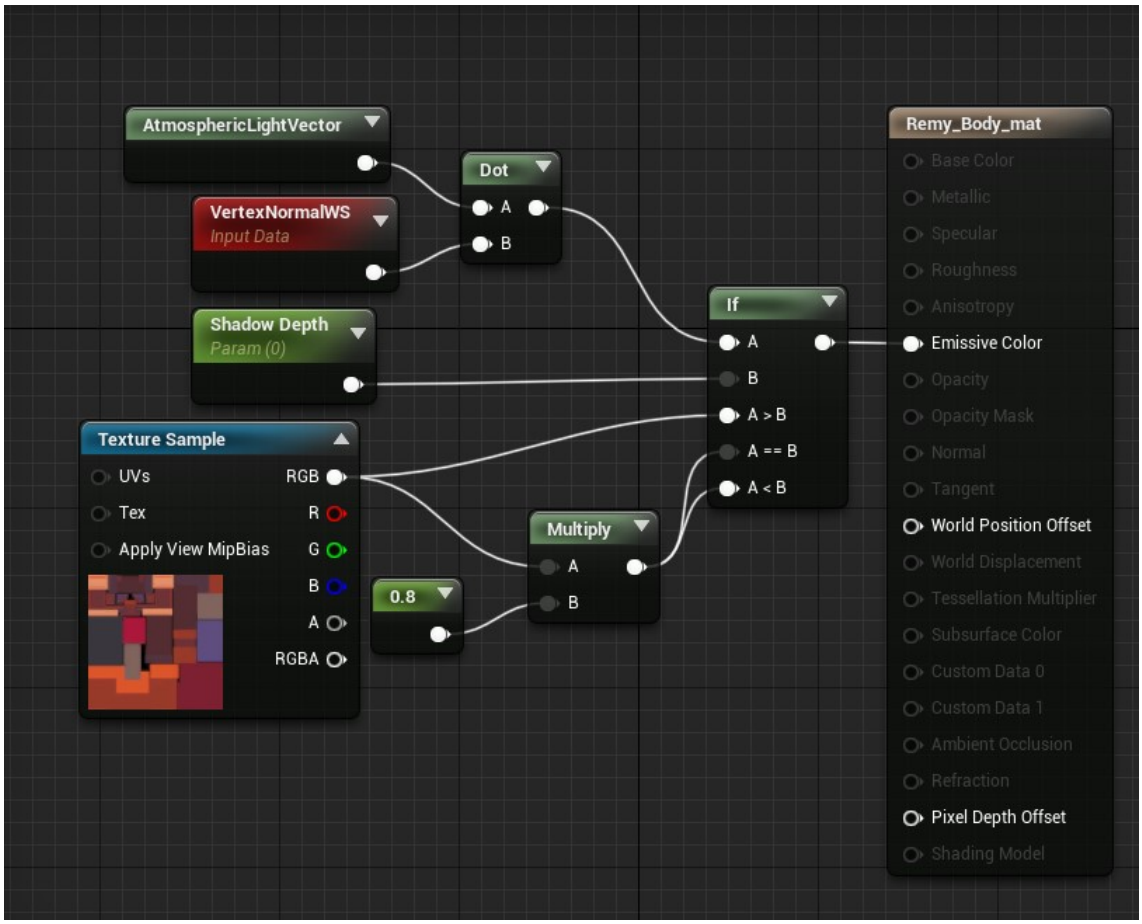



Kuva 52. Cel-shading Unreal Enginen post process -materiaalilla.

Myös cel-shading tapahtui ensimmäisellä yrityksellä jälkikäsittelyssä. Tässä oli se etu, että ei ollut väliä mikä materiaali objekteissa oli, sillä tämä noodisysteemi otti renderöidystä kuvasta (PostProcessInput0) pois värit (DiffuseColor) jättäen vain varjot mustavalkoisena kuvana, joka jaettiin thresholdin arvolla (0.5) valkoiseen ja mustaan, jotka väritettiin uudestaan alkuperäisellä värillä tai tummennetulla varjovärillä.

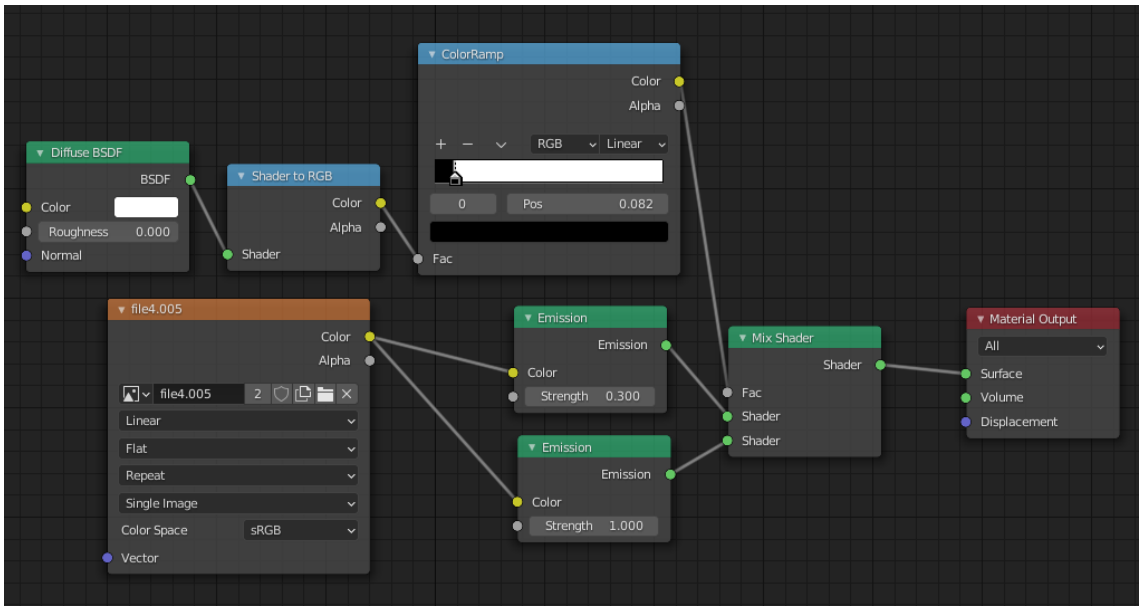
Ohjelmistoissa on aina se mukava asia, että joskus niitä päivitetään uusien ominaisuuksien sijaan ottamalla vanhoja pois tai muuttamalla niitä niin, etteivät ne enää toimi kuin ennen. Tämä cel-shading metodi ei toimi enää, koska Unreal Engine muutti tapaansa laskea DiffuseColorin renderöinti. Kuvista tulee nyt liian tummia, eivätkä varjotkaan toimi.

Onneksi ensimmäisenä löytynyt tapa tehdä asioita ei yleensä ole se paras. Löysin paremman tavan tehdä cel-shading jo objektien materiaaleissa ennen jälkikäsittelyä. Se on parempi tapa jo sen takia, että renderöintivaiheessa pystytään vielä vaikuttamaan objektien materiaaleihin toisinkuin jälkikäsittelyssä, jossa kaikki on jo renderöity. Tämä tapa on myös se mitä Guilty Gearissa käytettiin.




Kuva 53. Cel-shading Unreal Enginen normaalilla materiaalilla.

Kuvan 53 Unreal Engine -materiaalissa AtmosphericLightVector on ympäristön valon vektori, VertexNormalWS mallin pinnan vektori ja Shadow Depth on thresholdin arvo. Materiaali toimii siis juuri niin kuin normaalien muokkausta käsittelevässä luvussa nähdyt Guilty Gear -materiaalitkin. Arvon 0.8 tilalle voisi Multiply noodiin laittaa Tint tekstuurin, ja Shadow Depth -arvon tilalle voisi liittää vertex color -arvot hahmon mallista. Näin Arc System Works teki. Tämä materiaali on ”Unlit”, joka tarkoittaa, ettei se vastaanota valaistusta normaalisti, eikä siinä siis olisi varjoja ilman valon vektorin laskemista mukaan materiaalissa.




Kuva 54. Cel-shading Blenderin materiaalilla.

Kuvassa 54 on suunnilleen sama materiaali Blenderissä. Eroja ovat Diffuse shader noodi, joka laskee pehmeän valaistuksen objektille, josta sitten otetaan valkoinen ja musta alue ColorRamp noodilla (tämä toimii thresholdina).

Kuva 55. Cel-shading renderöinnin lopputulos Blenderissä.

Kuva 56. Valmis hahmo renderöitynä Blenderissä (vasen) ja Unreal Enginessä (oikea).

Tulos on Blenderissä suunnilleen saman näköinen kuin Unreal Enginessä. Eri arvoiset varjon tummuudet ovat oikeastaan ainoa ero kuvassa 56, paitsi että Blenderissä (vasen) puuttuu seuraava ominaisuus, jonka tein Unrealissa.

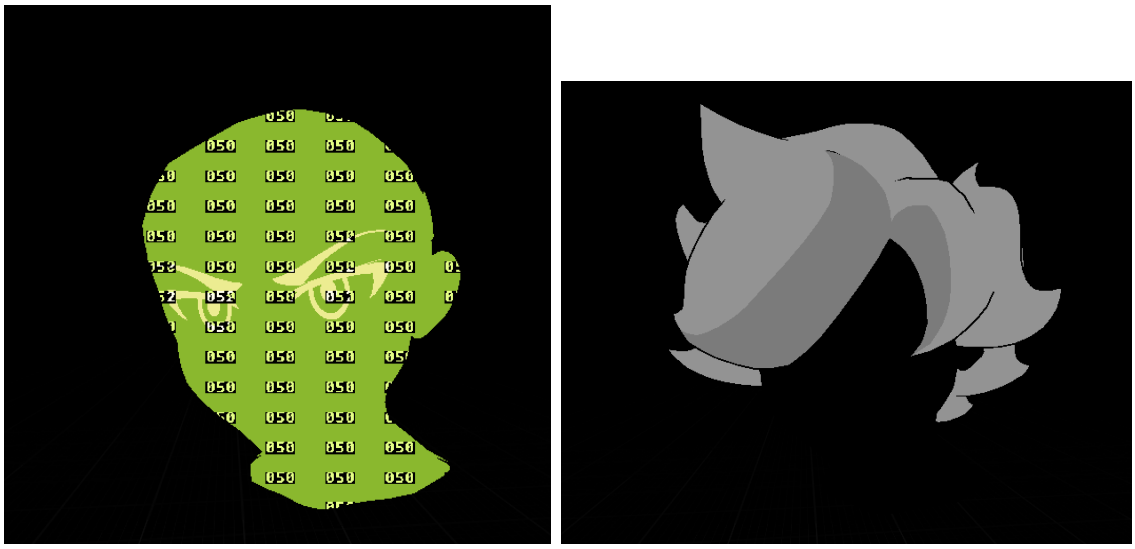
### 3.6.2 Läpinäkyvyysmaski hiuksiin

Halusin saada silmät renderöityä hiusten tietyn alueen päälle, tähän osoittautui parhaaksi tavaksi käyttää **Custom Depth Pass** ominaisuutta, jolla Unreal renderöi maskin objektille annetulla numeroarvolla. Tämä maski näkyy sellaisten objektien läpi, joissa ei ole tätä ominaisuutta päällä, mutta päällekkäin olevista custom depth -objekteista vain edessä oleva renderöidään maskiksi.

Kuva 57. Hahmo jaettuna kolmeen erikseen exportattavaan ryhmään.

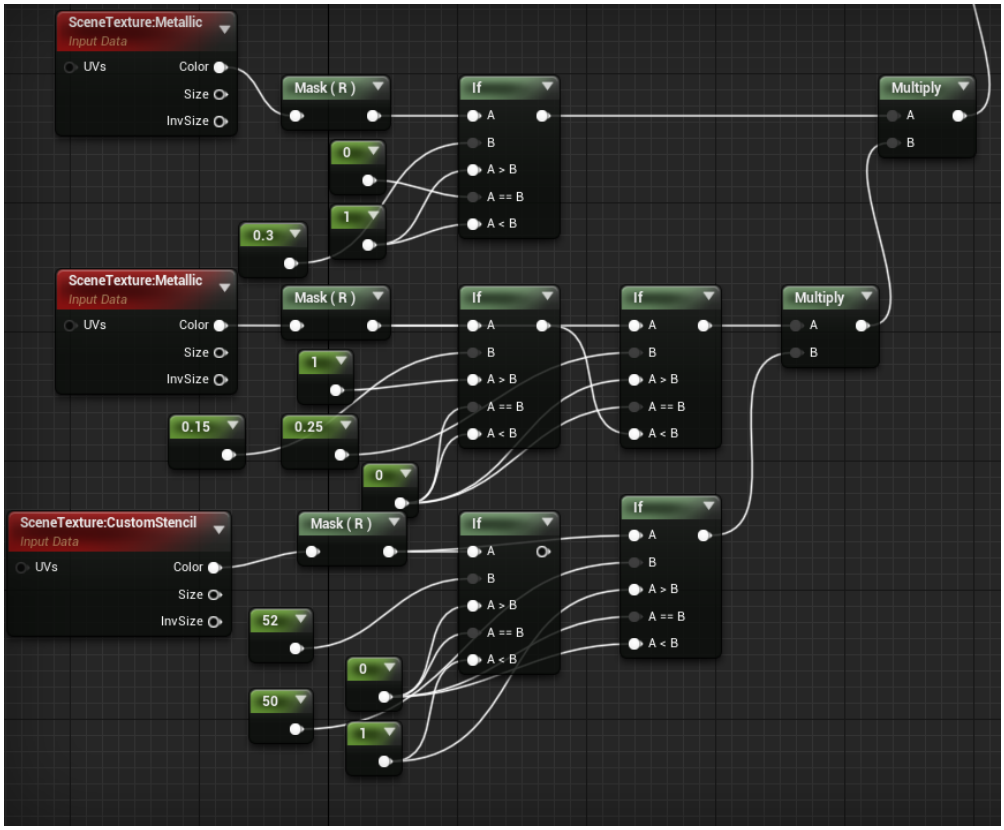
Koska custom depth -arvon voi antaa vain kokonaiselle hahmolle erillisen mallin tai materiaalin sijaan, oli hahmo jaettava kolmeen erilliseen osaan, jotka importattaisiin Unrealiin erikseen. Hahmo ilman silmiä tai hiuksia, johon tulisi custom depth -arvo, hiukset ja silmien iiris-pupilli-levyt, joista tulee näkyä läpi, ja lopuksi kulmakarvat, ripset ja silmien levyjen mallien sisään tätä tekniikkaa varten tekemäni silmämallit, jotka tein sen muotoiseksi kuin halusin hiuksissa näkyvien silmien olevan. Ne eivät näy normaalisti, mutta ne saavat kulmien ja

ripsien kanssa custom depth arvon näkyäkseen läpi hiuksista ja normaaleista silmistä. Alun perin tekniikkaan kuului myös, että hiusten alue, josta silmien piti näkyä läpi, oli maskattu metalness arvolla (joka renderöityy mustavalkoisena kuvana jälkikäsittelyä varten).



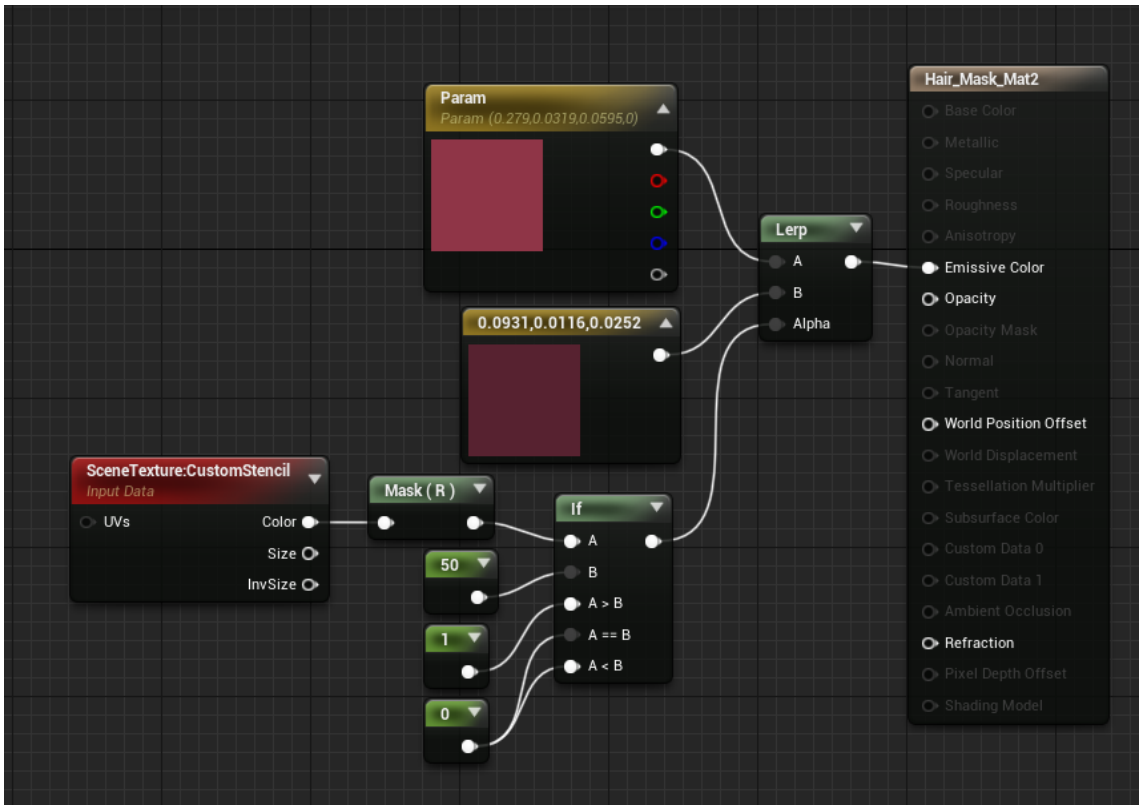
Kuva 58. Custom depth pass ja metalness pass.

Vasemmalla kuvassa 58 silmien custom depth maski arvolla 52 renderöityy pään maskin 50 päälle, koska se on fyysisesti pään edessä, pään takaa katsottuna silmien maski ei näy. Oikealla taas on mustavalkoinen metalness maski, tarkoitus on saada silmien maskin alue tumman harmaalle hiusten alueelle tummalla hiusten värillä.




Kuva 59. Ensin maskaamiseen käyttämäni jälkikäsittelymateriaali.

Koska hahmoni materiaalit käyttivät unlit-renderöintiä, ne hyväksyivät vain Emissive Color kanavan, eli niillä ei voinut renderöidä metalness maskia. Sain kuitenkin selville, että taas kerran pystyin tekemään saman asian paremmin jo ennen jälkikäsittelyä objektin materiaalissa, sillä asettamalla materiaalin Translucent renderöintiin se suostui käyttämään SceneTexture:CustomStencil noodia (custom depth pass, jota maskaukseen tarvittiin), jota normaalisti voi käyttää vain jälkikäsittelyssä.




Kuva 60. Hiusten läpinäkyvyysmateriaali.

Koska kuvan 60 maski meni kiinni suoraan etuhiusten vaalean alueen omaan materiaaliin (muilla hiusten alueilla oli eri materiaalit ilman tätä läpinäkyvyyttä), en tarvinnut enää metalness arvoa saadakseni silmät näkymään vain tällä alueella niin kuin jälkikäsittelyssä.

Kuva 61. Valmiin hahmon kasvot renderöitynä Unreal Enginessä.

Hiusten alueen maskaamisessa oli jälkikäsittelyä käyttäessä myös se ongelma, että antialiasing (puolipikselien, eli kovien, sahalaitaisten pikselireunojen pehmentäminen) ei toiminut sen cel-shading tyylin kanssa tärisemättä, eli olisin joutunut käyttämään joko pikselöitynyttä tai tärisevää hahmoa. Tämä liittyi järjestykseen, jossa jälkikäsittelyn renderöinti ja antialiasing tapahtuivat.

# 4 Yhteenveto

Projekti kokonaisuudessaan osoittautui vielä suuremmaksi urakaksi kuin odotin. Se paljasti, että tästä aiheesta, 2D- ja 3D-animaation tekniikoiden yhdistelystä videopeleissä, riittää monelle tutkittavaksi alueita mallintamisen, rigaamisen, animaation ja renderöimisenkin alueilta, puhumattakaan kaikista pelisuunnitteluun liittyvistä 2D-tyylin käyttämiseen vaikuttavista seikoista. Jo kameran käytössä olisi tarpeeksi aihetta kokonaiseen opinnäytetyöhön. Pelien ja elokuvien tällä aihealueella johtavat studiot ovat edenneet viime vuosina paljon ja niiden esimerkeistä on hyvä ottaa mallia, mutta jokainen projekti, jokainen visio lopputuloksesta vaatii kuitenkin omat tekniikkansa. Tutkimustyö ei siis luultavasti koskaan lopu.

Tärkein opetus, jonka itse tästä tutkimuksesta sain, oli kuinka tärkeää kontrollin tietoisesti jakaminen eri asioissa eri määrin tietokoneen ja artistin, tai pelisuunnittelijan ja pelaajan välillä on. Mitä tarkemmin artisti haluaa visionsa toteuttaa ja mitä paremmin aika ja raha siihen riittävät, sitä enemmän käsityötä lopputulos vaatii. Samoin mitä enemmän pelisuunnittelija haluaa vaikuttaa pelaajan kokemukseen, sitä enemmän pelaajalta tarvitsee ottaa kontrollia. Nämä kaksi mittaria on otettava huomioon pelejä tehdessä, sillä budjetti ja aika eivät ole loputtomia, eivätkä pelaajat halua vain istua katsomassa, kun peli pelaa itseään. Kaiken kontrollin ottaminen koneelta ja pelaajalta ei siis ole realistista, joten päätös tulee tehdä tietoisena siitä, mikä oman projektin kohdalla on tärkeintä, oli se sitten pelin tunnelma, pelaajan vapaus, ympäristön navigointi, uusi pelimekaniikka tai jonkin aivan uuden näköisen artistisen kokemuksen luominen.

Vastaani tulleet ongelmat näyttivät, että pelimoottoreita, tai ainakaan Unreal Engineä, ei ole suunniteltu tällaisen visuaalisen tyylin tuottamiseen, ja siksi niiden kanssa saa painia, kun yrittää tehdä tiettyä asiaa, joka omassa päässä tuntuu loogiselta ja helpolta, mutta jota ei ole kyseiseen ohjelmaan koodattu helpoksi tai edes mahdolliseksi. Ehkä tulevaisuudessa, kun työmetodit tällaisen visuaalisen ilmeen tuottamiseksi yleistyvät, joku tekee pelimoottorin juuri näitä metodeja varten.

Oman projektini prosessi oli monessa kohtaa hakuammuntaa, koska en ollut aikaisemmin asiaa tutkinut, enkä siksi tiennyt, mitä ongelmia eri tekniikoiden kanssa tulisi vastaan ja miten niihin tulisi valmistautua aikaisemmissa työvaiheissa. Eniten aikaa vei keksiä, miten tekisin valaistut ääriviivat ja silmien hiuksista läpinäkymisen. En näistä aiheista löytänyt esimerkkejä mistään pelaamistani peleistä, enkä osannut hakea Unreal Enginen ominaisuuksista työkaluja avukseni. Näihinkin ongelmiin löytyi ennen pitkää ratkaisut, jotka ovat toisella yrityksellä varmasti paljon helpompi toteuttaa, kun tietää mitä ongelmia odottaa. Jokaisen uuden tekniikan kanssa on varmasti sama hakeminen ja ideointi, riippumatta siitä, mitä pelimoottoria käyttää tai mitä tyyliä suunnittelee.

## Lähteet

1. Arc System Works 1998. Guilty Gear.
2. Arc System Works 2014. Guilty Gear Xrd.
3. Capcom 1998. Resident Evil 2.
4. CGSociety 2019. The Look of the Spider-Verse. https://cgsociety.org/news/article/4384/the-look-of-the-spider-verse (Luettu 01.05.2021)
5. Craig 2018. Using Image Planes and Depth. https://forums.autodesk.com/t5/maya-forum/using-image-planes-and-depth/td-p/8061307 (Luettu 05.05.2021)
6. IGN 2019. Promare – Official Movie Trailer. https://www.youtube.com/watch?v=UO91f-C_hlA
7. Jiffycrew 2019. Post Process Line. https://www.unrealengine.com/marketplace/en-US/product/jiffycrew-post-process-line
8. Motomura, Junya C 2015. Guilty Gear's Art Style : The X Factor Between 2D and 3D. https://youtu.be/yhGjCzxJV3E
9. Rooster Teeth 2020. RWBY.
10. Snyder, Chris 2019. How Oscar-Winning 'Spider-Man: Into the Spider-Verse' changed comic book movies forever. https://www.businessinsider.com/spiderman-spider-man-into-spider-verse-animated-frame-rate-marvel-stan-lee-2019-2?r=US&IR=T#:~:text=It%20took%20one%20week%20to,seconds%20they%20were%20happy%20with. (Luettu 01.05.2021)
11. Sony 2018. Hotel Transylvania 3: Summer Vacation.
12. Sony 2019. Spider-Man: Into the Spider-Verse.
13. Studio Trigger 2013. Kill la Kill.
14. Studio Trigger 2019. Promare.
15. Wit Studio 2019. Attack on Titan.

# Liitteet

## Hahmon kasvoanimaatiotesti

Animoimani testi kun olin tehnyt vasta hahmon pään.

https://www.youtube.com/watch?v=t-hprPww2yA

**Hahmon koko kehon animaatiotesti**

Animoimani testi projektin loppuvaiheilta.

https://www.youtube.com/watch?v=bQ-kYR-6bus