Daniel Palmi

Mobiilipeligrafiikan optimointi

Case: Trials Frontier

Metropolia Ammattikorkeakoulu

Medianomi (AMK)

Viestintä

Opinnäytetyö

Metropolia

Tekijä(t) Otsikko Sivumäärä Aika	Daniel Palmi Mobiilipeligrafiikan optimointi 36 sivua 8.9.2013
Tutkinto	Medianomi (AMK)
Koulutusohjelma	Viestintä
Suuntautumisvaihtoehto	3D-animointi ja -visualisointi
Ohjaaja(t)	Lehtori Kristian Simolin

Opinnäytetyön tavoitteena on tuoda esille merkittävimmät tekniikat ja seikat peligrafiikan optimointiin 3d-graafikon näkökulmasta. Oppimani tekniikat liittyvät tiiviisti Trials Frontier –mobiilipeliin, joka on kirjoitushetkellä kehitteillä Redlynx Ubisoft -studiossa. Olen projektissa 3d-graafikkona ja luonut peliin suurimman osan 3d-objekteista. Projektin parissa on työn kautta tullut opittua monia tekniikoita reaaliaikaisen grafiikan optimointiin ja esitän tässä opinnäytetyössä keskeisimmät monilla esimerkkitapauksilla.

Suurin osa optimoinnin esimerkeistä liittyy teksturointiin tai tekstuurikoordinaattien määrittelemiseen. Käytänkin esimerkeissäni paljon havainnollistavia kuvia malleista ja tekstuureista, joita olen luonut peliin projektin aikana. Reaaliaikaisen grafiikan optimointiin liittyy myös paljon teknisiä asioita pelimoottorin ja koodin puolelta. Näistä pyrin käymään läpi perusteet, joita graafikon on hyvä ymmärtää, vaikka rajauksen kannalta ne eivät graafikolle olisi tärkeimpiä tekijöitä.

Opinnäytetyön rajauksen kannalta työni keskittyy erityisesti mobiilipeleihin, enkä käsittele varsinaisesti konsolipelien reaaliaikaista grafiikkaa optimoinnin kannalta. Samat periaatteet pätevät kuitenkin myös konsolipuolella, ja sivuan työssäni myös tätä aluetta hieman.

Avainsanat	3d, peli, mobiili, reaaliaikainen, grafiikka, optimointi

Author(s) Title Number of Pages Date	Daniel Palmi Mobile game graphics optimization 36 pages 8 September 2013
Degree	Bachelor of Media
Degree Programme	Media Communication
Specialisation option	3D Animation and Visualization
Instructor(s)	Kristian Simolin, Senior Lecturer

The aim of this study is to highlight the most important aspects and techniques of game graphics optimizations from the 3D-graphic artist's point of view. The techniques I learned relate closely to the Trials Frontier - mobile game, which is under development at Redlynx Ubisoft -studios at the time this thesis is being written. I work as a 3d graphic artist in this project and I have created most of the 3d-objects and textures for this game. During this project I have come across many techniques for optimizing real-time graphics under the development of this game, and my target in this thesis is to present the most important ones with several examples.

Most of these examples associate with defining good textures and effective unwrapping. I use a lot of images to illustrate these models and textures that I have created for the game. Real-time graphics optimization also involves a lot of technical things in the game engine and the code side. Of these, I will try to go through the fundamentals, even though from the artist's point of view they would not be the most important factors.

This thesis focuses mostly on mobile games, and it does not include the side of console-games and technology related to them. The same principles do also apply to the console side, and for this reason I also examine this area a little bit.

Keywords	3d, game, mobile, real-time, graphics, optimization

Sisällys

1 Johdanto

Tämän opinnäytetyön tavoitteena on esitellä lukijalle keskeisimmät tekniikat ja keinot mobiiligrafiikan optimointiin. Kaikki esittelemäni esimerkit liittyvät kirjoitushetkellä kehitteillä olevaan Redlynxin Trials Frontiers –mobiilipeliin. Trials Frontier on juurilleen uskollinen ja Redlynxille ominainen Trials-peli, joka julkaistaan vuoden 2014 alussa. Trials-pelit ovat Redlynxin tavaramerkkejä ja tunnettuja koukuttavasta pelimekaniikasta ja tarkasta kontrollista. Trials-pelit ovat moottoripyöräpelejä, joissa pelaajan tulee selvittää esterata alusta loppuun. Sarjalle ominaista on myös tarkka fysiikka, johon pelaaja saa helposti otteen ja pystyy kontrolloimaan pyöräänsä haluamallaan tavalla. Trials-pelit ovat haastavia, mutta eivät epäreiluja. Tämä onkin sarjan koukuttavin tekijä, mikä saa pelaajan yrittämään aina uudelleen ja uudelleen.

Tulen käymään opinnäytetyössä esittelemäni tekniikat läpi lukuisilla esimerkeillä ja kuvilla, joilla pyrin havainnollistamaan käytännön esimerkit ohjelmistosta riippumatta. Rajaukseen liittyy olennaisesti Deferred Rendering –renderöintitekniikka, joka asettaa tiettyjä haasteita mobiiligrafiikan tuottamiseen. Keskityn tässä työssä optimointiin nimenomaan 3d-graafikon näkökulmasta enkä tekniseltä kannalta pelimoottoriin tai koodiin liittyen. Oletuksena opinnäytetyössä on lukijan ymmärrys mallintamisen ja tekstuuroinnin perusteista. Käytän esimerkeissäni Blender -ohjelmistoa, mutta tekniikat mallintamisen tai tekstuurikoordinaattien määrittelemiseen eivät ole sidonnaisia ohjelmistoihin.

Käyn luvuissa 2-3 läpi tarkemmin keskeisimmät käsitteet ja tekniikat, jotka ovat lähtökohtana reaaliaikaisen peligrafiikan tuottamisessa ja optimoinnissa. Avaan tässä osiossa hieman myös käsiteltävän renderöinti-tekniikan perusteita ja pelimoottorien toimivuutta yleisesti. Tämä osio toimii pohjustuksena käytännön esimerkeille, joita käsittelen seuraavassa luvussa. Reaaliaikaisen ja peligrafiikan aiheet käsittävät laajoja asioita, joista pyrin avaamaan vain olennaisimmat kontekstin ymmärtämiseksi. En siis pyri syventymään mihinkään näistä aiheista erityisen tarkasti, vaan tarkoituksena on avata nämä asiat ymmärrettäviksi aiherajauksen puitteissa. Käytän tässä osiossa havainnollistavia kuvia aiheiden ymmärtämiseksi.

Luku 4 sisältää kaikki käytännön esimerkit ja sovellutukset. Käytän esimerkeissäni apuna itse luomiani tekstuureita ja mallinnuksia, jotka perustuvat valmiisiin pelin konsepteihin. Esimerkkimallit ja tekstuurit liittyvät kaikki pitkälti Trials Frontier-peliin ja sen maailmaan. Osa näistä malleista on varhaisia versioita, mutta suurin osa on lopullista tuotantoa ja tulevat myös peliin sellaisina kuin tässä opinnäytetyössä esiintyvät. Vertailukohtana näille esimerkeille käytän kuitenkin ajoittain muita malleja ja tekstuureita muista lähteistä. Neljäs luku tulee myös sisältämään lukuisia kuvia asioiden havainnollistamisessa.

2 Käsitteiden määrittely

2.1 Kohderyhmänä 3d-graafikot

Työni on suunnattu erityisesti pelialan 3d-graafikoille, jotka työskentelevät paljon tekstuurien ja uv-koordinaattien kanssa mobiili- ja lowpolygrafiikan tuotannossa. Tässä opinnäytetyössä oletetaankin lukijalla olevan tietyn tason ymmärrys 3d-grafiikasta ja sen tuottamisesta erityisesti peligrafiikan puolella. Työssäni pyrin antamaan hyödyllisiä vinkkejä optimaalisemman grafiikan tuottamiseen niille, joille aiheet ovat jo tuttuja. Esimerkit painottuvat suurelta osin teksturointiin ja uv-koordinaattien määrittelemiseen. Oletan myös lukijan ymmärtävän useimmat työssä esiin tulevat käsitteet, vaikka keskeisimmät näistä pyrin avaamaan auki seuraavassa osiossa. Luvussa kolme käyn läpi kokonaisuuksina suurempia aihealueita, jotka eivät ole listattuna keskeisimmissä käsitteissä. Erityisesti pelialasta kiinnostuneet 3d-grafiikan opiskelijat voivat hyötyä tästä opinnäytetyöstä ja esittämistäni optimoinnin tekniikoista.

2.2 Keskeisimmät käsitteet

Polygoni – Monikulmio, joka muodostuu kolmesta tai neljästä verteksistä, eli pisteestä 3d-koordinaatistossa. Polygonia, joka muodostuu useammasta kuin neljästä verteksistä kutsutaan **N-goniksi**.

Low-Poly – Termi, joka viittaa 3d-objektiin, joka koostuu suhteellisen pienestä määrästä polygoneja.

UV-koordinaatit / Unwrap – Viittaan näillä termeillä ajoittain opinnäytetyössäni tekstuurikoordinaattien määrittelemiseen. Tekstuurikoordinaatit määrittelevät kuvatekstuurin polygonin pinnalle.

Polygonimäärä – (polycount) monikulmioiden määrä 3d-objektissa

Alphakanava – kuvatiedoston sisältämä kanava, jolla kontrolloidaan läpinäkyvyyttä kuvassa.

FPS – (Frame per second) Näytölle piirrettävien kuvien määrä sekunnissa.

3 Peligrafiikan optimoinnin lähtökohdat ja tekniikat

3.1 Työkalut

Kaikissa esimerkeissäni olen käyttänyt 3d-työkaluna Blender-ohjelmistoa ja tekstuurien maalaamisessa Photoshop-kuvankäsittelyohjelmistoa. Käsittelemäni tekniikat eivät ole sidonnaisia näihin ohjelmistoihin, enkä käsittele tarkkoja toimintoja ohjelmistojen sisällä erikseen. Kaikki käyttämäni tekniikat ovat siis riippumattomia käyttämistäni ohjelmistoista. Toisin sanoen, vaikka Blender ei olisi lukijalle tuttu, voi tekniikoita soveltaa samalla tavalla vaikka 3dsMax- tai Maya-hjelmistoissa.

3.2 Lowpoly- ja mobiiligrafiikka

Lowpoly- ja mobiiligrafiikka mielletään perinteisesti vähäisiksi polygonimääriksi ja pieniksi tekstuuriko'oiksi. Mobiililaitteiden nopea kehittyminen kuitenkin mahdollistaa jo nykypäivänä suhteellisen laadukkaan grafiikan toistamista kannettavilla alustoilla verrattuna PC- tai konsolialustoihin. Tekniikan ja laitteiston tasolla nämä poikkeavat toisistaan kuitenkin merkittävästi ja mobiilipuolella rajoitteet ovat vielä melko suuret korkeatasoisen grafiikan piirtämisessä. Mobiiligrafiikan tuottamisessa on edelleen pidettävä mielessä polygonimäärät, tekstuurikoot ja drawcall –vaiheet. Erityisesti Redlynxillä käyttämässämme Genetek-pelimoottorissa Trials Frontierin kanssa polygonimäärät olivat pienin murheemme suorituskyvyn kannalta. Todellisuudessa enemmän päänvaivaa aiheuttivat vähäinen muistin määrä, drawcall-vaiheet jokaisen kuvaruudun aikana, sekä läpinkyvyyden renderöinti, eli alpha-kanavan käyttäminen tekstuureissa. Graafi-

kon näkökulmasta on siis otettava huomioon kaikki nämä seikat 3d-peligrafiikan tuotannossa, mutta erityisesti projektissa korostui optimaalisten tekstuurien luominen. Graafikolle tämä siis tarkoittaa sitä, että tekstuurien tulisi olla mahdollisimman pienikokoisia tinkimättä liikaa kuitenkaan kuvanlaadusta. Parhaiten näihin tuloksiin pääsee tehokkaalla tekstuurikoordinaattien määrittämisellä ja toistuvien osuuksien käyttämisellä tekstuureissa. Luvussa 4 palaan näihin tekniikoihin tarkemmin havainnollistavien esimerkkien kanssa. (Eat 3D 2013.)

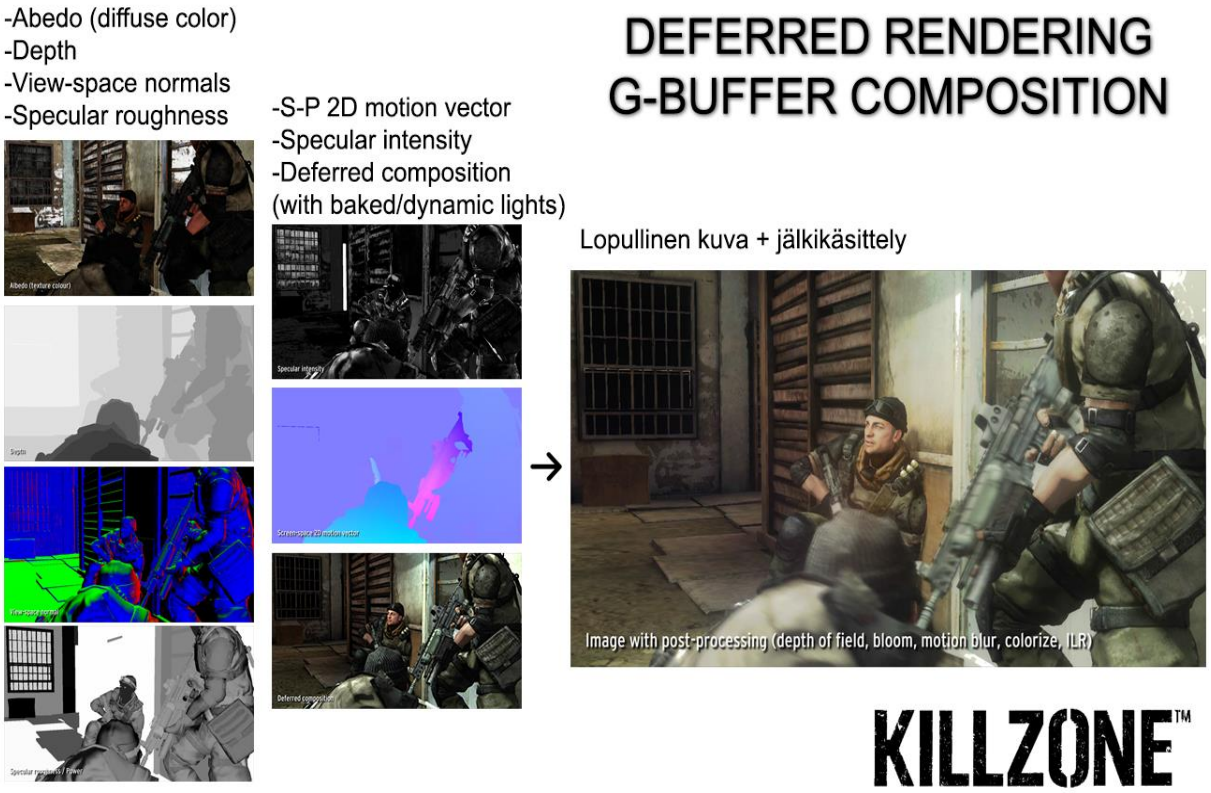
3.3 Drawcall

Drawcall tarkoitta vaihetta pelimoottorissa, jossa se kutsuu objektin ja sen tekstuurit piirrettäväksi jokaisen yksittäisen kuvaruudun aikana. Jokainen objekti on siis itsessään jo yksi drawcall. Näihin lasketaan myös mukaan jokainen valonlähde, materiaalit ja tekstuurit. Toisin sanoen suuret drawcall -määrät ovat raskaita ja näitä vähentämällä voidaan optimoida suorituskykyä. Drawcall –määrien vähentämiseen on monia keinoja, joista yksi on objektien ja tekstuurien yhdisteleminen niin, että moni objekti käyttää samaa tekstuuria. Jos useat objektit käyttävät yhtä tekstuuria, säästetään siis suoraan drawcall-määrissä. Käsittelemäni optimoinnin keinot opinnäytetyössä liittyvätkin pitkälti myös drawcall-määrien vähentämiseen muistin säätämisen ohella. (Eat 3D 2013.)

3.4 Deferred Rendering –tekniikka lyhyesti

Osa optimoinnin tekniikoista liittyy myös keskeisesti käytettäviin renderöintitekniikoihin, joita pelimoottorit käyttävät. Useissa mobiilipuolen pelimoottoreissa käytetään Deferred Rendering -tekniikkaa sen paremman suorituskyvyn vuoksi. Kaikki käymäni esimerkit on tuotettu juuri tätä silmällä pitäen ja huomioitu tuotannossa. Trials Frontierin Genetek-engine käyttää ainostaan tätä renderöintitekniikkaa. Opinnäytetyön kannalta syventyminen kyseisiin renderöintitekniikoihin ei ole olennaista, mutta silti hyödyllistä ymmärtää edes pintatasolla. Pelimoottori, jota varten olen grafiikan tuottanut käyttää siis vain ja ainoastaan Deferred Rendering –tekniikkaa, vaikka muilla alustoilla olisi mahdollista käyttää myös Forward Rendering –tekniikkaa tämän rinnalla. Tämä aiheuttaa haasteita tekstuurien ja mallien luomisessa, koska tekniikka rajoittaa läpinäkyvyyden käyttämistä huomattavasti. Esimerkiksi kaikenlaiset kasvit ja puiden lehdet ilman alpha-kanavaa tuovat haasteita graafikolle mallinnuksessa ja tekstuurien maalaamisessa. Läpinäkyvyyden renderöinti on mahdollista Deferred Rendering –tekniikan kanssa, mutta äärimäisen raskasta suorituskyvyn kannalta. Miksi siis käyttää Deferred Rendering –

tekniikkaa Forward Rendering –tekniikan sijaan? Forward Renfering –tekniikka piirtää kuvan klassisin menetelmin, renderöiden jokaisen objektin ja valon erikseen. Jos renderöitävässä tilassa on siis käytössä esimerkiksi seitsemän valoa, renderöidään jokainen objekti ja valo seitsemän kertaa. Tämä on suorituskyvyn kannalta raskasta, kun mukaan lasketaan myös objektien tekstuurit ja materiaalit. Deferred Rendering -teknikka taas erottaa valon eri informaatiot omiksi tekstuureiksi, jotka tallentuvat G-buffer -kanavaan. Tämän jälkeen tekstuureista tehdään kompositio, ja näin syntyy lopullinen kuva. Deferred Rendering -tekniikan etuna on siis suorituskyvyn säästäminen yhdistämällä G-bufferin informaatio lopulliseksi kuvaksi ilman, että jokainen valo ja objekti pitäisi renderöidä erikseen. Kuvassa 1 esimerkki kyseisestä prosessista Killzone 2-pelissä. (Valient, 2007, Gat. 2010, Hargreaves, 2013)



Kuva 1. Deferred rendering g-buffer kompositiossa. (Valient, 2007)

3.5 Alpha-kanavan ongelmallisuus

Tapa, jolla kuva tuotetaan Deferred Rendering –tekniikalla, on ongelmallinen läpinäkyvyyden ja läpikuultavuuden suhteen. Prosessin aikana eri valoinformaatioiden tekstuurit objekteista voidaan käytännössä asettaa päällekkäin missä järjestyksessä vain, sillä objektien järjestys ja syvyys ei ole olennaista. Läpikuultavien objektien renderöinti vaatii kuitenkin kummankin näistä informaatioista. Käytännössä läpikuultavat objektit tulee renderöidä useaan kertaan, kuorien jokaisella kerralla yhden kerroksen objektin syvyydestä kunnes lopulliset läpikuultavat pikselit pystytään määrittelemään. Tätä kutsutaan ”depth peel” –tekniikaksi. Jos läpikuultavia objekteja on useampia peräkkäin, prosessi toistuu monikertaisesti. Tämä on kuitenkin erittäin raskasta ja vaatii suuret määrät muistia. Vaikka läpikuultavuus on siis tällä tekniikalla täysin mahdollista, se voi romuttaa suorituskyvyn täysin. Tämä korostuu tietysti mobiililaitteilla, joilla muistin määrä on entistä pienempi. Usein tämän vuoksi monet konsoli- ja pc-pelimoottorit käyttävätkin Forward Rendering –tekniikkaa pitkälti vain läpikuultavuuden renderöintiin, sillä sen kanssa se on kevyempää. Jos käytössä on kuitenkin vain Deferred Rendering -tekniikka, on alpha-kanavan käyttäminen hyvin harvinaista tai usein minimaalista. (Null_ptr 2009, GameDev.net 2013, Everitt 2013, Hargreaves 2013)

Pinnan alla nämä tekniikat ovat tietysti huomattavasti monimutkaisempia ja tämä on vain esitys pähkinänkuoressa. Lopulta tähän tekniikkaan liittyen tärkeää graafikon näkökulmasta on vain alphakanavan ja läpinäkyvyyden käyttäminen, tai sen käyttämättä jättäminen. Käytännössä en olekaan käyttänyt alphakanavaa laisinkaan malleissani ja tekstuureissani, vaan olen pyrkinyt ratkaisemaan graafiset ongelmat toisilla keinoilla, usein lisäämällä reilusti enemmän geometriaa. Polygonimäärien lisääminen kohtuuden rajoissa voi silti olla usein suorituskyvyn kannalta parempi vaihtoehto kuin läpinäkyvyyden renderöinti. (Null_ptr 2009, GameDev.net 2013, Everitt 2013, Hargreaves 2013)

3.6 Tekstuurikoot ja muisti

Tekstuurien kanssa on usein haasteena tasapainoilla tekstuurin koon ja sen kuvanlaadun välillä. Pienemmät tekstuurit vievät tietysti vähemmän muistia, mutta voivat verottaa kuvanlaatua ratkaisevasti. On myös muistettava, että siirtyminen esimerkiksi tekstuurikoosta 128*128 kokoon 256*256 tarkoittaa nelinkertaista pikselimäärää ja pinta-

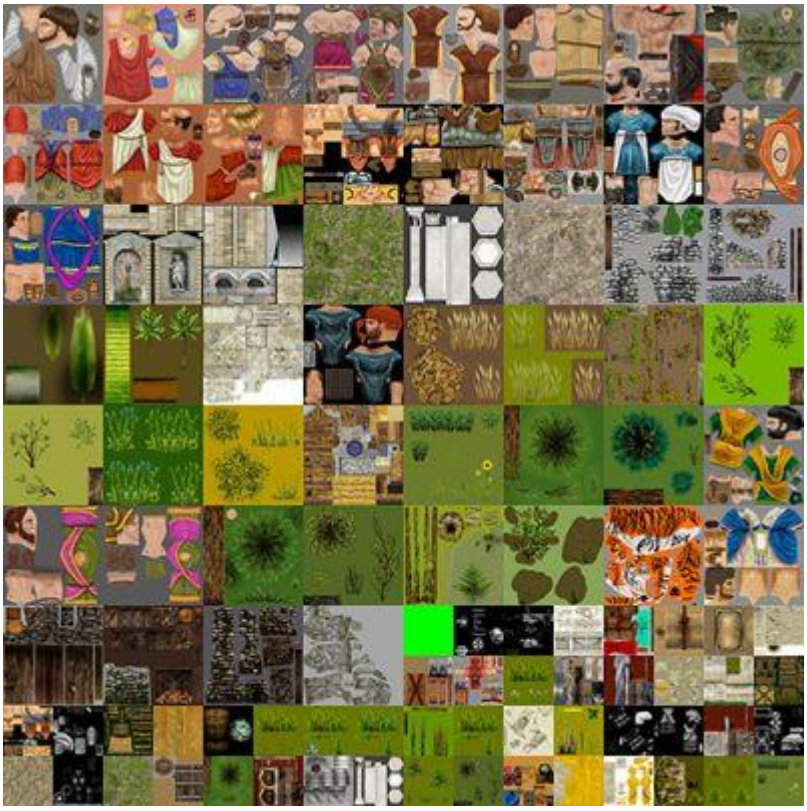
alaa tekstuurikoordinaateille. Jokainen uniikki tekstuuri lisää myös yhden drawcall-määrän lisää, mikäli se ei ole atlasoitavissa. Mitä useampi objekti pystyy siis käyttämään samaa tekstuuria, sitä vähemmän drawcall –vaiheita ja muistia se tarvitsee. Kun mobiilipeleissä muisti on usein ensimmäinen rajoittava tekijä suorituskyvyn kannalta, on graafikolle tärkeintä optimoida juurit tekstuurit ja UV-koordinaatit. Optimointi 3d-graafikon kohdalla kulminoituukin tässä opinnäytetyössä juuri tähän problematiikkaan. Kuinka luoda mahdollisimman vähän muistia vievät tekstuurit uhraamatta liikaa laadussa?

Kuva 2. Ero kuvanlaadussa, kun siirrytään resoluutiosta 256x256 resoluutioon 128x128

3.7 Atlasointi ja saumattomat tekstuurit

Atlasointi tekstuureissa tarkoittaa käytännössä useiden tekstuureiden yhdistämistä yhdeksi suureksi tekstuuriksi. Etu tässä on se, että näin tämä yksi tekstuuri tarvitsee vain yhden drawcall-määrän niissä objekteissa, jotka tätä tekstuuria käyttävät. Usein tämä on täysin automaattinen prosessi pelimoottorin sisällä, mutta on myös tilanteita, jolloin manuaalista atlasointia eli tekstuurien yhdistelemistä kuvankäsittelyohjelmassa on hyvä harkita. Esimerkiksi tehokasta olisi luoda omat tekstuurit eri materiaalityypeille, kuten puulle ja metallille. Jos pelissä on esimerkiksi paljon puisia samankaltaisia objekteja, on optimaalista luoda yksi puutekstuuri näille kai kille objekteille. Palaan näihin vielä esimerkkien kanssa neljännessä luvussa. Saumattomat tekstuurit ovat tietysti itsessään jo optimaalisia, kun samaa tekstuuria voidaan toistaa peräkkäin loputtomasti.

Saumattomat tekstuurit eivät kuitenkaan ole tietenkään atlasoitavissa. Saumattomia tekstuurin osia voi kuitenkin käyttää itse tekstuurin sisällä hyödyksi, mikäli geometria on luotu sitä varten. Eli voidaan silti luoda atlasoitava tekstuuri, jonka sisällä on saumattomasti toistuva osa, mikäli mallinnus on tehty tätä silmällä pitäen. Tämä vaatii kuitenkin tekstuurifilteröinnin vuoksi enemmän tarkkuutta tekstuuria maalatessa. Palaan tähänkin esimerkeillä luvussa 4. Kuvassa 3 esimerkki atlasoidusta tekstuurista.



Kuva 3. Esimerkki atlasoidusta tekstuurista. (Gamasutra 2013)

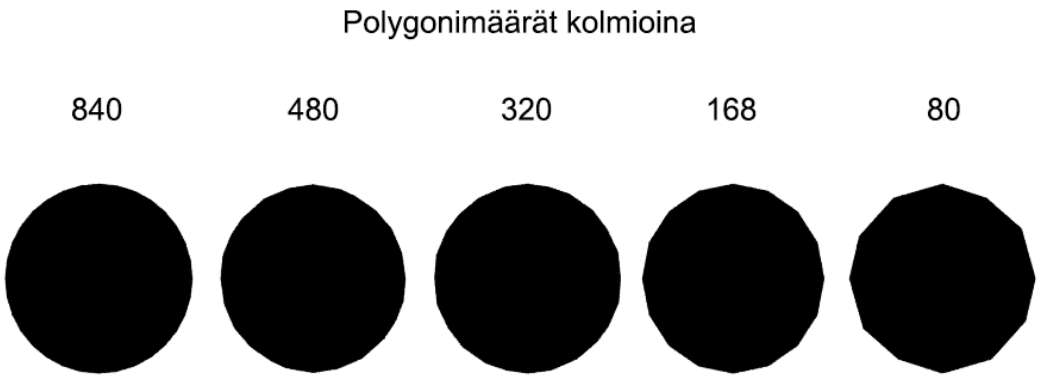
3.8 Vertexcolor

Vertexcolor on perinteinen tekniikka värjätä mallin osia syöttämällä väri-informaatio suoraan verteksiin. Tällöin väri-informaatio säilyy suoraan mallin jokaisessa verteksissä, eikä se välttämättä tarvitse tekstuuria. Verteksivärjäys on erittäin tehokasta suorituskyvyltään ja sen lisäksi voidaan käyttää myös normaaleita teksturointimenetelmiä. Verteksivärit voivatkin olla hyvä lisä jo teksturoidulle mallille, luoden siihen esimerkiksi varjostusta tai värivariaatioita, etenkin jos sama malli toistuu useaan kertaa ruudulla. Näistäkin näytän muutaman esimerkin luvussa 4. (Eat 3D 2013, Polycount wiki 2011, Polycount wiki 2013.)

4 Käytännön sovellutukset ja esimerkit

4.1 Polycount ja siluetti

Low-Poly-mallinnuksessa on muistettava vähäisten polygonimäärien kanssa mallin tunnistettava siluetti. Mallinnuksessa tulisikin siluetin kannalta pyrkiä helposti tunnistettavaan muotoon, vaikka malli olisi täysin teksturoimaton tai varjostamaton. Jos siluetti on tunnistettava, helpottaa se myös teksturointia. Samanlaiseen siluettiin voidaan myös päästä huomattavan vähäisillä polygonimäärillä. Esitän havainnollistavan esimerkin kuvassa 4 yksinkertaisella 3d-pallogeometrialla, jonka polygonimäärät vaihtelevat suuresta vähäiseen. Objekti käyttää täysin mustaa valaisematonta materiaalia. Kolme keskimmäistä palloa muistuttavat toisiaan siluetiltaan melko paljon, vaikka polygonimäärä olisi moninkertainen. Kauempaa katsottuna ero on vielä vaikeammin havaittavissa. Mallinnuksessa pitää toki pitää mielessä lopullinen mallin sijoitus ja sen fokus, eli tarkastellaanko mallia lopulta läheltä vai kaukaa. On siis tärkeää tasapainoilla polygonimäärän ja laadun kanssa sen mukaan, mihin rooliin malli tulee lopulta sijoittumaan pelimaailmassa.



Kuva 4. Eri polygonimäärät ja niiden luoma siluetti

Tämänhetkisissä mobiililaitteissa kuitenkin polygonimäärät eivät usein ole ensimmäinen vastaantuleva ongelma, kuten aikaisemmin jo mainitsin. Selkeästi enemmän vaivaa aiheuttavat liikaa muistia vievät tekstuurit. Monet pelimoottorit pystyvätkin pyörittämään huomattavasti enemmän polygoneja kuin voisi kuvitella, mikäli optimointi on kohdallaan tekstuureiden ja drawcall-määrien suhteen. Omassa työssäni Redlynxillä tulin useasti tilanteeseen, jossa polygonimäärien vähentäminen malleissa oli turhaa, ja

minut ohjattiin sen sijaan optimoimaan tekstuureita. Toki edelleen polygonimäärissä on pidettävä kohtuus, eikä niissä päästä konsolipelien tasolle kuitenkaan. Suuremmat polygonimäärät mahdollistavat myös tekstuurin entistä tehokkaampaa käyttämistä, sillä toistuville osille voidaan määrittää päällekkäiset tekstuurikoordinaatit.

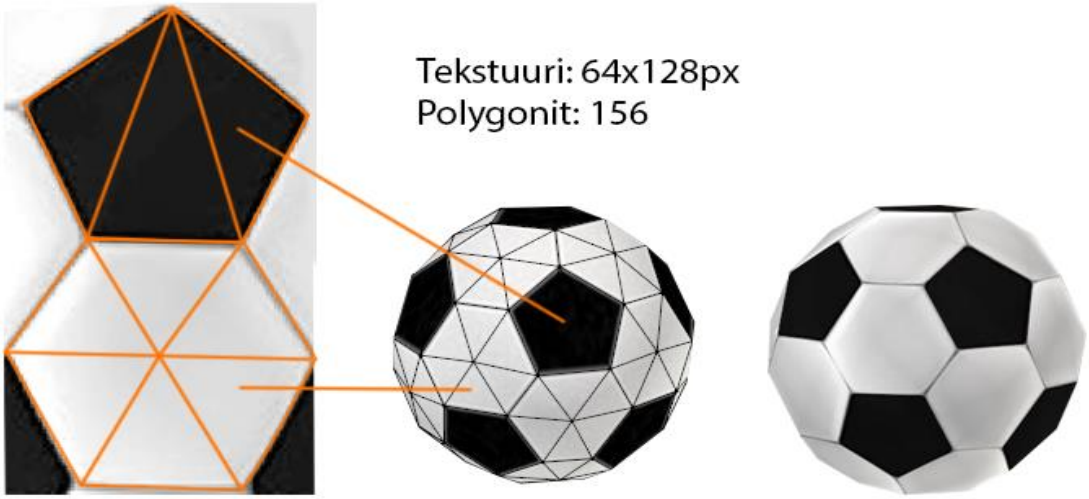
4.2 Teksturoitavan mallin analysointi ja suunnittelu

Mallinnusvaiheessa tulisi pitää jo alusta alkaen mielessä, kuinka tekstuurikoordinaatit voisi määritellä. Joissain tapauksissa mallinnus on hyvä tehdä jopa täysin tekstuurin ehdoilla. Esimerkkinä käytän tässä jalkapallo-objektia. Jalkapallo koostuu tekstuuriltaan pitkälti mustista ja valkoisista viisi- ja kuusikulmioista, jotka toistuvat sen ympäri tasaisesti. Tästä voidaan tehdä analyysi, ja päätellä tekstuuriksi riittävän yksi valkoinen ja musta tekstuurialue. Koko palloa ei siis kannata määritellä yhdeksi suureksi UV-saareksi, vaan toistuvat kohdat kannattaa määritellä päällekkäin. Tätä varten täytyy jalkapallo kuitenkin mallintaa sen mukaisesti. Vertailukohtana käytän Turbosquid -sivustolta ladattua esimerkkiä, joka on mallinnusta ja teksturointia voisi kutsua perinteiseksi. Tämä malli sisältää 360 polygonia ja sen tekstuuri on jopa 4000 pikseliä leveä. Tällaisissa esimerkeissä on kuitenkin aina hyvä pitää mielessä, että kuitenkin mallin lopullinen käyttötarkoitus määrittelee tarpeellisen määrän geometriaa ja tekstuuri- pinta-alaa.

Kuva 5. Esimerkki teksturoidusta jalkapallosta. (Turbosquid 2013)

Tässä esimerkissä (kuva 5) tekstuurikoordinaatit on määritelty niin, että jokainen polygoni saa uniikin alueen tekstuurista, eikä päällekkäisiä alueita ole määritelty. Tämä tyyli ei itsessään ole huono, mutta sen voisi tehdä huomattavasti optimaalisemmin, mikäli mallia käytettäisiin esimerkiksi pelituotannossa. Jos mallinnettaisiin pallo suoraan viisi-

ja kuusikulmioiksi, joista pallo oikeasti muodostuu, voidaan jokaiselle tällaiselle polygoniryhmälle määrittää samat tekstuurikoordinaatit. Tämä tarkoittaa kuitenkin sitä, että pallo tulee olemaan kauttaaltaan tekstuureiltaan toistuva. Tekstuuriin voidaan tietysti aina määritellä erillinen alue, jossa toistuvuus ja symmetria voidaan rikkoa tarvittaessa. Turbosquid -sivustolta ladatusta tekstuurista olen nyt rajannut vain yhden valkoisen kuusikulmion ja mustan viisikulmion tekstuuriksi.

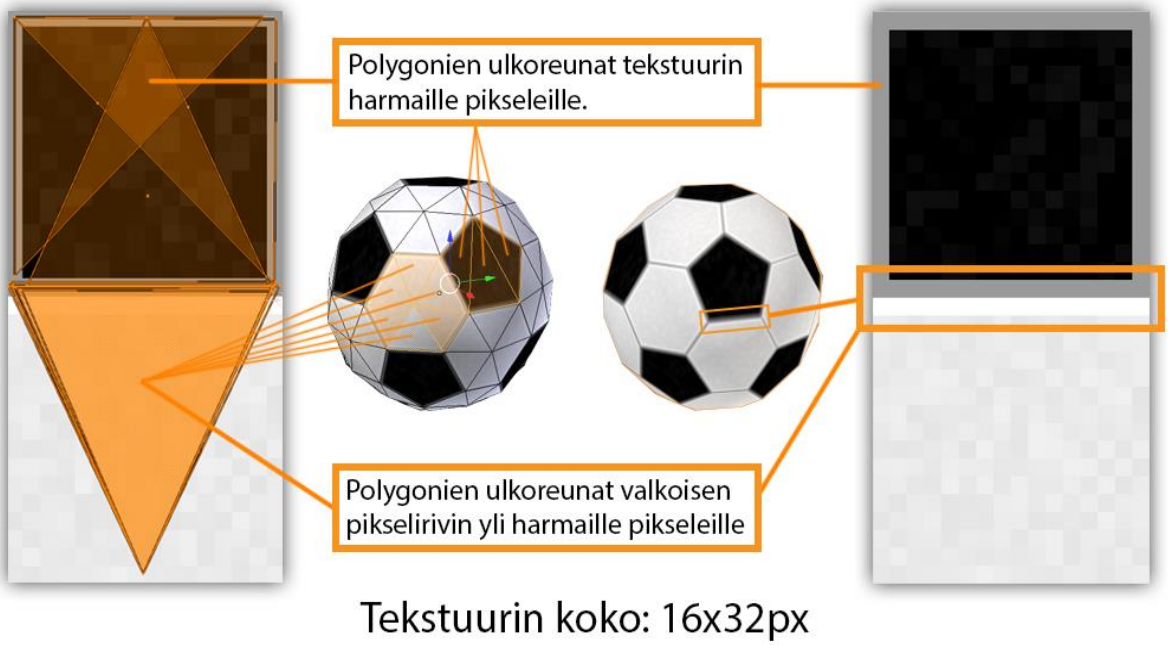


Kuva 6. Rajattu tekstuuri ja uusi mallinnus.

Pallon kaikki valkoiset kuusikulmiot on siis määritelty samaan kohtaan tekstuuria samalla tavalla kuin mustat viisikulmiot mustaan alueeseen. Tällöin nämä alueet toistavat samaa tekstuuria. Lähes samankaltaiseen lopputulokseen siis päästiin vain 64x128 kokoisella tekstuurilla, joka pelituotannossa on merkittävästi parempi vaihtoehto 4000 pikselin kokoisen tekstuurin sijaan. On tietenkin pidettävä myös mielessä pelin kannalta se, mihin mikäkin malli tulee lopulta pelin sisällä sijoittumaan.

Omassa tapauksessani jalkapallo on vain koriste-esine pelin sisällä, eikä sillä ole niin merkittävää roolia pelimekaniikan tai visuaalisen presentaation kannalta. Pallon tulee kuitenkin olla dynaaminen ja sen täytyy olla pyöriteltävissä, eli palloa ei voi korvata pelkällä litteällä geometrialla kamerakulman suhteen. Olennaista on myös, missä koossa malli tulee ruudulla lopulta esiintymään. Jos jalkapallo tulee näkymään esimerkiksi taustan maisemissa vain korkeintaan muutaman kymmenen pikselin kokoisena, on sille turhaa uhrata liikaa tekstuuritilaa ja muistia. Tein siis tästä jalkapallosta vielä äärimmilleen menevän optimoidun tekstuurin. Tässä esimerkissä tekstuuri on itse maalattu. Geometria on täysin sama, mutta tekstuuri ja sen koordinaatit on määrittely uu-

siksi. Tämä esimerkki on jo melko äärimmilleen viety tapaus, ja on hyvä huomauttaa, että tuotannossa edellinenkin esimerkki olisi jo luultavasti riittänyt. Seuraava esimerkki on kuitenkin tekniikan opetusmielessä erinomainen ja esittää tekstuurikoordinaattien tehokasta määrittelemistä entistä optimaalisemmalla tavalla samassa geometriassa.



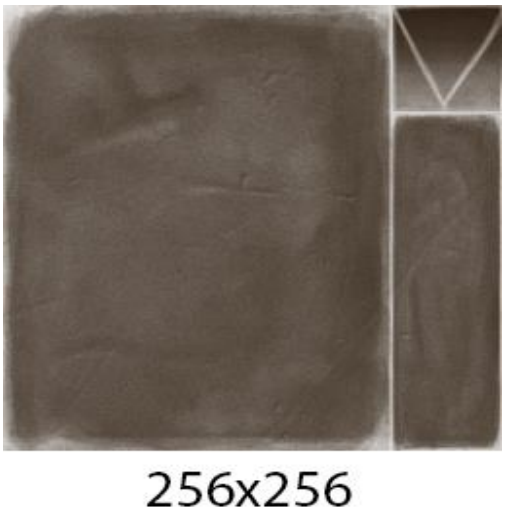
Kuva 7. Esimerkki äärimmäisen pienestä tekstuurista ja uv-koordinaattien määrittelystä

Tällaisessa teksturoinnissa mennään jo pikselitason tarkkuuteen, jossa kahdella tai kolmella eri värillä voidaan luoda hienovarainen varjostus polygonien reunoille tekstuurifilteröinnin ansiosta. Tässä esimerkissä valkoiset polygonit on määritelty tekstuurikoordinaatistoon niin, että jokaisen ulkoreuna ylittää valkoisen pikselirivistön tumman-harmaille pikseleille. Tällöin tekstuurin filteröityessä pelimoottorissa siitä muodostuu hienovarainen gradientti. Muutamalla pikselillä saadaan siis jalkapallon muodot helposti näkyviin. Efektiä voisi vielä korostaa vaikka toisella täysin valkoisella ja mustalla pikselirivistöllä, jolloin saumat vaikuttaisivat entistä syvemmiltä. Tämä esimerkki teksturoinnista on jo melko äärimäinen, ja vaatii vastapainona enemmän aikaa unwrappauksessa ja mallinnuksessa. Kuvan 6 esimerkissä päästään hieman helpommalla ja todennäköisesti tarpeeksi tehokkaalla teksturoinnilla tyydyttävään lopputulokseen. Tekniikan kannalta on kuitenkin hyödyllistä ymmärtää, mihin kaikkeen voidaan päästä erittäin pienillä tekstuureilla.

4.2.1 Mallinnus tekstuurin ehdoilla

Edellinen jalkapallo-objekti edustaa myös hyvin esimerkkiä mallinnuksesta tekstuurin ehdoilla, mutta esitän tästä vielä muitakin esimerkkejä. Seuraavassa esimerkissä (kuva 8) olen luonut kokoelman erilaisia ja eri kokoisia kivi- ja kallionlohkareita, jotka käyttävät kaikki samaa tekstuuria. Tavoitteena näissä objekteissa on ollut luoda variaatioita kivilohkareista ilman, että jokainen kivi tarvitsisi oman tekstuurin. Tällöin samaa tekstuuria käyttämällä siis optimoidaan suoraan myös drawcall -määriä. Jokainen kivi koostuu noin 70 - 110 polygonista. Tärkeä huomio on myös se, että jokainen rako kalliossa on mallinnettu geometriana, eikä niitä ole maalattu tekstuuriin. Näin saadaan hieman realistisempi lopputulos muutaman lisäpolygonin hinnalla. Erilaisia variaatiota tällä mallintamistekniikalla voisi luoda periaatteessa loputtomasti.

Kuva 8.

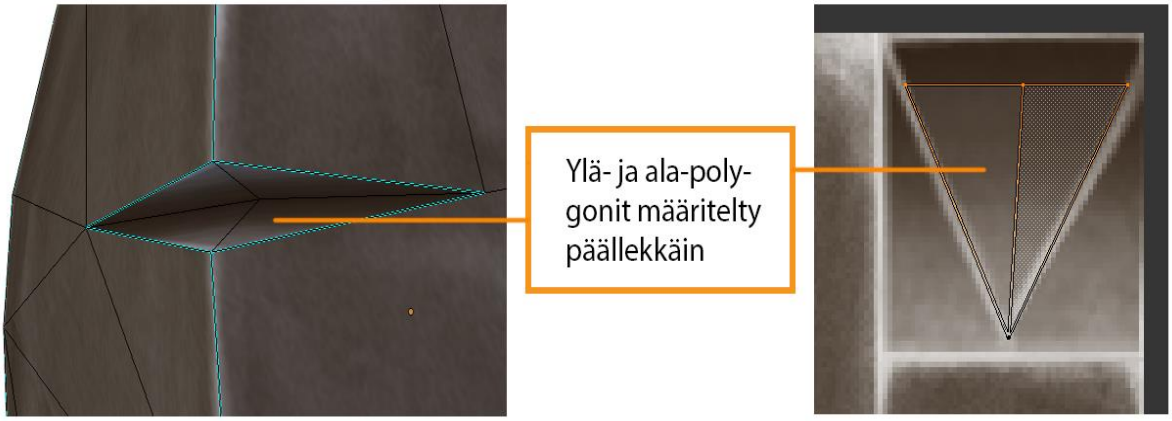


Kuva 9.

Tämä myös mahdollistaa raon sijoittamisen mallinnusvaiheessa mihin kohtaan vain, eikä sen tarvitse olla sidottuna tekstuuriin. Tässä mallinnus on siis selkeästi mietitty käytettävää tekstuuria ajatellen. Jokainen tasainen lohkareen sivu käyttää samaa tekstuurialuetta käännettynä ja venytettynä eri asentoihin. Kivien kaikki sivut on määritelty tekstuuriin tällä samalla tavalla. Vaikka polygonialueet olisivat eri muotoisia ja eri suhteessa kuin tekstuuri, voi niitä melko surutta tässä tapauksessa skaalata sopiviksi tekstuuriin. Tällöin muodostuu tietysti venymistä tekstuuripinnassa, mutta orgaanisessa pinnassa sen havaitseminen on vaikeampaa eikä niin häiritsevää. Jotkut sivut malleissa ovat kuitenkin muita huomattavasti kapeampia, jolloin näille on varattuna tekstuurissa oikeanpuoleinen kapeampi alue. Näin saadaan tekstuuri pysymään venymisen kannalta suurin piirtein samassa skaalassa.

Kuva 10. Kivien tekstuurikoordinaatit.

Variaation muodostumiseksi tekstuurikoordinaatteja on myös käännetty 90-180 astetta oikealle tai vasemmalle ja peilattu Y- ja X-koordinaatin suunnassa. Maalatessa tällaista tekstuuria on tärkeää tehdä siitä tarpeeksi geneerinen ilman silmäänpistäviä yksityiskohtia, jotka toistuisivat kiven jokaisessa sivussa. Kivilohkareiden raot on mallinnettu ja niille on varattu pieni alue tekstuurista. Tekstuurialue muodostaa rakoon hienovaraisen gradientin tummasta sisäraosta vaaleaan reunaan ja korostaa näin sen syvyyttä. Jokainen rako kivessä on muodostettu tällä tavalla ja näin saadaan rikottua myös kiven muotoa tehokkaasti.



Kuva 11. Kivirakojen tekstuurikoordinaatit.

Selkeänä etuna tässä tekniikassa on tietysti se, että kiven jokainen sivu saa käytettäväkseen lähes koko tekstuurin pinta-alan. Näin säästetään tekstuurin koossa ja säilytetään silti suhteellisen korkea laatu tekstuurin kuvanlaadussa. Olennaista on siis aina pitää mielessä toistuvien osien määritteleminen tehokkaasti.

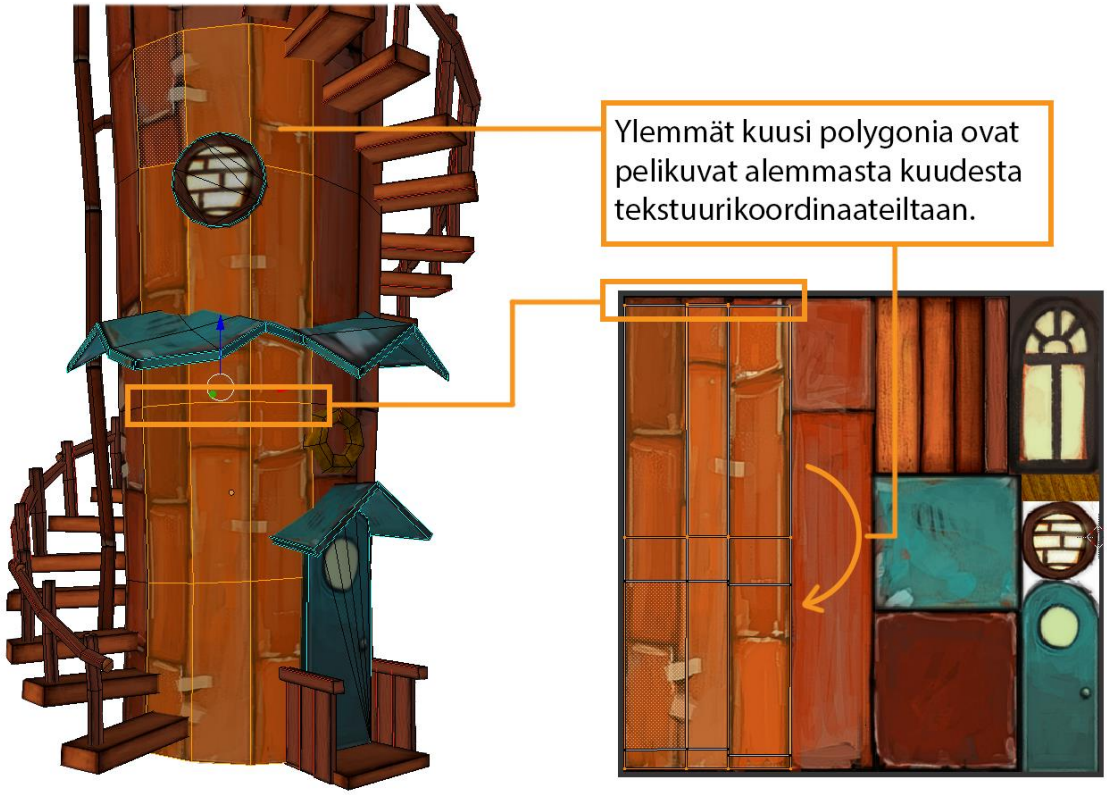
Seuraavassa esimerkissä olen teksturoinut kokonaisen rakennuksen käyttäen mahdollisimman paljon toistuvia alueita tekstuurista. Olen pyrkinyt myös välttämään selkeitä saumoja tai häiritsevää toistumista läheisillä alueilla. Tässäkin tapauksessa mallinnus on osittain tehty tekstuurin ehdoilla, osittain jopa tekstuurin maalaamisen jälkeen. Tekstuurin voi siis maalata jopa etukäteen ennen mallintamista, mikäli mallista on olemassa valmis ja selkeä konsepti. Tällöin voidaan päätellä valmiiksi tarvittavat materiaalit ja maalattavat tekstuurit jo ennen kuin mallinnusta on edes aloitettu. Tämä tietysti edellyttää suunnitelmaa ja hahmotusta mallin muodostumisesta etukäteen tekstuurikoordinaattien ja topologian kannalta. Tämä taito harjaantuu melko nopeasti useamman tällä tavalla laaditun mallin jälkeen, jolloin myös koko prosessi omassa tuotannossa nopeutuu. Mallinsin tässä esimerkissä talon rungon vasta tekstuurin maalaamisen jälkeen, jolloin pystyin kontrolloimaan lankkujen saumakohtia polygonitasolla, eli jokainen lankku vastaa suurin piirtein yhtä polygonia leveydeltään. Tällä keinolla pystyn varmistamaan tekstuurin toistuvuuden tasaisesti talon ympäri niin ettei lankun sauma osuisi keskelle polygonia. Tällöin tekstuuria toistaessa saumat jäävät siis sopivasti piiloon lankkujen väliin. Toinen hyvä vaihtoehto olisi voinut olla mallintaa jokainen lankku omana polygoninaan, jolloin variaatio tekstuurissa olisi ollut vielä suurempi. Tällöin kuitenkin polygonimäärät olisivat voineet nousta liian suuriksi. Kuvassa 12 lopullinen 3d-malli rakennuksesta ja sen käyttämästä tekstuurista. Kuvaa tarkastelemalla voi myös

huomata monesta kohtaa, kuinka samaa tekstuurikohtaa on käytetty eri talon osissa. Yksi toistuvista kohdista sekä tekstuuriltaan ja geometrialtaan ovat rappusten askelmat. Jokainen rapuista käyttää samaa puu-tekstuuria, mutta vaikutelma toistuvuudesta ei ole siltikään liian häiritsevä. Variaation lisäämiseksi tekstuuriin on maalattu kuitenkin useampi eri sävy puun pintatekstuurista. Näitä käyttämällä oikeissa paikoissa voidaan luoda tehokasta variaatiota ja välttyä liialta toistumiselta.

Kuva 12.

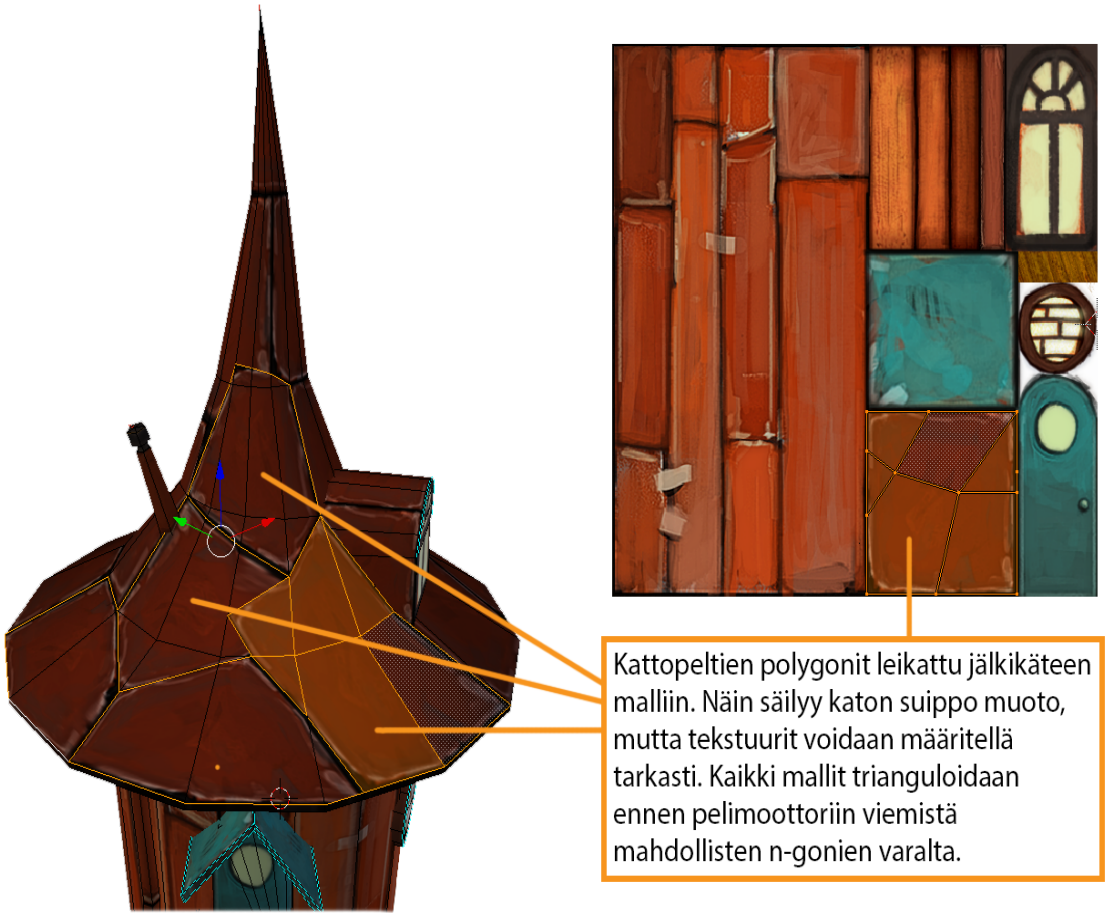
Talon rungossa on hyvä esimerkki kuinka peilattuja tekstuureita on käytetty hyväksi saumattomasti. Peilaamista ja kääntämistä hyväksikäyttäen voidaan luoda variaatiota rungon pintaan suhteellisen pienilläkin tekstuurialueilla. Koko runko käyttää siis vain tekstuurin vasemmanpuoleista neljää riviä lankkuja eri asennoissa. Tämä jälleen edellyttää mallintamista tekstuurin ehdoilla. Vaikka tekstuuri itsessään ei ole saumaton, voi

sitä käyttää samalla tavalla mustien rakojen jakaessa lankut sopivasti toisistaan. Seuraava kuva havainnollistaa kuinka peilaus on toteutettu rungon alaosassa.



Kuva 13.

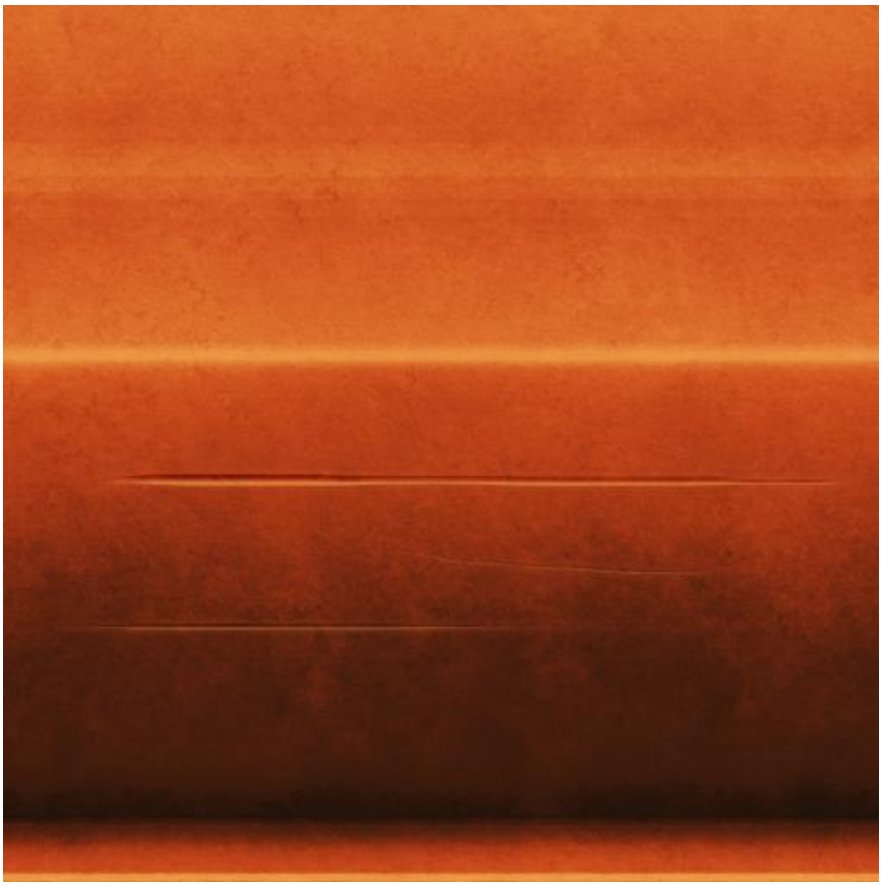
Talon katon mallinnus eteni ensin suipon muodon tekemisellä, jonka jälkeen leikkasin siihen karkeasti kattopeltien määrittämät nelikulmiot. Lopputulos voi vaikuttaa karulta topologian suhteen, mutta muutama näennäisesti ylimääräinen polygoni vaikuttaa suorituskykyyn hyvin marginaalisesti. Tällä tavalla säilytin myös katon suipon muodon alkuperäisenä. Tämän jälkeen määritin jokaisen kattopellin tekstuurikoordinaatit samalla tavalla kuin kivilohkare-esimerkissä. Jokainen kattopeli toistaa siis samaa tekstuuria eri kulmassa tai peilattuna X tai Y koordinaatissa. Leikkaamalla malliin tällä tavalla nopeasti uusia polygoneja, saattaa geometriaan jäädä joitain n-goneja, jotka on tärkeää muuttaa kolmioiksi tai neliöiksi ennen pelimoottoriin viemistä. Tämäkin tapa teksturoida on siis jälleen mallintamista tekstuurin ehdoilla, jotta kattopellit voidaan muotoilla halutulla tavalla. Tämä tyyli antaa enemmän vapautta luoda pinnoista lukuisia erilaisia variaatioita, mutta jättää jälkeensä enemmän polygoneja. Kuva 14 havainnollistaa kuinka olen muodostanut katon lopulliset pellit ja mihin tekstuurien koordinaatit sijoittuvat.



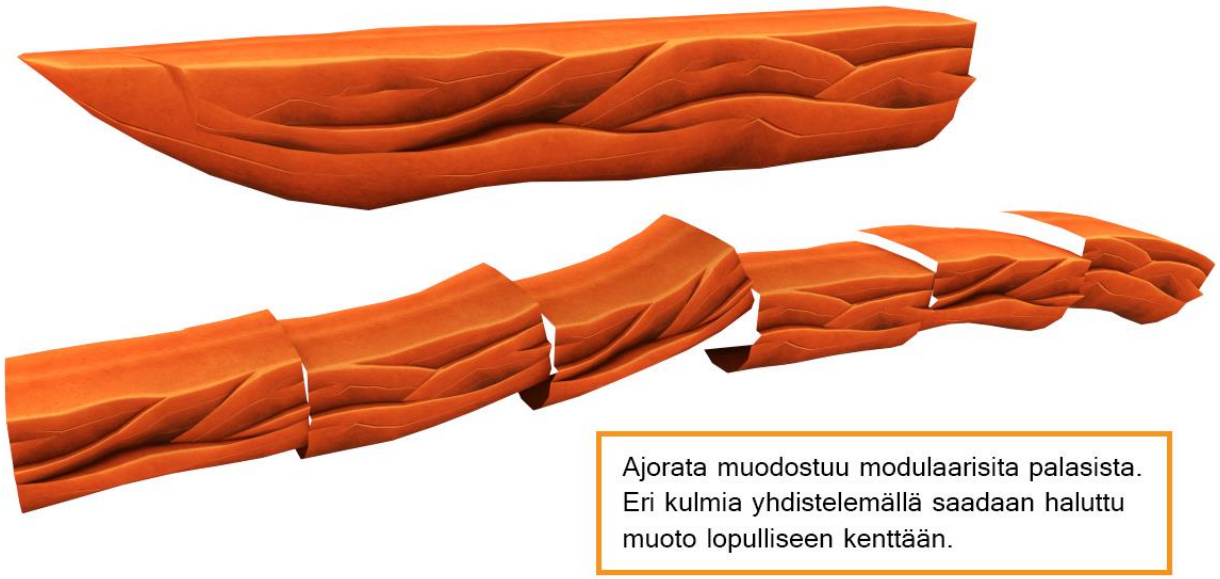
Kuva 14. Kattopeltien geometria.

Viimeisenä esimerkkinä tämänkaltaisesta mallinnuksesta ja teksturoinnista esitän pelin aavikko-teemaisen ajorataan ja ympäristöön liittyvät objektit ja niiden tekstuurit. Tässäkin ajatuksena on ollut tekstuurikoordinaattien määritteleminen mallinnuksen ehdoilla, vaikka tekstuuri on maalattu jo etukäteen.

Trials Frontierin ajoradat on tehty myös samankaltaisilla tekniikoilla kuin edelliset esimerkit. Ne muodostuvat palasista, jotka ovat toisiinsa nähden modulaarisia ja mahdollisimman saumattomia. Palasia on eri käyryysasteina ja niistä rakentamalla saadaan lopullinen ajorata kenttään. Kaikki palaset käyttävät myös samaa tekstuuria. Vaihtelua palasiin on tehty jälleen geometriaa muokkaamalla, ei erilaista tekstuuria maalaamalla. Näiden lisäksi samaa tekstuuria käyttävät myös etu- ja taka-alan kalliogeometriat.

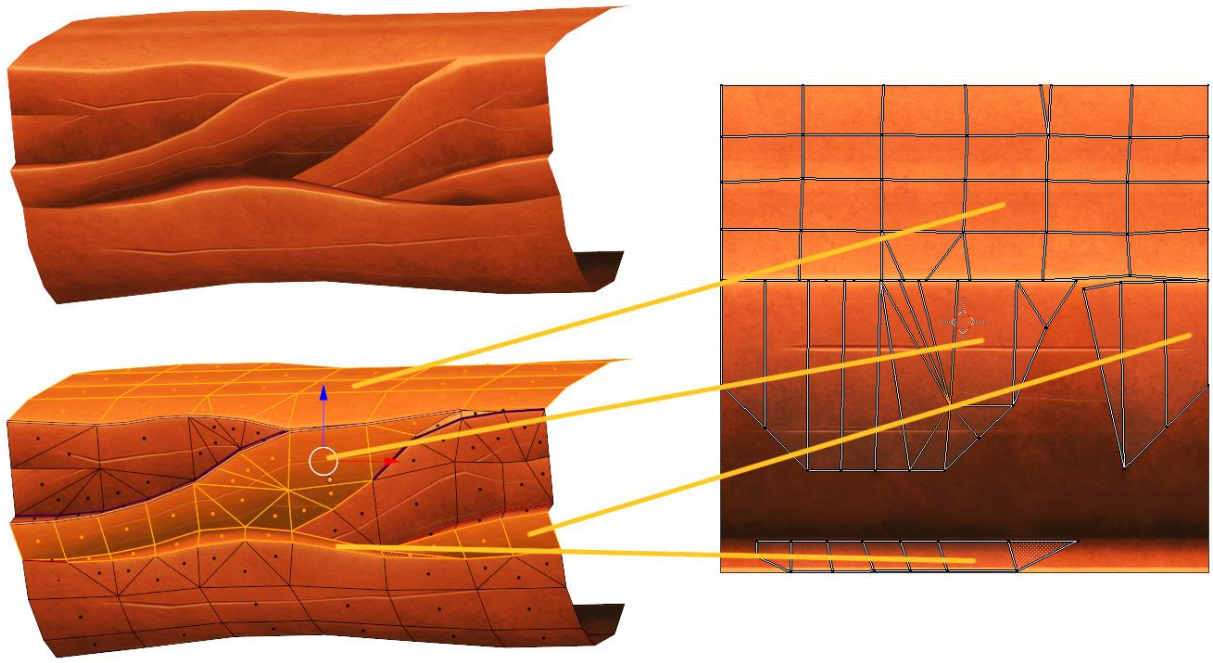


Kuva 15. Kalliomuodostelmien käyttämä tekstuuri

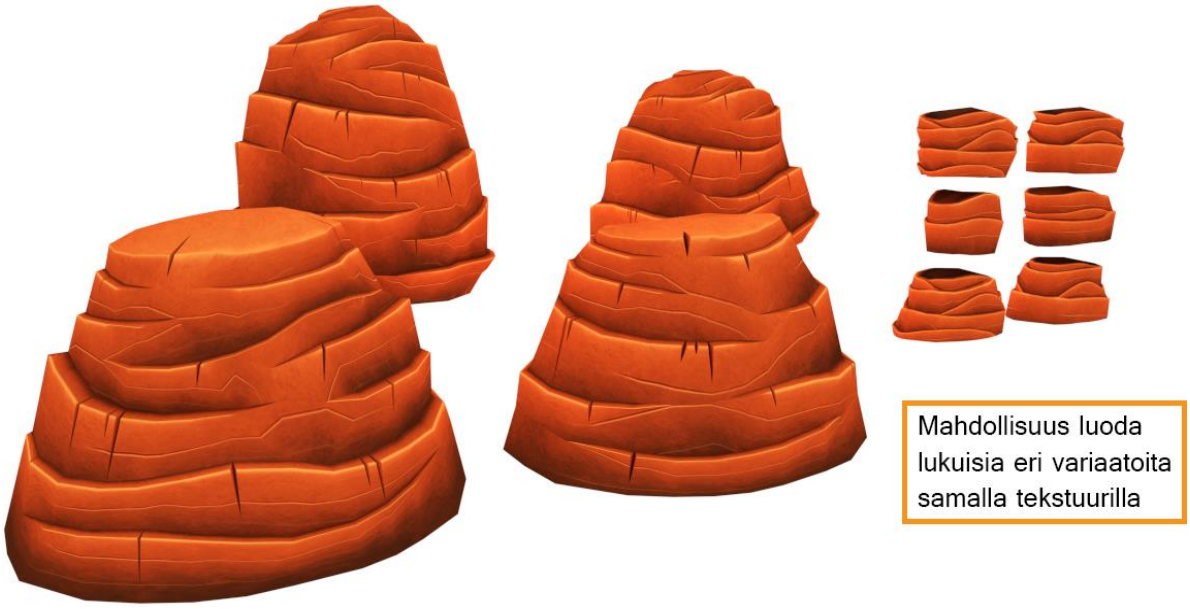


Kuva 16. Ajoradan eri palaset.

Otan tarkasteluun yhden ajoradan palasen tekstuurikoordinaattien havainnollistamiseksi. Kuten aikaisemmissa esimerkeissä, myös tässä tapauksessa mallinnus tekstuurikoordinaattien määritteleminen on pyritty saada sovitettua tekstuuriin jälkikäteen. Tämä siis edellyttää jälleen selkeää suunnitelmaa tai konseptia lopullisesta mallista, josta tiedetään etukäteen kuinka se tullaan teksturoimaan jo valmiilla tekstuurilla. Näiden kalliomuodostelmien kanssa oli siis jo selkeä visio, kuinka tekstuurikoordinaatit voitaisiin määritellä jokaiselle palaselle ja kielekkeelle. Samalla tavalla kuin kivitekstuurin kanssa, myös tässä tapauksessa muodostuu tekstuuriin lievää venymistä ja paikoittaista saumojen näkymistä, mutta lopputulos on silti tyydyttävä. Tekstuuri on maalattu alusta alkaen silmälläpitäen kallion muodostelmia. Tekstuuri toistuu myös saumattomasti vaakasuunnassa. Tekstuuriin on varattu tilaa kallion syvennyksiin, ulokkeisiin sekä teräville reunoille. Tällaisella tekstuurilla voidaan mallintaa teoriassa rajattomasti erilaisia variaatioita samankaltaisista kalliomuodostelmista ja määritellä vain tekstuurikoordinaatit oikeisiin paikkoihin. Tällä tekniikalla onkin muodostettu teemaan sopivasti samankaltaiset taustan ja etu-alan kalliot.



Kuva 17. Ajoradan tekstuurikoordinaatit.



Kuva 18. Erilaiset kalliomuodostelmat.

4.2.2 Materiaalitekstuurit

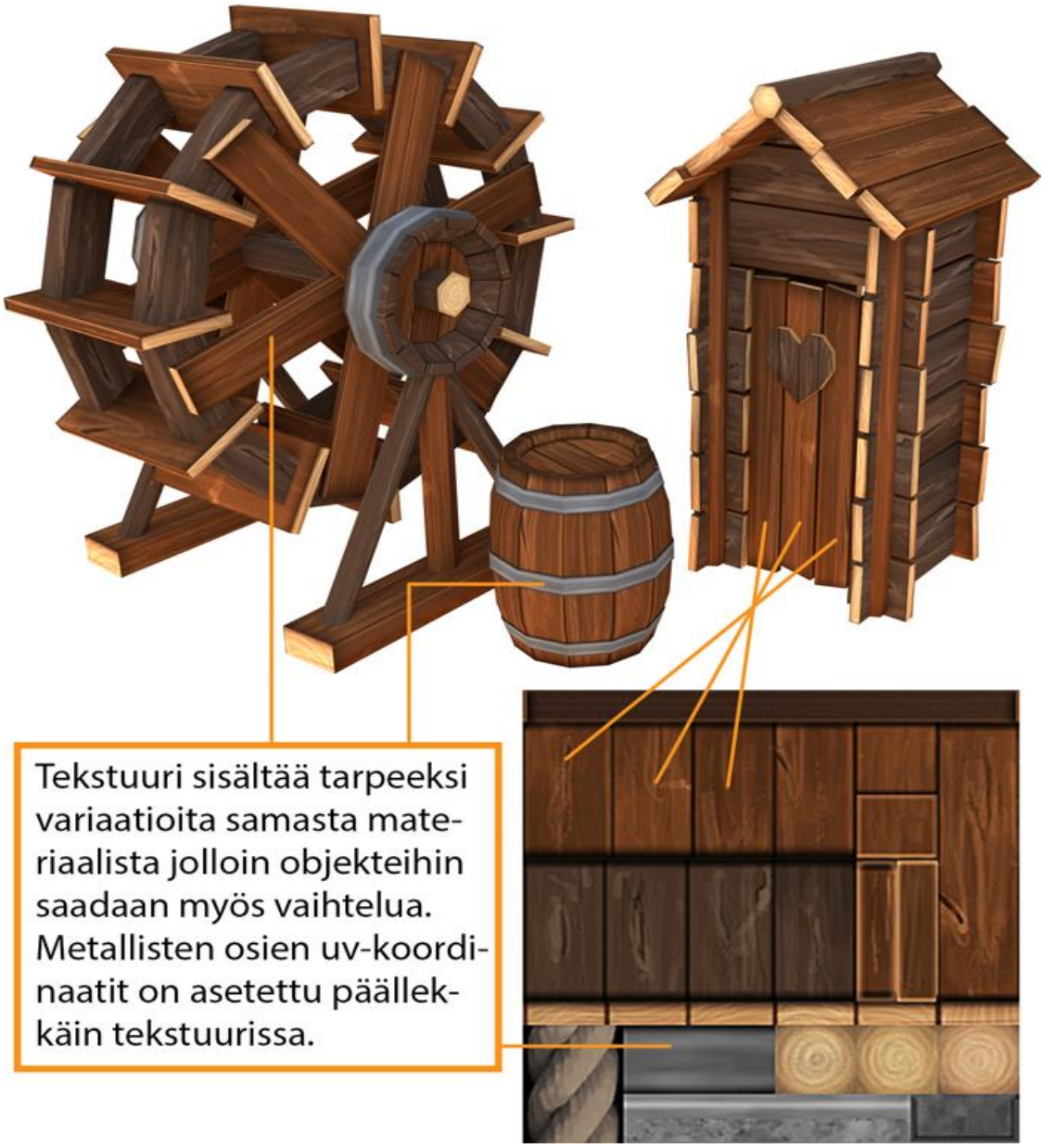
Joissain tilanteissa on myös hyödyllistä laatia erillinen tekstuuri samankaltaisille ja samaa materiaalia käyttäville objekteille. Tällainen esimerkki voisi olla kokoelma erilaisia puisia tai metallisia esineitä, jotka vaihtelevat muodoltaan, mutta eivät materiaaliltaan. Käytän tällaisista tekstuureista tässä opinnäytetyössä omaa termiäni ”materiaalitekstuurit”. Samankaltaisille puuobjekteille voidaan siis tehdä yksi suuri puutekstuuri, joka sisältää lukuisia pintavariaatioita samasta tai samanlaisesta puusta. Olen seuraavassa esimerkissä mallintanut erilaisia puisia esineitä käyttäen samaa tekstuuria jokaisessa. Jälleen kerran optimoidaan drawcall-määriä, kun yksi tekstuuri tarvitaan kutsua vain yhden kerran lukuisaa objektia kohden.

Kuva 19. Erilaiset objektit jotka käyttävät samaa tekstuuria.

Tällaisella tekniikalla voidaan mallinnus viedä ikään kuin rakennusalan puolelle, jolloin voidaan rakentaa puisia esineitä valmiista mallinnetuista lankuista ja pölkyistä modulaarisesti. Jokaista lankkua ei siis kannata lähteä mallintamaan uudestaan uuden objektin kohdalla, vaan voidaan kopioida valmiiksi mallinnettuja lankkuja joista on jo olemassa monia variaatioita. Kokonaisia rakennuksia ei ehkä tällä tekniikalla kannata mallintaa suuren polygonimäärän vuoksi, mutta pienemmät objektit muodostuvat näppärästi. Kuvassa 18 on kokoelma erilaisia esineitä ja rakennelmia, jotka olen muodostanut melko pitkälti samoista lankuista skaalaamalla ja sijoittamalla niitä sopiviin asentoihin. Lankkujen tekstuuri vaihtelee sen mukaan mihin sen koordinaatit on määritelty. Kuvassa 20 tekstuuri, jota kaikki nämä objektit käyttävät.

Kuva 20. Puutekstuuri.

Otan näistä objekteista tarkasteltavaksi huussin, tynnyrin ja vesimyllyn. Huussin ja vesimyllyn olen mallintanut lähes täysin modulaarisesti jälkikäteen valmiista teksturoiduista lankuista poikkeuksena muutamat yksityiskohdat, kuten huussin oven puinen sydän ja myllyn metallivahvistettu runko-osa. Tällaisen tekstuurin ja mallintamisen yhdistelmällä voidaan luoda suuria määriä erilaisia objekteja nopeasti ja helposti. Mikään ei tietenkään estä lisäämään tekstuuriin muitakin materiaaleja, mutta on optimaalisempaa rajata materiaalit niihin kategorioihin, joita mahdollisimman moni objekti tulee varmasti käyttämään samanaikaisesti. Etukäteen konseptoidut ideat ja objektit auttavat graafikkoa miettimään mitä kaikkia materiaaleja voisi ryhmittää samaan tekstuuriin. Näin saadaan tekstuureista kustannustehokkaita ja säästetään pelin suorituskyvyssä.



Kuva 21. Kohdat tekstuurissa joita objektit käyttävät.

4.3 Alphakanava ja kasvisto

Alphakanavan ongelmallisuus tulee esiin Deferred Rendering –tekniikan yhteydessä ja se asettaa haasteita erityisesti kasviston mallinnuksessa ja teksturoinnissa. Normaalitilanteessa olisi esimerkiksi lehden mallintaminen helppoa yhdellä polygonilla ja tekstuurilla, joka sisältää alphakanavan läpinäkyvyyden kontrolloimiseen.

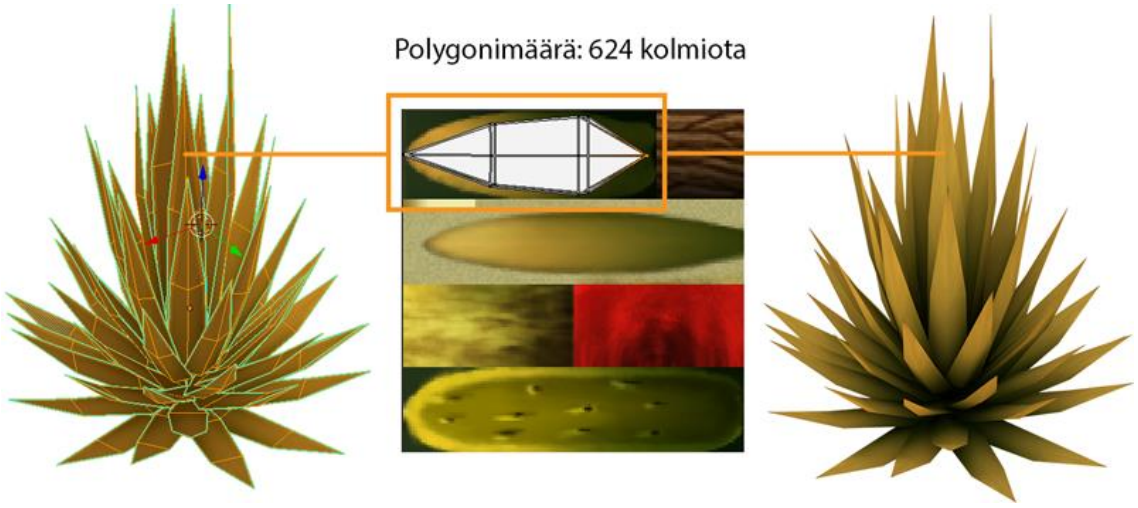
Kuva 22. Esimerkki alphakanavan käytöstä tekstuurissa.

Tämä keino on yleisin ja tehokkain tapa luoda ympäristö, joka sisältää paljon kasvistoa. Mobiilipuolella Deferred Rendering -tekniikan kanssa tähän ei kuitenkaan yleensä ole tehojen puolesta varaa, joten on keksittävä muita keinoja tyydyttävään lopputulokseen ilman alphakanavaa. Tämä oli myös tilanne itselläni Redlynxillä työharjoittelun aikana käytettävissä olevan Genetek-pelimoottorin kanssa. Ratkaisu ongelmaan on lopulta niinkin yksinkertainen kuin mallintamalla lehdet kokonaan polygoneina. Kuten aikaisemmin jo toin esille luvussa 3, polygonimäärät eivät aina ole kriittisin ongelma pelituotannossa pelin suorityskyvyn kannalta. Käytännössä voimme siis kohtuuden rajoissa luoda kasveja sekä lehtiä täysin polygoneina ilman alphakanavan käyttämistä.

Kuva 23. Kokoelma erilaisia kasveja

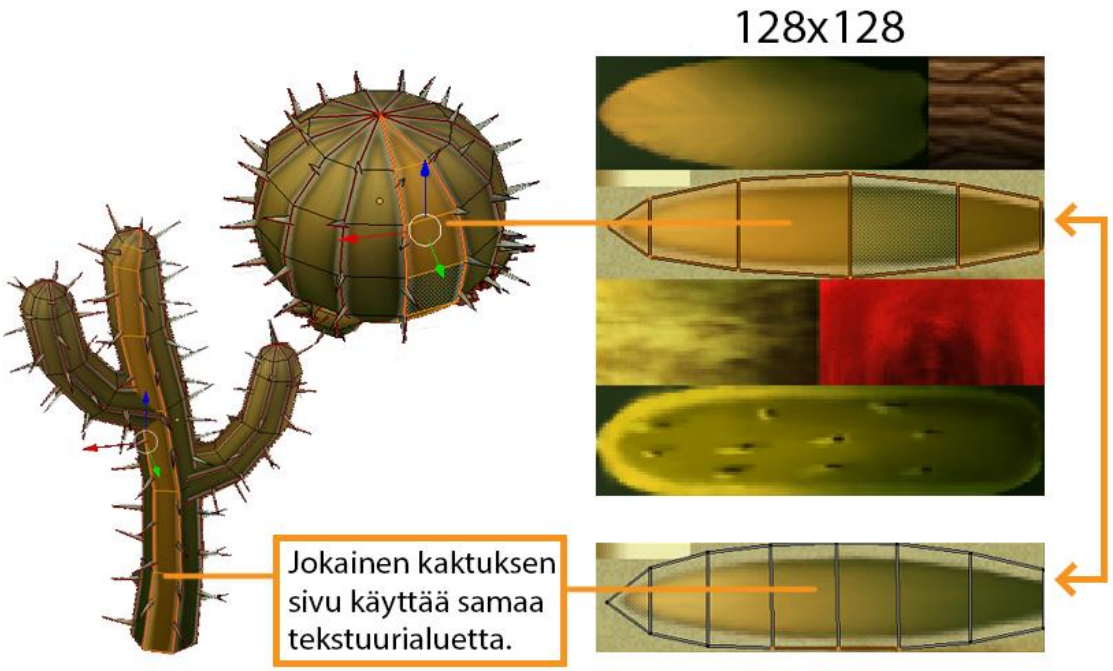
Mallinsin pelin aavikko-teemaan kokoelman erilaisia kasveja, jotka jakavat kaikki saman tekstuurin eivätkä käytä alpha-kanavaa. Kaikki lehdet ja kaktuksien piikit ovat siis polygoneja. Ratkaisu voi vaikuttaa suurelta polygonimäärien hukkaamiselta, mutta lopputulos on suorituskyvyn kannalta vieläkin edullisempi kuin alpha-kanavan käyttämi-

nen. Kokonaisia viidakoita ja metsiä tähän tapaan ei ole järkeä tietenkään tehdä, mutta kohtuullinen määrä polygonilehtiä viljeltynä satunnaisiin paikkoihin auttaa lisäämään realismia.



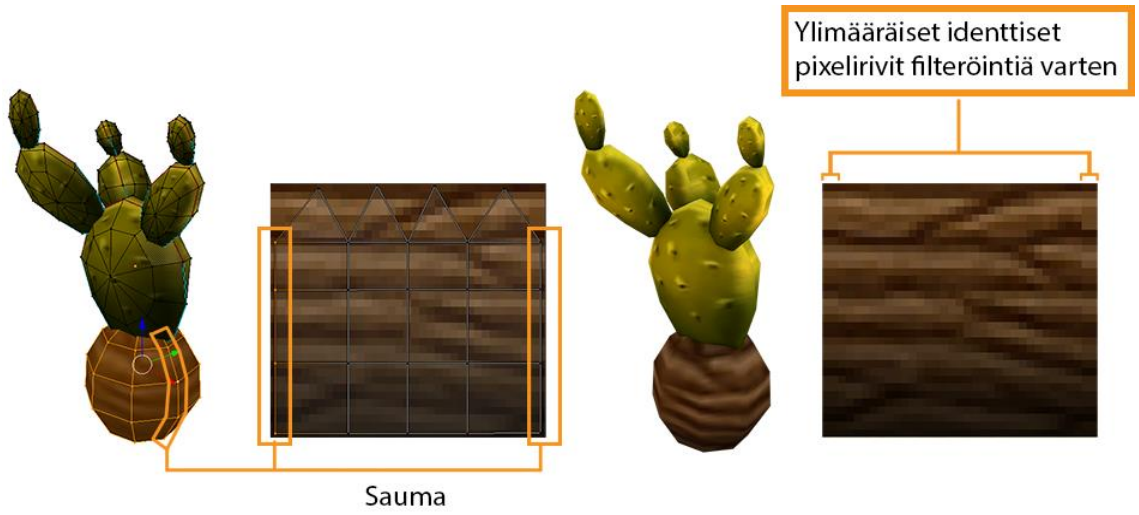
Kuva 24. Kasvin tekstuurikoordinaatit.

Otan jälleen tarkempaan tarkasteluun muutaman kasvin topologian ja tekstuurikäytön havainnollistamiseksi. Erityistä huomiota ansaitsevat kaktukset ja niissä toistuva tekstuuri. Kaikki kaktukset on teksturoitu lähes täysin samalla tavalla, mutta muodot poikkeavat toisistaan.



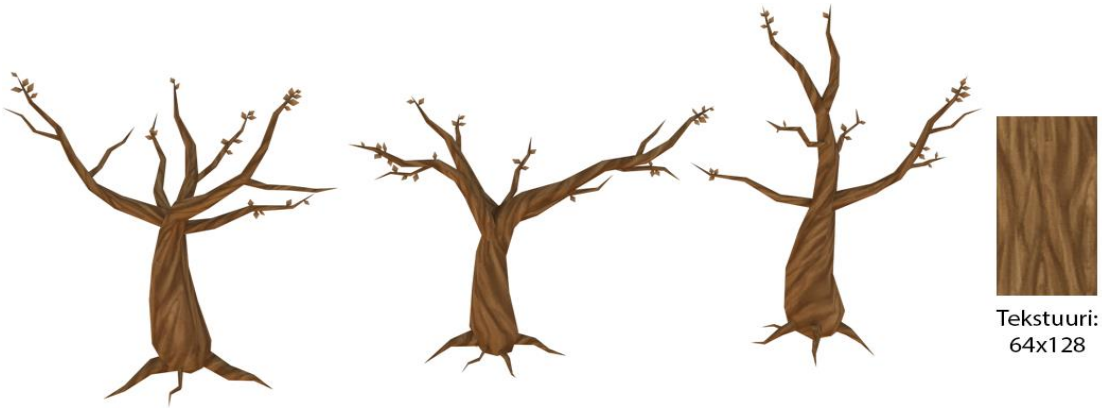
Kuva 25. Kaktuksen tekstuurikoordinaatit.

Tekniikka on samankaltainen kuin kivilohkare-esimerkissä. Nopean analyysin jälkeen kaktuksistakin voidaan päätellä toistuvat alueet ja suunnitella sen perusteella tekstuuri sekä mallinnus. Kaikkiin kaktuksiin tällä tekniikalla riittääkin jälleen erittäin pieni tekstuuri. Tässä tekstuurissa on myös esimerkki erillisestä saumattomasta tekstuurialueesta tekstuurin sisällä. Tekstuurissa oikeassa yläkulmassa kasvin rungon tekstuurikoordinaatit on määritelty saumattomalle tekstuurialueelle, joka toistuu vaakatasossa. Tekstuurifilteröinnin vuoksi alue vaatii kuitenkin ylimääräiset kaksi identtistä tai saumatonta pikseliriviä kummallekin puolelle, jotta siihen ei vuotaisi väriä tekstuurin toisista laidoista filteröinnin jälkeen.



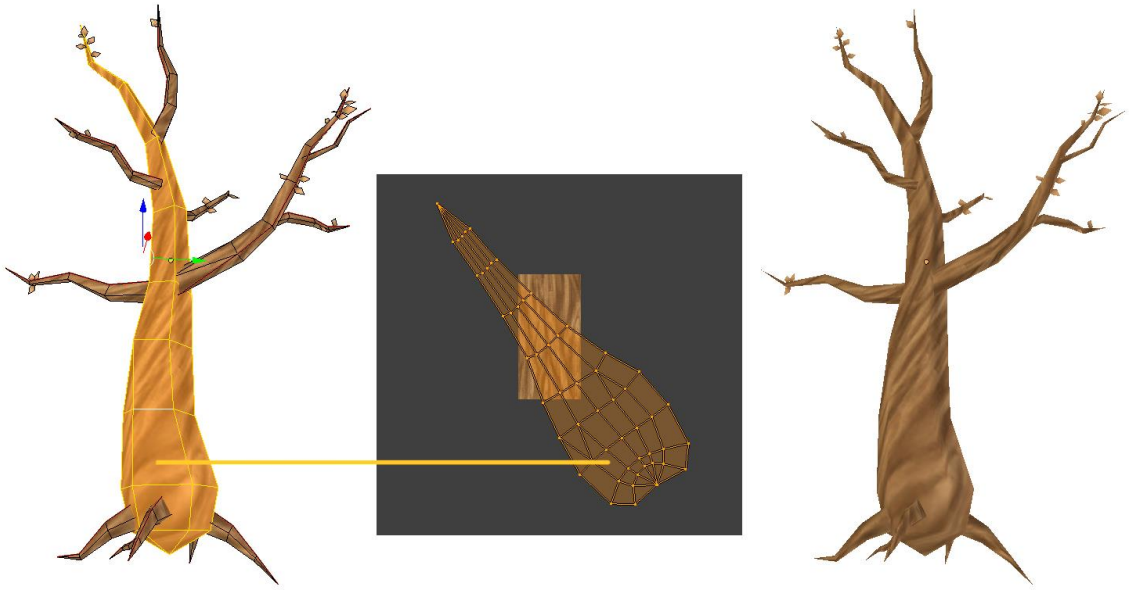
Kuva 26. Saumattoman kohdan tekstuurikoordinaatit.

Täysin saumattomasta tekstuurin käytöstä olen luonut puun, jonka lehdetkin olen teksturoinut karkeasti samalla tekstuurilla. Tällaisen objektin tekstuurikoordinaatit voi määritellä bittikartan rajojen yli, jolloin tekstuuri ei kuitenkaan ole enää atlasoitavissa. Tekstuuri siis toistuu saumattomasti sekä pysty- että vaakatasossa.



Kuva 27. Erilaiset puut samalla tekstuurilla.

Tällaisella tekstuurilla voidaan luoda variaatioita määrittelemällä tekstuurikoordinaatit eri kulmiin ja skaaloihin, sillä tekstuuri on saumaton. Tekstuurikoordinaatit voivat siis olla huomattavasti suuremmat kuin itse tekstuuri, jolloin voidaan kontrolloida tekstuurin tiheyttä objektin pinnalla. Tällä tavalla voidaan myös kontrolloida pinnan spiraalimaisuutta, mikäli se on haluttu lopputulos. Lehdet saavat tässä tapauksessa vain muutaman pikselin sävyt tekstuurista, joka riittää luomaan kuivuneen lehden vaikutelman. Vaihtoehtoisesti lehtien väriä olisi voinut kontrolloida esimerkiksi verteksiväreillä. Tällöin tekstuuriin olisi voinut piilottaa esimerkiksi yhden valkoisen pikselin, johon kaikki lehtien tekstuurikoordinaatit olisi voinut määritellä. Valkoisen värin läpi verteksivärit eivät muutu, joten jos lehtien verteksit olisi värjätty vihreiksi, näkyisivät ne täysin samanvärisinä myös pelissä. Verteksiväreihin palaan vielä tarkemmin luvussa 4.4.

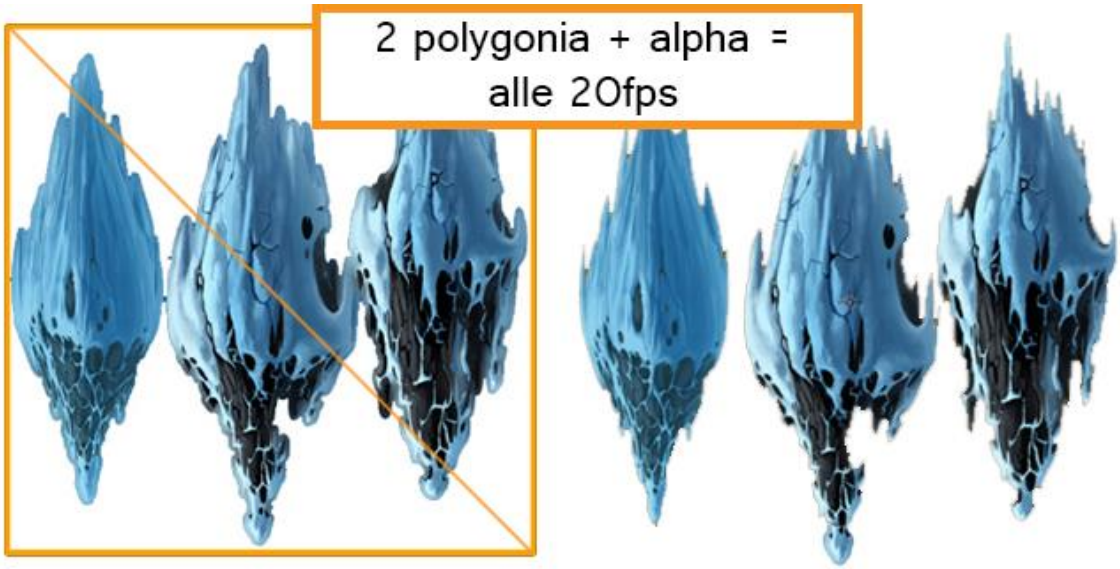


Kuva 28. Puun tekstuurikoordinaatit.

4.4 Alphakanava vs. polycount

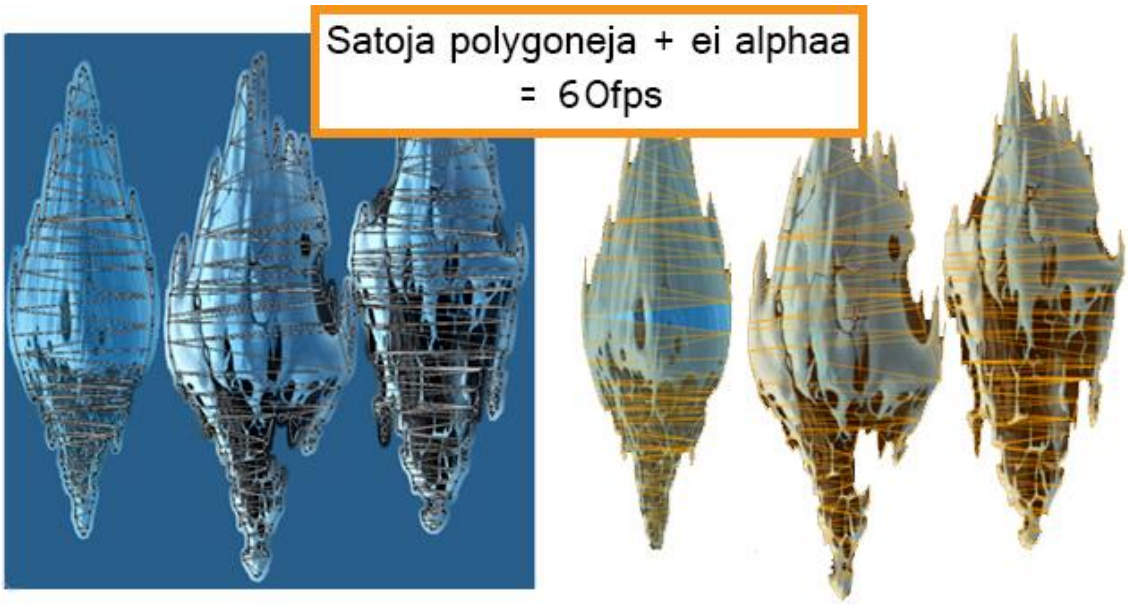
Kuten aikaisemmin on jo todettu, alphakanavan käyttäminen Deferred Rendering –tekniikan kanssa voi olla suorituskyvyn kannalta ongelmallista. Tässä luvussa demonstroin lyhyesti pelissä käytettävän Genetek-moottorilla, kuinka polygoneilla voidaan säästää suorituskyvyssä myös sellaisissa tilanteissa, joissa alphakanavan käyttäminen voisi tuntua luonnolliselta. Tässä esimerkissä kyseessä on Redlynxin edellisestä Motoheroz -mobiilipelistä, joka käyttää samaa Genetek-pelimoottoria. Esimerkissä on ajokentän taustalla leijuvat saaret, jotka ovat omalla tasollaan irrallisina taustakuvasta. Tällä tavalla pystytään luomaan kenttään syvyyttä ja parallaksia kameraliikkeen kanssa. Kuvat ovat maalattuja, eikä objektilla ole oikeaa geometrista syvyyttä. Tällöin luon-

nollisena ratkaisuna voisi käyttää yhtä nelikulmaista polygonia ja alphakanavaa kuvan rajaamiseksi. Yhteensä tässä tapauksessa käytössä olisi kaksi kolmiota ja alphakanava. Kuitenkin alphakanavan raskas vaikutus suorituskykyyn laski fps:n joillain mobiililaitteilla jopa alle kahdenkymmenen. Tässä tapauksessa testissä oli iPhone4S -laite.



Kuva 29. Esimerkki alphakanavan käytöstä saari-objekteissa.

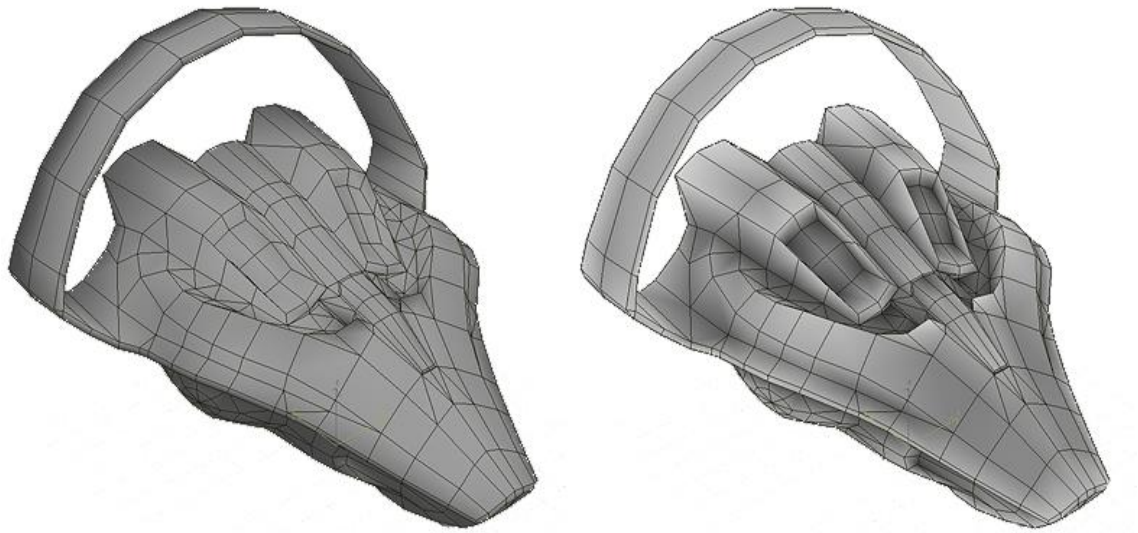
Tällaisessa tapauksessa kiertotie on melko yksinkertainen, vaikka se tuntuisi epäloogiselta. Jokainen saari täytyi siis mallintaa erikseen ääriviivoja myöten geometriaksi, ja teksturoida oikeille kohdilleen. Lopputuloksena on siis huomattava määrä geometriaa, mutta alphakanavaa ei tarvitse käyttää laisinkaan. Suorituskyvyn kannalta ero on merkittävä. Tällä keinolla saatiin peliin jälleen 60fps, joka on Genetek-pelimoottorin kannalta optimaalinen ja maksimi määrä. Sivuhuomiona on myös tärkeää mainita, että Genetek-pelimoottorin tapauksessa moottori ei tue vaihtelevaa fps:ää, eli peli hidastuu samaan tahtiin kuin kuvaruudut vähenevät sekunnissa. Peli siis pyörisi ikään kuin hidastettuna. Käytännössä tämä tarkoittaa sitä, että pelin on pakko säilyttää 60fps aina, mikäli halutaan pelin toimivan oikealla nopeudella. Tämä tapaus alphakanavan käytöstä osoittaa kuitenkin selkeästi sen vaikutuksen suorituskykyyn Deferred Renderin – tekniikan kanssa. On tietenkin tärkeää huomioida, että tulokset voivat vaihdella eri pelimoottoreiden välillä, mutta Genetekin tapauksessa tämä osoitti selvästi, että polygonimääriä ei tarvitse säikähtää monissakaan tapauksissa.



Kuva 30. Saaret ilman alphakanavaa mallinnettuna.

4.5 Verteksiväri, Ambient Occlusion ja värivariaatiot objekteista

Verteksivärjäys ei ole uusi tekniikka 3d-mallien värjäämisessä, mutta sitäkin tehokkaampi suorituskyvyltään. Kävin alustavasti tämän tekniikan läpi luvussa 3.7. Nykyaikaisissa konsolipeleissä verteksivärjäystä voidaan käyttää ambient occlusionin jäljentämiseen tai muun varjostuksen luomiseen. Erityisesti mobiilipuolella mallien värjääminen tai kokonaisen valaistuksen luominen on käytännöllistä.



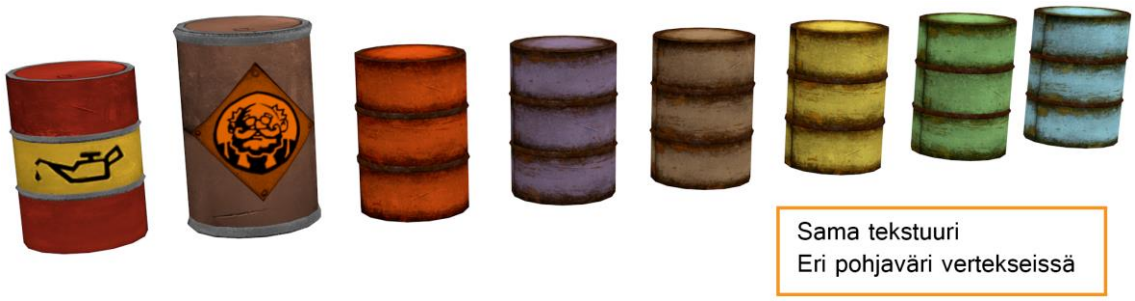
Kuva 31. Vertekseihin voidaan värjätä esimerkiksi varjostusta. Kuvassa esimerkki ambient occlusionin maalaamista suoraan vertekseihin.

Mobiilipuolella tekniikan hyödyntäminen on kannattavaa, sillä kompleksien valojen ja pikselivarjosuodattimien käyttäminen voi olla usein laitteille liian raskasta. Tästäkin johtuen koko Genetek-pelimoottorin valaistus perustuu verteksivärjäys-tekniikkaan. Käytännössä kaikki valot sekä Motoheroz että Trials Frontier pelissä luodaan täysin vertekseihin upottamalla. Kummassakin pelissä valon värit määritetään etukäteen maailmaan ja malleihin jo kentän lataamisen aikana. Jokaiseen verteksiin pelimaailman geometriassa määritellään siis oma väri, joka muodostaa lopulta yhdessä muiden objektien kanssa vaikutelman valaistuksesta. Tämä siis tarkoittaa, että pelissä ei ole dynaamisia valoja tai varjoja lainkaan. Tämän tekniikan ansiosta kuitenkin pelin suorituskyky säilyy korkeana. Tämä myös kannustaa graafikkoa mallintamaan malleihin hieman enemmän polygoneja, sillä tällöin valaistuksestakin tulee tarkempi. (Polycount wiki 2011, Polycount wiki 2013.)

Tämän lisäksi objektien mallinnusvaiheessa voidaan luoda samanlaisista objekteista erilaisia värivariaatioita samalla tekstuurilla. Voidaan siis värjätä koko objekti tai jokin sen osa eriväriseksi ja luoda näin variaatioita samasta esineestä. Erinomaisina esimerkkeinä tällaisista esineistä toimivat Trials Frontierin tynnyrit. Kaikki nämä ruostuneet tynnyrit kuvassa 32 jakavat saman tekstuurin, mutta niiden rungon verteksit on värjätty eri sävyillä. Verteksien värit tulevat puhtaimmin läpi silloin, kun ne eivät sekoitu tekstuurin väreihin lainkaan, eli täysin valkoiseen väriin. Harmaasävyisen tekstuurin kanssa siis väri vain tummuu tai vaalenee, mutta muiden värien kanssa ne sekoittuvat. Tekstuuri voi olla totta kai minkä värinen tahansa, mutta silloin pitää vain ottaa huomioon värin sekoittuminen verteksin väriin. Tätä sekoittumista voi kuitenkin käyttää myös hyödyksi, mikäli sitä halutaan käyttää tarkoituksenmukaisesti tiettyjen sävyjen saavuttamiseen tekstuurin kanssa.

Verteksivärjäystä tai varjostamista voisi käyttää pelin jokaisessa objektissa, mutta se vaatii tietysti lisää työtä graafikolta. Tämän lisäksi monet pelimoottorit osaavat generoida tällaiset varjostukset automaattisesti, joten todellinen hyöty omassa tilanteessani painottui värivariaatioiden luomiseen samanlaisista objekteista.

Kuva 32. Tynnyreiden käyttämä tekstuuri.



Kuva 33. Eri värivariaatiot tynnyreistä verteksivärjättyinä.

Tässä tynnyri-esimerkissä tekstuuri on värjätty hieman ruskeaksi, joka yhdistettynä tynnyrin pohjaväriin vertekseissä, luo halutun lopputuloksen ruosteisesta tynnyristä. Tämä tekstuuri sisältää myös osia muista tynnyreistä jotka käyttävät samaa tekstuuria. Verteksivärjäyksellä on suorituskyvyn kannalta täysin marginaalinen vaikutus ruudunpäivitykseen erityisesti Genetek-pelimoottorin kanssa. Tynnyrien tapauksessa pystymme siis jälleen optimoimaan myös Drawcall-vaiheita, kun kaikki tynnyrit käyttävät samaa tekstuuria ja materiaalia.

5 Yhteenveto ja pohdinta

Olen käynyt tässä työssä läpi keskeisimmät tekniikat, joita itse olen soveltanut työssäni ja Trial Frontier-pelin yhteydessä. Vaikka tässä tapauksessa on kysymys mobiilipelistä ja tietynlaisesta pelimoottorista, eivät tekniikat silti rajoitu pelkästään mobiilipeleihin tai lowpoly-mallinnukseen. Samankaltaisia optimoinnin keinoja graafikot voivat soveltaa myös uuden sukupolven konsoli- tai pc-peleissä. Olisikin ollut mielenkiintoista tutkia näitä tekniikoita erilaisen pelimoottorin kanssa ja malleilla, jotka käyttävät esimerkiksi normal map-tekstuureita. Vaikka peliteknologia ja grafiikka kehittyy hurjaa vauhtia, olen Trials Frontierin kanssa kokenut oppineeni jotain arvokasta ja sellaista, mihin en viimeisimpien konsolipelien kanssa olisi välttämättä törmännyt. Tästä johtuen olenkin erittäin tyytyväinen siihen, että sijoituin juuri mobiilipeliprojektiin. Vastaavanlaista opetusta olisi voinut olla tarjolla jo opiskelun aikana, ja siksi uskonkin tämän opinnäytetyön olevan hyödyllinen monille pelialasta kiinnostuneille graafikoille.

Trials Frontier on ollut projektina erittäin opettavainen, ja erityisesti tekstuurien maalaamisessa koen kehittyneeni huomattavasti. Kaikki Trials Frontierin tekstuurit ovat käsityötä, eikä valokuvapohjaisia tekstuureita käytetä lainkaan. Tämä pakottaa artistit maalamaan tekstuurit alusta alkaen, tietenkin mukaillen pelin konsepteja. Kirjoitushetkellä peli on kuitenkin vielä kesken, ja työtä vielä riittää. Tämä peli kuitenkin tuntuu itselleni oivalliselta ponnahduslaudalta seuraavaan projektiin, oli se sitten mikä tahansa.

Lopuksi haluan esittää vielä muutaman kuvakaappauksen pelistä sellaisena, kun se tällä hetkellä kehitysvaihetta on. Kuvat ovat aikaisia kaappauksia, mutta suurin osa malleista luultavasti säilyy sellaisena kuin ne kuvissa esiintyvät. Olen yrittänyt valita sellaiset kuvat, joissa esiintyvät samat mallit joita olen opinnäytetyössäni esitellyt. Nämä kentät ja radat ovat kenttäsuunnittelijoiden luomuksia, jotka siis keskittyvät tekemään kenttiä ja ratoja peleihimme. Nämä kuvat osoittavat myös sen, kuinka mielikuvituksellisia rakennelmia tällaisilla malleilla on mahdollista luoda, kun niiden muodot eivät rajoita niitä yhteen ainoaan käyttötarkoitukseen. Tarkoitan tällä esimerkiksi kiviä ja lankkuja, joista tarjoamme kenttäsuunnittelijoille erilaisia variaatioita leveyden ja pituuden suhteen. Emme siis mallinna heille valmiita siltoja tai muita rakennelmia, vaan annamme heille palaset niiden rakentamiseen. Tällöin emme rajoita heidän työtään tai mielikuvitustaan, vaan he voivat itse soveltaa näitä malleja haluamallaan tavalla. Mainittakoon vielä pelistä mekaniikan kannalta se, että peli soveltaa fysiikkaa samalla ta-

valla dynaamisesti kuin Trials Evolution. Tämä mahdollistaa sen, että kentissä voidaan käyttää monenlaisia fysiikkaan liittyviä esteitä ja sovelluksia. Trials Frontier julkistettiin E3-messuilla 2013 ja suunniteltu julkaisupäivä on vuoden 2014 puolella.

TRIALS FRONTIER™

Lähteet

Eat 3D 2013. Unreal Development Kit 3 – iOS Mobile Game Production. [Verkkodokumentti] <http://eat3d.com/udk_mobile> (luettu 16.4.2013).

Valient, Michal 2007. Deferred Rendering in Killzone 2. Guerilla. [Verkkodokumentti] < http://www.guerrilla-games.com/publications/dr_kz2_rsx_dev07.pdf> (luettu 24.4.2013).

Gat, Noam 2010. Deferred Rendering Demystified. GameDev.net. [Verkkodokumentti] <http://www.gamedev.net/page/resources/_/technical/graphics-programming-and-theory/deferred-rendering-demystified-r2746> (luettu 17.4.2013).

Hargreaves, Shawn. Deferred Shading. GameDevelopers Conference. [Verkkodokumentti] < http://www.shawnhargreaves.com/DeferredShading.pdf> (luettu 16.4.2013).

Null_ptr 2009. Deferred Shading, AA, alpha-blending. [Verkkodokumentti] < http://null-ptr.blogspot.fi/2009/01/deferred-shading-aa-alpha-blending.html> (luettu 16.4.2013).

GameDev.net 1999-2013. Deferred Rendering Alpha Blending/Transparency Issue. [Verkkodokumentti] < http://www.gamedev.net/topic/619554-deferred-rendering-alpha-blendingtransparency-issue/> (luettu 16.4.2013).

Everitt, Cass. Interactive Order-Independent Transparency. [Verkkodokumentti] < http://gamedevs.org/uploads/interactive-order-independent-transparency.pdf> (luettu 16.4.2013).

Polycount wiki 2011. Vertex Color. [Verkkodokumentti] < http://wiki.polycount.com/VertexColor#Usage> (luettu).

Polycount wiki 2013. Ambient-Occlusion Vertex Color. [Verkkodokumentti] <http://wiki.polycount.com/AmbientOcclusionVertexColor?action=show&redirect=Ambient-Occlusion+Vertex+Color > (luettu).

Kuvalähteet

Kuva 1. Valient, Michal 2007.
http://www.guerrilla-games.com/publications/dr_kz2_rsx_dev07.pdf

Kuva 3. Ivanov, Ivan-Assev 2006.
http://www.gamasutra.com/view/feature/130940/practical_texture_atlases.php?page=1

Kuva 30. Oberson, Pior 2013.
http://wiki.polycount.com/AmbientOcclusionVertexColor?action=show&redirect=Ambient-Occlusion+Vertex+Color