Opinnäytetyö (AMK)

Tietotekniikka

Click here to enter text.

2017

Sampo Pihlaja

PHYSICALLY BASED RENDERING

– Case Godsbane ja Hear No Evil

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tietotekniikka

2017 | 32

Yliopettaja Mika Luimula, dosentti

Sampo Pihlaja

PHYSICALLY BASED RENDERING

- Case Godsbane ja Hear No Evil

Tämän työn tarkoitus on esitellä Physically Based Rendering teknologiaa sekä esittää kuinka sitä voi soveltaa omiin peliprojekteihinsa. PBR-tekniikan ollessa niin uutta siitä löytyy hyvin vähän julkaistua materiaalia. Olen saanut poimittua kuitenkin olennaisimmat asiat kattavaksi selostukseksi PBR:n vaiheista. PBR soveltuu peligrafiikassa hyvin realistiseen tyyliin ja siihen se on tarkoitettu. Useat artistit ovat kuitenkin osoittaneet, että PBR:n sisäistettyään se venyy myös paljon muuhuin kuin vain realismiin. Projektimme olivat realismia tavoittelevia, ja PBR:n opetteluun se oli juuri oikea lähestymistapa.

ASIASANAT:

PBR, renderöinti, teksturointi, shaderit, 3d-malli

BACHELOR´S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information Technology

2017 | 47

Principal Lecturer Mika Luimula, Adj. Prof.

Sampo Pihlaja

PHYSICALLY BASED RENDERING

- Case Godsbane and Hear No Evil

This work explains the theory of Physically Based Rendering (PBR), and its use in the Rockodile Games projects, Godsbane and Hear No Evil. The work generally describes the basics of PBR and focuses on PBR uses and applicability. The work presents a more detailed description of one PBR workflows, namely the Metalness -workflow. PBR is a rather new technology in the gaming industry and this work sheds some light on how a small team of game developers can use it successfully.

Click here to enter text.

KEYWORDS:

PBR, rendering, texturing, shader, 3d-model

SISÄLTÖ

KUVAT

KÄYTETYT LYHENTEET TAI SANASTO

CPU	Tietokonelaitteiston keskusprosessori.
FPS	”Frames per second”. Kuvia sekunnissa.
GPU	Näytönohjain.
Map	Tekstuuri eli 2d-kuva
PBR	Physically Based Rendering
Shader	Näytönohjaimella suoritettava ohjelma.
Tekstuuri	Kuvatiedosto josta näytönohjain lukee dataa.
Unreal Engine 4	Kaupallinen pelimoottori joka on ilmainen käyttää
Workflow	Rajattu työskentelytapa jossa tietty asioiden tekojärjestys.

1 JOHDANTO

Tässä opinnäytetyössä käsittelen PBR-renderöintiä Godsbane sekä Hear No Evil –peliprojektien yhteydessä sekä teoriaa yleisesti. Tutkin PBR-renderöintiä eri näkökulmista ja kahta erilaista PBR:n työskentelytapaa, jotka vaikuttavat työnkulkuun vain hieman toisistaan poiketen.

PBR-renderöinti on suhteellisen uusi asia peliteknologiassa. Sen ensimmäisen esiintymisen tarkka ajankohta ei ole yleisessä tiedossa, mutta usealla foorumilla on viittausta vuoteen 2013. Vaikka PBR on uutta teknologiaa, sen on omaksunut jo usea AAA-kokoluokan pelimoottori. Sen näyttävyys suhteessa tehonkulutukseen on erittäin optimoitua ja kilpailukykyistä.

PBR-tekniikka pohjautuu oikean maailman fysiikoihin ja valon käyttäytymiseen. Sen kehittämistä varten on tutkittu paljon oikean maailman ilmiöitä valonsäteiden heijastumisen, absorboitumisen ja hajonnan osalta. PBR-renderöinnin tarkoituksena on mahdollisimman realistinen jälki, mutta sitä sovellettaessa mahdollisuudet ulottuvat tyyliteltyynkin grafiikkaan.

PBR on vaikeasti määriteltävä asia. Sitä ei voi määrittää tarkasti, koska siihen liittyy niin monta osa-aluetta 3d-grafiikan ja pelimoottorin osalta. PBR on pelimoottoriin kirjoitettuja algoritmejä, joille syötetään arvoja 2d-kuvista. PBR on 3d-malleja, tekstuureita sekä materiaaleja. PBR on monivaiheinen prosessi, jonka lopputuloksena on näytönohjaimen renderöimä kuva tietokoneen näytöllä.

PBS (Physically Based Shading) ja PBL (Physically Based Lightning) ovat toisia nimittyksiä PBR:lle. PBR on näistä kolmesta kuitenkin yleisimmin käytetty, suorastaan vakiintunut termi.

2 PBR YLEISESTI

2.1 PBR käsitteenä

PBR eli Physically Based Rendering on vaikeasti rajattava uusi renderöintitapa, joka jäljittelee algoritmeiltään mahdollisimman hyvin oikean maailman fysiikoita. PBR:n tarkoitus on aikaansaada mahdollisimman realistisia lopputuloksia optimoiden näytönohjaimen suorituskykyä. Olennaisimpana asiana PBR:n toimivuudessa on oikeasta maailmasta omaksuttu sääntö valon kuljettaman energian säästymisestä: Mistään pinnasta ei voi heijastua enempää valoa, kuin mitä siihen on tullut valonlähteestä. PBR käyttää hyväkseen oikeasta maailmasta mitattuja arvoja, valon heijastumisen, läpäisyn sekä dispersion osalta, jotta saavutettaisiin mahdollisimman realistinen lopputulos. (Wilson, J. 2015)

PBR-teknologiaa varten on tutkittu paljon valon käyttäytymistä ulko- sekä sisätiloissa. PBR:ssä valon käyttäytymistä ohjaillaan 2d-kuvilla eli tekstuureilla. Shaderit käyttävät tekstuurien pikseleiden arvoja parametreina renderöidyn kuvan aikaansaamiseksi. Shaderit lukevat kuvista niiden RGB-arvon ja sen mukaan muokkaavat 3d-mallin pintaa halutun materiaalin näköiseksi. Arvot luetaan välillä 0,0 - 1,0, vaikka kuvan RGB-arvo ulottuu välille 0.0-255.0. RGB-arvo sijoitetaan välille 0,0 – 1,0 vasta pelimoottorissa. 2d-tekstuurit tehdään erillisten ohjelmien avustuksella, lähes järjestään useampaa kuin yhtä ohjelmaa on käytettävä laadun ja työmäärän optimoimiseksi. (Wilson, J. 2015)

PBR-teknologia on hyvin uutta, ja siksi sen on omaksunut täysin vasta kourallinen pelistudiota. AAA-peleissä se on kuitenkin muuttumassa standardiksi, sen tehoja säästävän luonteen sekä näyttävyyttä rajoittamattomien ominaisuuksiensa ansiosta. Tietokoneisiin verrattuna tehottomat konsolit ja niiden yleistyminen on yksi syy PBR:n yleistymiseen.

PBR:stä mullistavan tekee sen tuoma mahdollisuus käyttää samaa 3d-mallia ja tekstuureita jokaisessa eri valaistusolosuhteessa. Tekstuurien antama data ohjailee vain sitä, miten valo käyttäytyy osuessaan objektin pintaan. Valonlähde voi siis olla eri vahvuuksia ja eri värejä, ja kukin vaihtoehto vaikuttaa objekteihin omalla tavallaan. Ennestään vastaavanlainen vapaus peleissä ei ollut itsestäänselvyys.

2.2 PBR työskentelytavat

Yleisimmät PBR-työskentelytavat (engl. workflow) ovat nimeltään Metalness ja Specular. Nämä työskentelytavat eroavat lähinnä käyttämiensä tekstuurien osalta. Työskentelytavoissa on yleisesti rajatut työskentelyolosuhteet.

Specular-työskentelytapa käyttää tekstuureja määrittämään esineen, 3d-objektin: karheutta, muotoa, kiiltävyyttä sekä väriä. Käytössä on myös Frensel-arvo, jolla määritetään esineen reuna-alueiden heijastavuutta. Specular-työskentelytavan etuna on vapaammin sovellettavat tekstuurit.

Metalness-työskentelytapa käyttää tekstuureja määrittämään esineen karheutta, muotoa sekä sitä, että onko esine metallia vai ei. Esineiden pinnat jaetaan metallisuudessa kahteen eri luokkaan: sähköä johtaviin ja eristäviin. Metallisuus pitää sisällään myös valon heijastuvuuden.

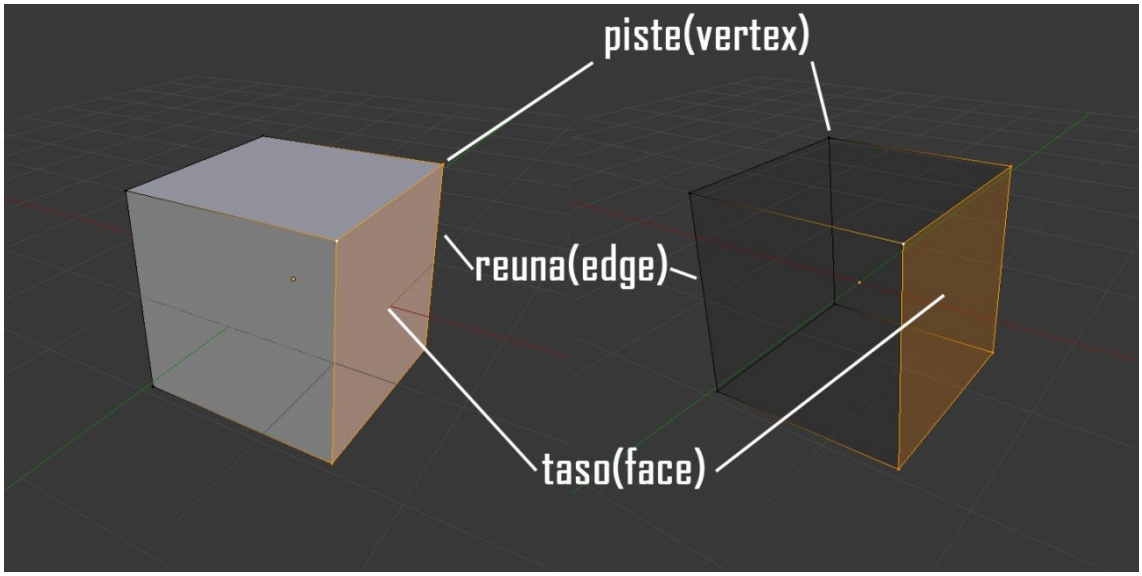
PBR:ää sovelletaan usealla tavalla riippuen kehitystiimistä ja pelimoottorin soveltuvuuksista. Eri työskentelytavoilla on kuitenkin mahdollista saada lähes identtisiä lopputuloksia, käytännössä vain tavat sekä työmäärä muuttuvat. Yleisimmät työskentelytavat ovat: Metalness, Specular & Microsurface, Albedo & Glossiness sekä Diffuse & Specular.Tässä työssä keskityn Metalnes-työskentelytapaan, jota Rockodile Games käyttää Creations of Conflict peliprojektissaan.

Eri työskentelytavat eroavat toisistaan lähinnä 2d tekstuurien ominaisuuksissa sekä niiden määrässä. Metalness-työskentelytapa on lukuisista tavoista vaivattomimpia ja yleisimmin omaksuttuja säilyttäen kuitenkin muokkautuvuuden ja mahdollisuuden luoviin kokeiluihin.

3 RENDERÖINTI

3.1 3d-malli

Kuvassa 1 on esitetty 3d-mallin pinnalla sijaitsevat pisteet (engl. vertex), reunat (engl. edge) sekä tasot (engl. face). 3d-malli on karkeimmillaan tekstitiedosto, joka pitää sisällään 3d-mallin pisteiden paikat XYZ-koordinaatistossa, ja pisteiden suhteet viereisiin pisteisiin. 3d-mallin pisteet määrittävät 3d-mallin muodon. Esimerkiksi kuution muodostamiseen tarvitaan vähimmillään kahdeksan pistettä. Halluttujen pisteiden välille muodostetaan reuna, ja pisteet yhdessä reunojen kanssa muodostavat tason.

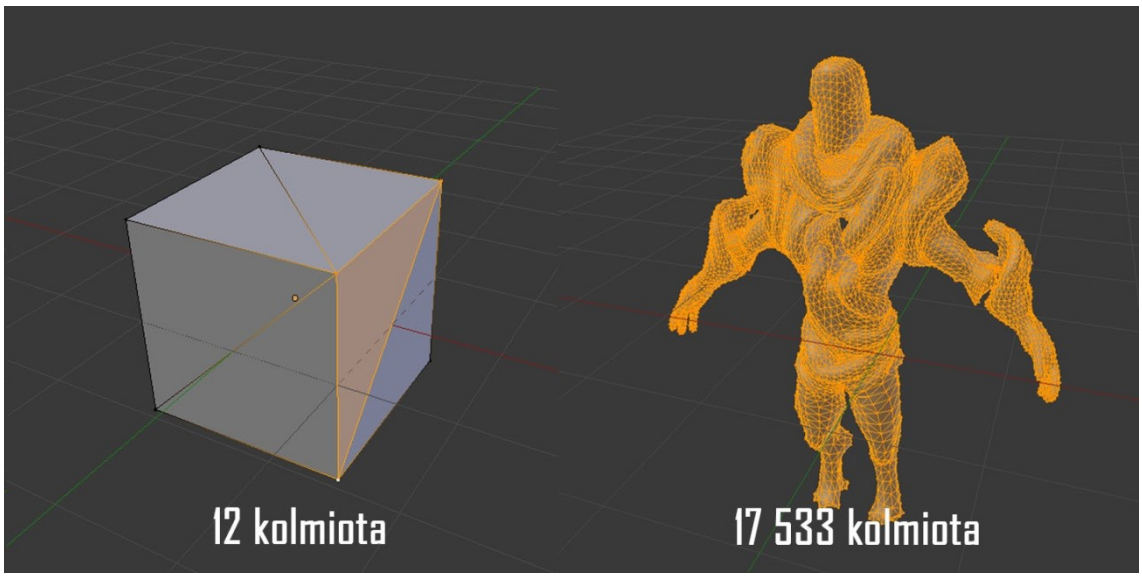


Kuva 1. 3d-malli kuutiosta tavallisena sekä rautalanka mallina.

Useimmissa peleissä 3d-mallit eivät muodostu neliö-tasoista kuten ylläolevassa kuvassa vaan kolmekulmaisista polygoneista. Neliöt, viisikulmiot ja sitä monimutkaisemmat muodot jaetaan kaikki polygoneiksi, joko 3d-mallinnusohjelmassa tai pelimoottorin toimesta. Polygoneiden määrä vaikuttaa 3d-mallin tehonkulutukseen. Tämän hetkisessä AAA-tason pelissä, jonka tehonkulutus on yli keskitason, näytönohjain joutuu työskentelemään useamman miljoonan polygonin kanssa, laskemaan niiden koordinaatteja

näytöllä, sekä piirtämään niiden muodostamat 3d-mallit loppuliseen renderöityyn kuvaan.

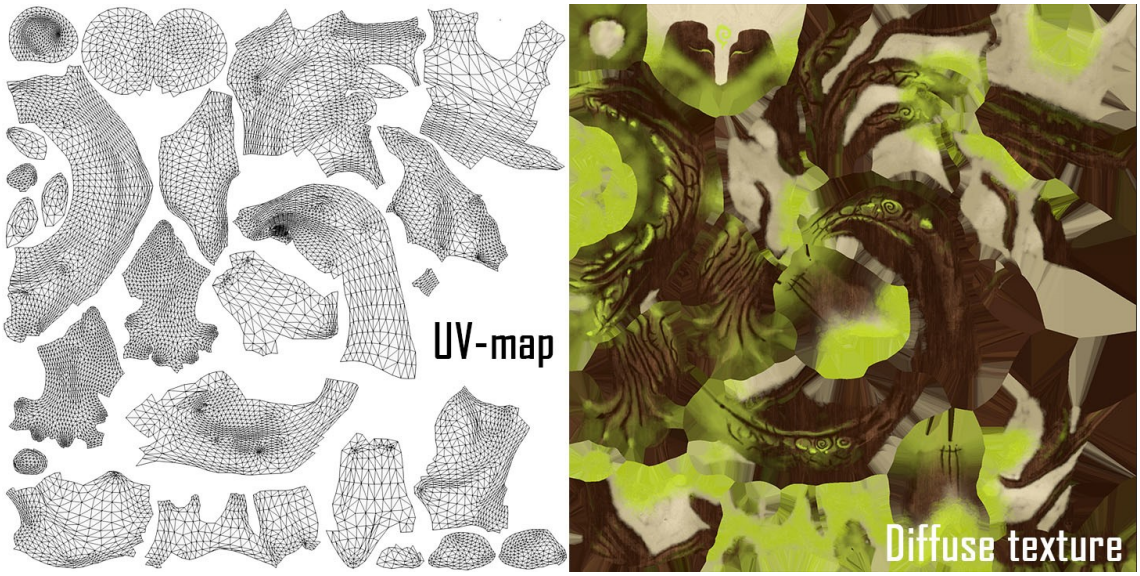
3d-mallien tehonvientiä mitataan ”polycountilla” eli polygonien, tässä tapauksessa polygoneiden, määrällä. PBR menetelmän yhtenä tarkoituksena on mahdollisuus saada tekstuurien avulla ja vähäisemmällä määrällä polygoneja, kilpailukykyistä ja yksityiskohtaista jälkeä. Esineestä tai hahmosta riippuen polycountin tarve vaihtelee. Alla olevasta kuvasta käy hyvin ilmi miten yksinkertaisen kuution saa paljon vähemmällä määrällä polygoneja kuin esimerkiksi pelihahmon. (Wikipedia – rendering. 2016)



Kuva 2. 3d-malli kuutiosta ja Godsbane pelin hahmosta sekä niiden sisältämien polygoneiden määrä.

3.2 3d-malli ja UV-koordinaatisto

UV-kordinaatiston avulla 3d-mallin pinnalle voidaan levittää 2d-kuva haluttuille alueille. Muokkausvaiheessa 3d-malliin merkitään saumakohtia, joista 3d-malli ”aukeaa”. Saumakohtien avulla 3d-malli voidaan projisoida 2d-tasolle. Tätä tapahtumaa voi visualisoida origamin purkamisena takaisin neliöksi. 3d-mallin UV-koordinaatiston perusteella 3d-mallin pinnalle voidaan levittää 2d-kuvia, tekstuureita. Ilman 3d-mallin UV-mappia sen pinnalle asetetut kuvat piirtyvät nurinkurin tai aivan vääriin paikkoihin, sillä renderöintiohjelma ei tiedä, mihin pikselit tulee 3d-mallin pinnalla piirtää. (UV-mapping)



Kuva 3. Godsbane pelin hahmon UV-map sekä Diffuse tekstuuri. UV-map kertoo 3d-mallin pisteiden paikat.

3.3 3d-mallin Hi-poly ja Lo-poly versiot

Hi-poly malliksi kutsutaan korkean polygonimäärän omaavaa 3d-mallia. Hi-polyn polygonimäärälle ei ole tiettyä alarajaa, mutta sillä tarkoitetaan usein mallia, joka on liian raskas peligrafiikkaan. Hi-poly malli on usein ensimmäinen vaihe objektin tekemisessä. Hi-poly mallin pohjalta tehdään paljon vähemmän polygoneja sisältävä Lo-poly malli.

Lo-poly mallilla tarkoitetaan todella vähäisen polygoni määrän omaavaa 3d-mallia. Polygonimäärää lasketaan suoritustehon takia. Näytönohjain joutuu käsittelemään jokaista näytöllä näkyvää polygonia, jotka muodostavat näytöllä olevat esineet, rakennukset, eläimet, hahmot ja pinnanmuodot.

3.4 Renderöinti

Renderöinti on peleissä tapahtuma siitä, kun näytönohjain luo näytölle kuvan perustuen ennaltamäärättyihin parametreihin. Renderöinti voi olla lukuisia asioita kuten esimerkiksi ääniraidan koostamista tai animaatioiden kohtauksien kuvakohtaista luomista. Renderöinti tapahtuu useimmiten näytönohjaimen tai koneen prosessorin avulla. Renderöinti, josta tässä työssä kerron, keskittyy pelikuvan renderöintiin. Renderöinnissä näytönohjain käyttää dataa 3d-mallin geometriasta, polygoniista, katselukulmasta, teks-

tuureista sekä valaistuksesta. Tuloksena saatu 2d-kuva, joka piirtyy näytölle, on esitys sen hetkisestä tilanteessa pelissä. Jotta pelikuva ei töksähtelisi, se joudutaan päivittämään näytölle useita kymmeniä kertoja sekunnissa. Sulavan kuvan aikaansaamiseksi renderöinti täytyy suorittaa vähintään 24 kertaa sekunnissa, jotta silmä ei erottaisi yksittäisi kuvia. Yhden kuvan renderöintiin saa kulua aikaa 0,042 s ja siksi sen pitää olla näytönohjaimelle verrattaen kevyt suoritus. Siinä, missä esimerkiksi täysin tietokonegrafiikkaa hyödyntäviä 30 s:n mainoksia saatetaan renderöidä tehokkailla koneilla jopa useita kymmeniä tunteja, Godsbane–pelin on tarkoitus pyöriä 60 fps, eli 60 kuvaa sekunnissa. (Wikipedia – rendering. 2016)

Pelkästään pelikuvan renderöintiin on käytössä useita kymmeniä erilaisia algoritmejä riippuen pelimoottorista. Renderöintitekniikka riippuu useasta muuttujasta, mutta suurimpana vaikuttajana on käyttötarkoitus. Peligrafiikan renderöinnin pitää olla nopeaa ja mahdollisimman yksinkertaista, kun taas mainoskuvan renderöinti saa kestää useita tunteja. Renderöintitekniikat vaihtelevat raskaasta aidon valon käyttäytymistä simuloivasta ray tracingistä kevyempään rasterirenderiin. Pelikuvan renderöinti on usein jälkimmäistä hyödyntävä ratkaisu. PBR-renderöinti on tekniikka, joka saa vaikutteita ray-tracingin valon käyttäytymistä laskevista algoritmeistä, kuitenkin hyödyntäen siinä tekstuureita eikä prosessoiden yksittäisten valonsäteiden käyttäytymistä. (Wikipedia – rendering. 2016)

3.5 PBR-renderöinti

3d-mallien tarkkuus, ”polycount”, on pidettävä vähäisenä, ettei renderöinti ole liian raskasta. Polycountin rajoittamisesta seuraa kuitenkin paljon huonompiresoluutioisia 3d-malleja, jotka ilman apukeinoja eivät vastaa nykypeligrafiikan tasoa. PBR-teknologia osaa muokata näitä vähäisen polycountin omaavia 3d-malleja tekstuurien avulla, tuottaen laadukkaamman näköisen renderöintilopputuloksen kuin tavanomaiset tekniikat. Tekstuureilla voidaan muokata jopa 3d-mallin muodostavien tasojen muotoa, koskematta 3d-mallin dataan. Aiemmin näin syvälle ulottuva 3d-mallin muokkaaminen tekstuurien avulla ei ole ollut mahdollista.

Kuva 4. Godsbane pelin Nature Heeros –hahmo. 3d-malli ja teksturoitu versio joka koostuu neljästä tekstuurista.

3.6 Shaderit

Shaderit ovat algoritmeja, joiden avulla näytönohjain tietää, miten muokata sen hetkistä kuvaa näytöllä. Shaderit ovat yksinkertaistettuna piirtokomentoja, jotka muokkavaat erilaisten parametrien avulla näytölle piirtyvän kuvan pikseleiden väriarvoja halutunlaisiksi. PBR tekniikka käyttää hyväkseen lukuisia shadereitä, ja riippuen pelimoottorista ja työskentelytavasta näitä shadereitä on joko enemmän tai vähemmän ja niiden tehovaatimukset muuttuvat. Esimerkkinä shadereiden lukumäärästä Metalness PBR-työskentelytapa käyttää hyväkseen kahtakymmentä erilaista shaderiä.

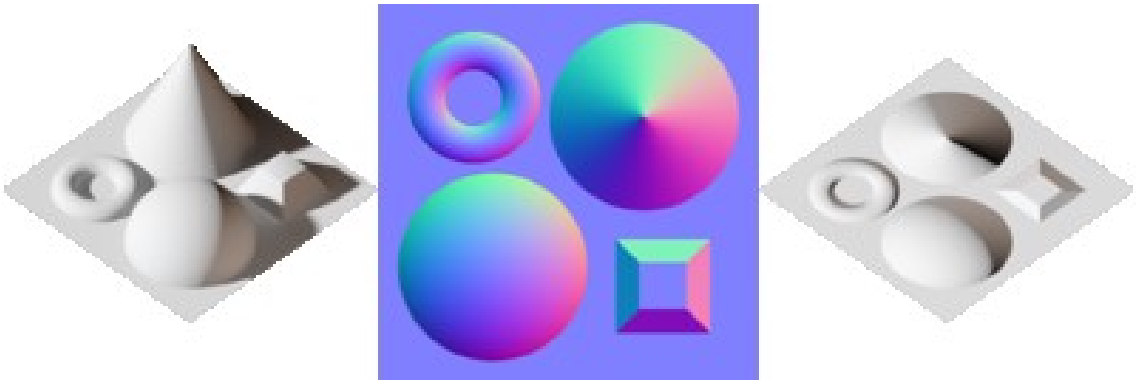
4 TEKSTUROINTI

4.1 Tekstuurit

PBR noudattaa oikean maailman valon käyttäytymisen fysiikkaa. Renderöintiä, eli näytönohjaimessa tapahtuvaa laskentaa jonka lopputuloksena kuva piirtyy näytölle, ohjaillaan erilaisilla 2d kuvilla, tekstuureilla. PBR teknologiassa on tekstuureja yleisessä käytössä toistaiseksi seuraavat: Normal Map, Diffuse Map, Ambient Occlusion Map, Cavity Map, Height Map, Position Map, Roughness Map, Metalness Map, Specular Map, Displace Map ja Curvature Map. Osan tarkoituksena on tukea työvaiheita ja helpottaa lopputulokseen pääsemistä, kun taas osa syötetään suoraan näytönohjaimelle ja ne ohjaavat renderöinnin lopputulosta shadereiden algoritmien mukaan. Tässä työssä tarkempaan tarkasteluun on valittu neljä tekstuuria, joita Rockodile Games –yritys käyttää nykyisessä projektissaan. (Russel, J. 2015)

4.2 Normal Map

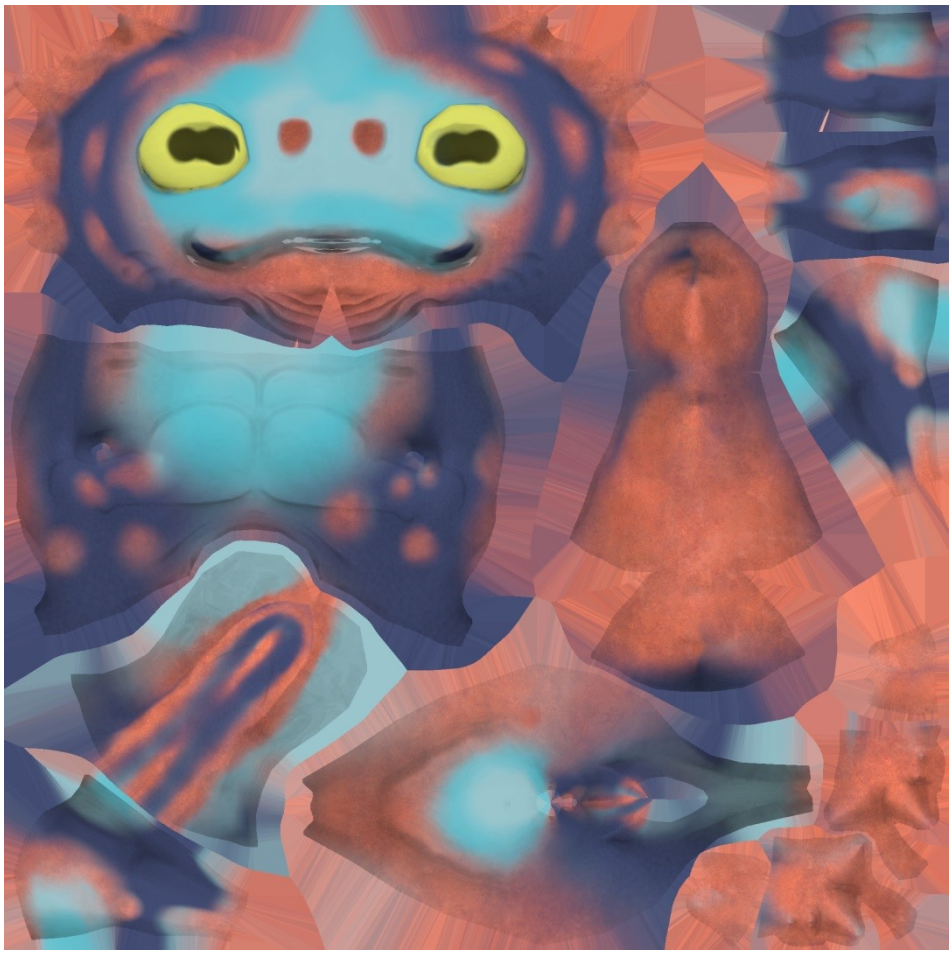
Normal Map on RGB värikanavia käyttävä tekstuuri, jossa värien tarkoitus on ohjailla 3d-mallin yksittäisten pikseleiden paikkaa näytöllä. Jokaisella värillä on oma akselinsa 3d-koordinaatistossa. Värin itensiteetin perusteella näytönohjain osaa muokata kyseisen pikselin paikkaa 3d-mallin pinnalla, ja täten muokaa varjostuksia sekä heijastuksia siten, että 3d-malli saa monipuolisempia muotoja piirtyessään näytölle. Normal map ei varsinaisesti muokkaa polygoneiden muotoja vaan pelkästään niihin osuvia heijastuksia. (Disqus. 2015)



Kuva 5. Oikeanpuoleisimmassa kuvassa on täysin tasapintainen 3d malli, jonka varjostuksia ja heijastuksia muokataan Normal Mapilla (keskimmäinen kuva).

4.3 Diffuse Map

Diffuse Map on 2d-tekstuuri joka kertoo 3d-mallin pinnan värityksen. Diffuse Map on väritykseltää täysin heijastukseton, eli värit ovat siinä luonnollisissa valoisuus- sekä kylläisyystasoissa. Diffuse Map ei yleensä sisällä heijastuksia tai varjostuksia, vaan näitä ohjaillaan muilla 2d tekstuureilla. Godsbane pelissä meidän tuli kuitenkin yhdistää Ambient Occlusion map Diffuse tekstuuriin, sillä Ambient Occlusion mappia, eli ahtaiden tilojen varjostuksia ei käsitelty pelimoottorissa erikseen. Diffuse tekstuuri on metallisissa pinnoissa lähes järjestään musta tai todella tumma. PBR:ssä metallisten pintojen väri muodostuu heijastuksista eikä metallin omasta väristä. Tämä on johdettu oikean maailman metallin ja valon käyttäytymisestä. (Russel, J. 2015)

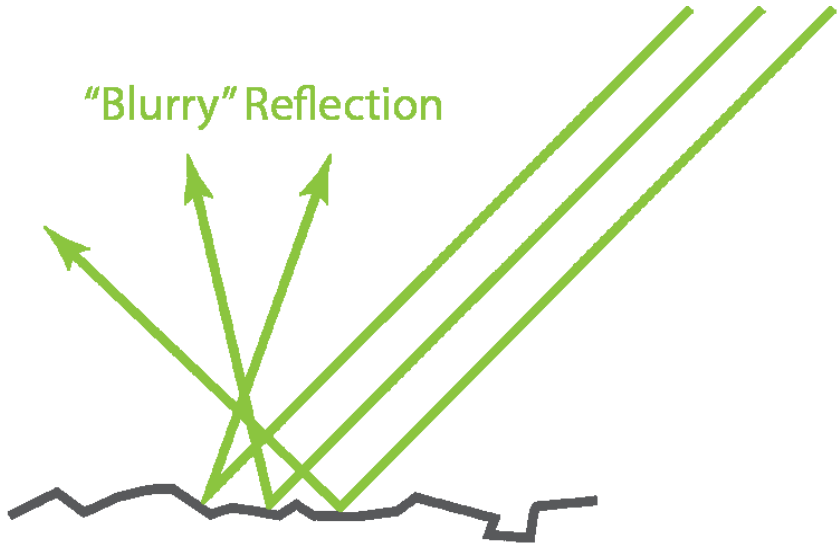


Kuva 6. Creations of Conflict pelin Amfibos hahmon Diffuse Map.

4.4 Roughness Map

Roughness Map on 2d-tekstuuri joka ohjailee 3d-mallin pinnan karheutta ja valon heijastumista mallin pinnalta. Oikeassa maailmassa pinnanmuodot saattavat olla lähes loputtoman komplekseja. Laskentatehon nimissä pinnankarheutta määritellään PBR:ssä arvoilla nollasta yhteen. Karhea pinta hajoittaa valon heijastumista, mutta siinä ei kuitenkaan kosketa varsinaisen 3d mallin pinnan muotoihin, sillä tämänhetkinen tarkkuus ja laskentateho eivät riitä. Roughness Map on mustavalkoinen kuva, jossa täysin musta pikseli, arvoltaan 0, viestii täysin sileästä pinnasta kun taas täysin valkoinen pikseli, arvoltaan 1, tarkoittaa karheaa pintaa. Valon energian säästymisen lakia

noudattaen pinnasta heijastuu aina sama määrä valoa, mutta karhealla pinnalla valon heijastukset ovat sumeampia ja asettuvat laajemmalle alueelle kun taas kiiltävällä pinnalla valon heijastukset ovat terävämpiä ja siksi kirkkaampia. (Wilson, J. 2015)



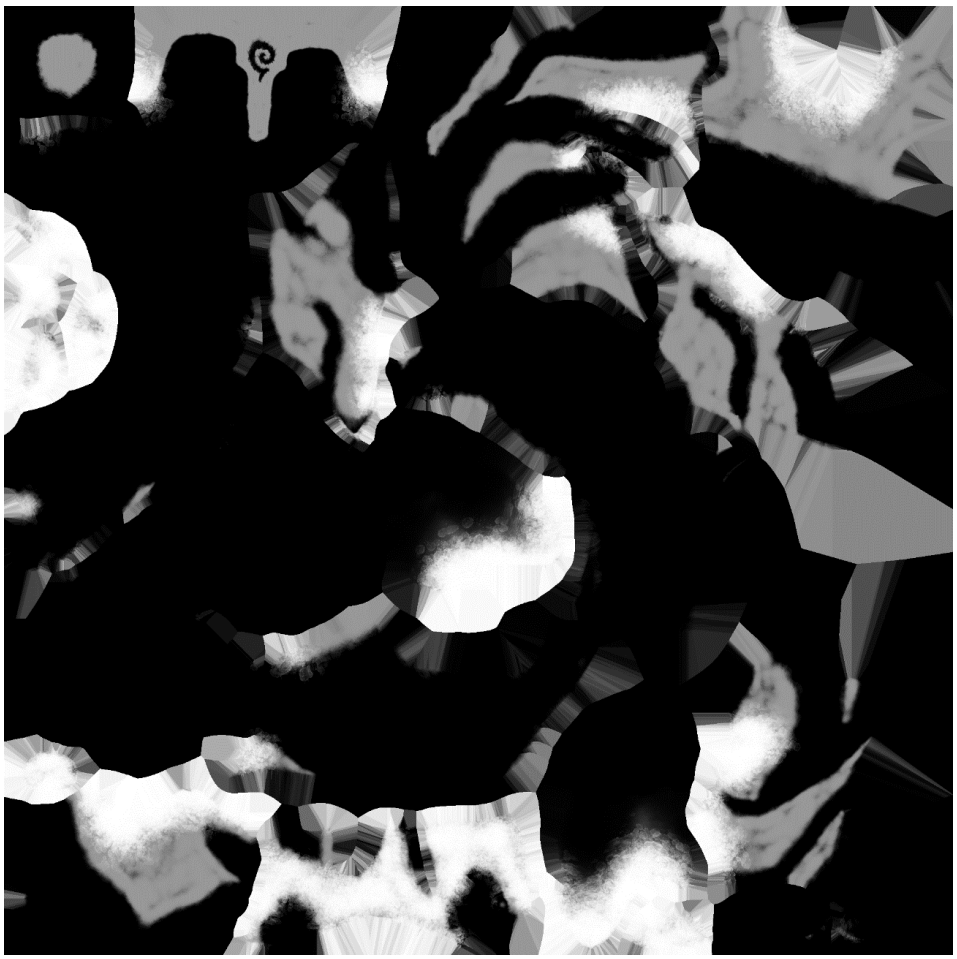
Kuva 7. Kuvaus siitä miten valo käyttäytyy osuessaan karhealle pinnalle, ja miksi karhean pinnan heijastukset ovat sumeampia kuin täysin sileän pinnan.

Kuva 8. 3d-mallin pinnan karheuteen vaikutetaan Roughness Mapilla.

4.5 Metalness Map

Metalness Map on 2d-tekstuuri joka kertoo 3d-mallin valoa heijastavista ominaisuuksista. Metalness Map on mustavalkoinen kuva, jossa musta pikseli tarkoittaa orgaanista ainetta ja valkoinen pikseli metallista. Harmaita värejä käytetään Metalness Mapissa hyvin niukasti. Orgaaninen aine eroaa metallisesta sen pinnan heijastuksien osalta, ja sitä on PBR:ssä hieman yksinkertaistettu. Orgaanisen aineen heijastuksia käsitellään

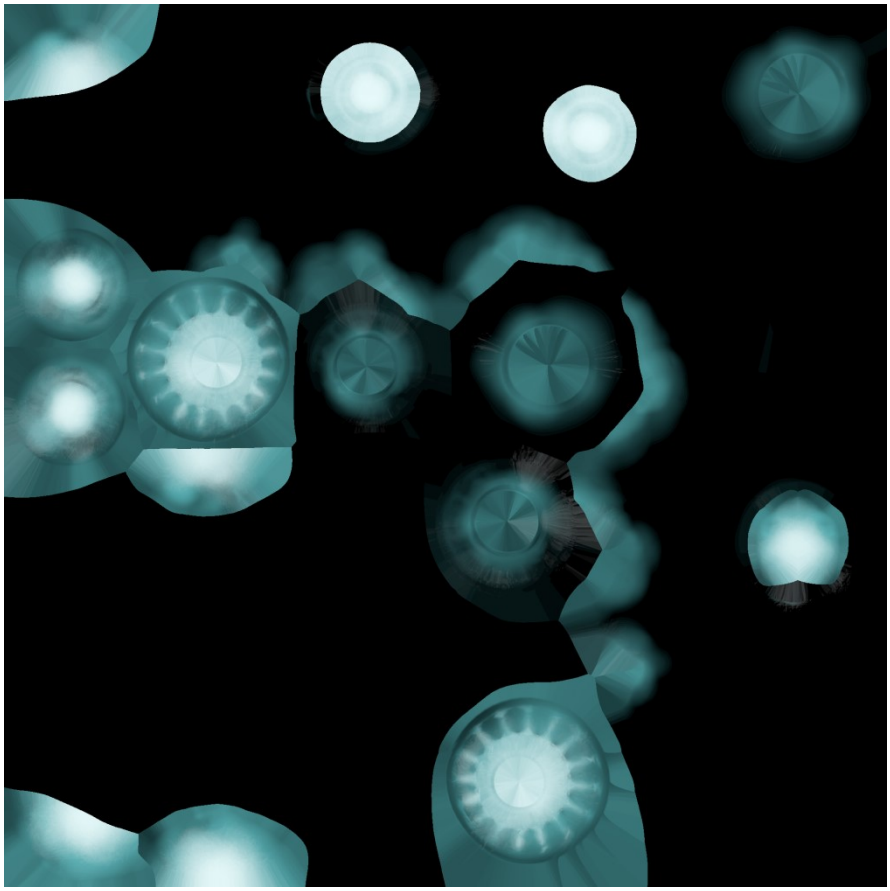
aina valkoisina kun taas metallinen aine voi heijastaa muitakin värejä. Orgaanisella aineen pinnalla on usein jonkinlainen väri, kun taas metallisia aineita käsitellään PBR:ssä aina mustana; niihin tulee värejä pelkästään heijastuksista. Godsbane pelissä suurin osa 3d-malleista kuvastavat orgaanisia aineita, joiden Metalness Map on täysin musta. Tästä syystä useimmissa 3d-malleissa ei käytetä ollenkaan tekstuuria määrittämään pinnan metallisuutta, vaan näytönohjaimelle annetaan suoraan arvo 0, joka tarkoittaa että malli on kokonaisuudessaan orgaanista materiaa. (Wilson, J. 2015)



Kuva 9. Metalness Map Godsbane pelin hahmosta. Valkoinen kuvaa metallia ja musta orgaanista ainetta.

4.6 Emissive Map

Emissive Map on 2d-tekstuuri, joka kertoo 3d-mallin valoa hohkaavien kohtien värin sekä intensiteetin. Emissive Map on värikuva jossa on värjättynä vain ne kohdat joiden on tarkoitus hehkua valoa. Tätä valoa jota PBR simuloi Emissive Mapin perusteella ei kuitenkaan käsitellä pelimoottorissa oikeana valonlähteenä, vaan sen lopputulos saa aikaan hohtavia kohtia, jotka eivät heijasta valoa muihin esineisiin pelissä. Tämän avulla saadaan aikaan valaistuilta näyttäviä kohteita kuitenkaan viemättä aitoa valoa simuloivien algoritmien laskentatehoa. (Russel, J. 2015)



Kuva 10. Emissive Map Hear No Evil –peliprojektin päähahmosta. Musta on normaalia pintaa ja värikkäät osat kertova hehkun värin sekä intensiteetin

4.7 Fresnell

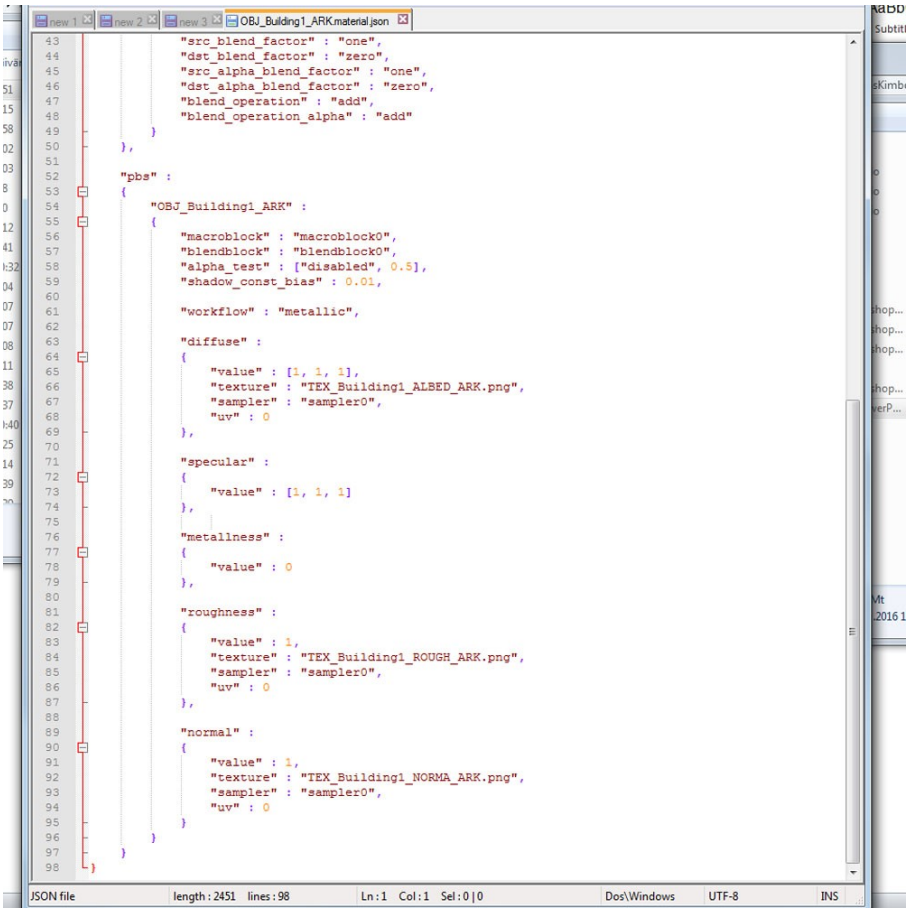
Fresnell ei ole oma tekstuurinsa vaan se on numeroarvo joka määritellään materiaalia tehtäessä. Fresnel arvo kertoo 3d-mallin reunoille levittäytyvien heijastuksien intensiteetin. Lähtökohtaisesta jokainen materiaali heijastaa reuna-alueilla valoa, kun objektia katsoo oikeasta kulmasta. Fresnel muokkaa näitä heijastuksia joko lähemmäs tai kauemmas objektin keskikohtaa. Frensel arvo määrittää objektin reunvaloa suorasta katselukulmasta, jota kutsutaan F0:ksi. (Disqus. 2015)

5 MATERIAALIT

5.1 Materiaalit

Materiaalit ovat tekstuurien käyttökohteen määrittäjiä. Materiaalit sisältävät myös tarpeellista dataa, jota tekstuurit eivät sisällä. Materiaaleissa määritetään arvot 3d-mallin pintaa muokkaavalle shaderille ja tekstuurit, joita tulee käyttää renderöintiin. Godsbane-peliprojektissa käyttämämme materiaalit editoitiin suoraan tekstitiedostoon, sillä omassa pelimoottorissamme ei ollut sen muokkaamiseen erillistä ohjelmaa. Pelimoottorista riippumattomat ohjelmat taas ovat riittämättömiä käytettävyydeltään sekä lopputulokseltaan.

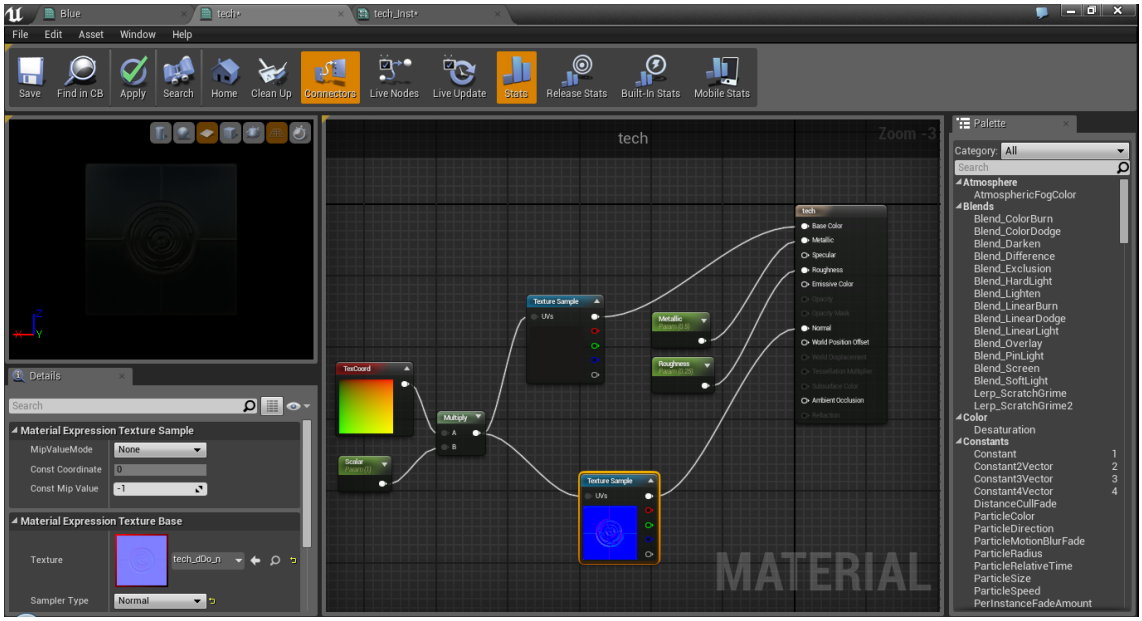
Materiaalissa määritellään käytettävien tekstuurien lisäksi UV-koordinaatien toistuvuus, tekstuurien skaalaus kohteen ollessa kauempana kamerasta sekä funktioiden valinnat.



Kuva 11. Notepad++ ohjelmassa avattuna Godsbane pelin talon materiaali.

5.2 Materiaalit Unreal Engine 4:ssä

Unreal Engine 4:ään sisäänrakennetut materiaali muokkaimet olivat kymmeniä kertoja laajemmat ja tehokkaamat kuin omassa pelimoottorissamme. Unreal Engine 4:n materiaalit mahdollistivat läpinäkyvien, valoa hehkuvien, satunnaislukuja hyödyntävien alati muuttuvien pintojen sekä normaalien PBR-mallien työstämisen. Unreal Engine 4:ssä materiaalit tehtiin erillisillä muokkausttyökaluilla, jossa erilaisia elementtejä yhdisteltiin visuaalisessa valikossa. (Epic Games Inc. 2016)



Kuva 12. Unreal Engine 4:n materiaalin muokkaus työkalu.

6 ROCKODILEN PELIPROJEKTIT PBR:LLÄ

Minun työtehtäväni Godsbane ja Hear No Evil -peliprojekteissa oli PBR-sisäistäminen sekä tekstuurien, 3d-mallien ja materiaalien työstäminen. Tapani tehdä näitä kolmea asiaa muuttuivat projektin kuluessa hieman, oppiessani lisää PBR:stä. Ohjelmat, joita käytin projektin aikana, vaihtuivat lähes täysin. Alussa työskentelin tutummilla ohjelmilla opiskellen samalla muutamaa uutta. Työskentely oli kuitenkin kankeaa ja soveltamaan joutui paljon, sillä ohjelmia ei ollut tarkoitettu PBR-teknologiaa varten. Oli siis kätytettävä standardeiksi nousseita ohjelmia, sillä ne olivat tarkoitettu tekemään juuri sitä, mitä Rockodile Games tarvitsi.

Godsbane on Rockodile Gamesilla työn alla ollut peliprojekti. Godsbane on Rockodile Gamesin ensimmäinen projekti, jossa hyödynnettiin PBR-renderöinnin tuomaa teknologiaa. 3d-peliin päädyttäessä Rockodile Gamesille oli selvää, että työskentelytavan olisi oltava nopeasti omaksuttava, mutta silti laadukas. PBR-renderöintiin tutustuttaessa kävi nopeasti selväksi, että se olisi juuri oikea vaihtoehto pienelle tiimille. PBR-teknologia oli projektin alkaessa kaikille jäsenistä täysin uusi kokemus, ja sen opettelu oli ensisijalla projektin alkaessa.

Aluksi PBR-renderöinnissä käytettäviä tekstuureita päädyttiin tekemään X-normal ohjelman sekä Photoshopilla. X-normal ohjelmaa käytettiin luomaan tiettyjä osia tekstuureista automaatiikan avulla. X-normalista saatiin tallennettua Normal Map jota pystyi käyttämään sellaisenaan pelissä. Sen lisäksi X-normal tuotti Ambient Occlusion, Curvature ja Height Map -tekstuurit, joita pystyin käyttämään Photoshopissa luodessani lopullisia tekstuureita.

Rockodile Games aloitti Godsbane -peliprojektin Specularity työskentelytavalla. Muutaman kuukauden jälkeen kävi kuitenkin selväksi, että Specularity-työskentelytavan antamat vapaudet olivatkin lähinnä haitaksi opetellessa uutta teknologiaa. Specularityn mahdollisuudet avautuisivat vasta, kun sen oppisi täysin, sitä ennen virheiden mahdollisuus oli liian suuri. Seuraavaksi Rockodile Games tutustui Metalness työskentelytapaan. Metalness tarjosi lähes identtisen lopputuloksen huomattavasti helpommin. Tässä vaiheessa mukaan tuli myös Substance Painter ohjelma, jossa tekstuurien aikaansaamia muutoksia 3d-mallissa pystyi katselemaan reaaliajassa, siinä missä aiemmin työ täytyi keskeyttää nähdäkseen sen hetkisen lopputuloksen.

Gosbane pelin tuli toimia hyvin nykysukupolven konsoleilla, mikä tarkoitti tarkasti rajattua tekstuurikokoa sekä polycounttia. Rockodile Games laski, että Godsbanessa näytöllä saisi olla yhtäaikaisesti n. 2 miljoonaa polygonia, polygonita. Tekstuurikoot rajoitettiin suurissa objekteissa 2048 x 2048 pikseliin, sekä pienemmissä 1024 x 1024 tai jopa 512 x 512. Diffuse tekstuurin oli oltava mahdollisimman tarkka sen sisältämien yksityiskohtien takia, ja se olikin pidettävä täydessä koossa. Normal mapin koko riippui sen sisältämistä yksityiskohdista, ja tarpeen tullen sen kokoa voitiin laskea puoleen. Roughness map ja metalness map eivät sisällä niin tarkkoja yksityiskohtia koskaan, ja niiden koko laskettiin usein puoleen alkuperäisestä.

Yhden graafisen assetin tekeminen oli usean vaiheen projekti. 3d-mallin tekeminen joko konseptin perusteella, tai ilman, oli ensimmäinen vaihe. Rockodile Games käyttää Zbrush-mallinnusohjelmaa tähän. Zbrushissa tehtävä 3d-malli, niin kutsuttu Hi-Poly, sisältää usein monia miljoonia polygoneja, ja sellaisenaan se on peliin aivan liian raskas. Esimerkkinä Godsbanen pelihahmojen rajoitus oli alle 20 000 polygonita per hahmo. Tämä tarkoitti sitä, että 3d-mallien polycount piti pudottaa muutamasta miljoonasta kymmeniin tuhansiin.

Seuraava vaihe assetin tekemisessä oli low-poly-mallin aikaansaaminen. Low-poly malli tarkoittaa vähäisen polygoni määrän omaavaa 3d-mallia. Low-poly malli tehdään Hi-poly mallin muotoja seuraillen, hävittäen mahdollisimman vähän Hi-poly mallin yksityiskohdista. Lo-poly malli tehtiin Blender ohjelmalla. Seuraavaksi Lo-poly malliin merkittiin saumakohdat, joista 3d-malli voidaan ”leikata auki” 2d-kuvaksi UV-koordinaatistoon.

Seuraavaksi 3d-malliin tehdään tekstuurit. X-normal ohjelmalla verrataan Hi-poly-mallin muotoja Lo-poly mallin muotoihin. Niiden erojen pohjalta X-normal muodostaa 2d-tekstuureita: Normal map, curvature map sekä ambient occlusion map. Roughness map, metalness map ja diffuse map tehdään Substance Painter ohjelmassa, joka osaa myös käyttää hyväkseen X-normalilla tuotettuja tekstuureita.

Lopulta tekstuurit yhdistetään materiaalissa, joka Gosdbane projektissa oli pelkkä tekstitiedosto. 3d-mallin Lo-poly versio, tekstuurit sekä materiaali siirretään pelimoottorissa määritettyyn kansiopolkuun, jotta peli osaa käyttää niitä hyväkseen.

Godsbane pelin viimeisessä asussa oli näkyvillä vain tekemiäni 3d-malleja. Objekteja peliin tein projektin aikana ~60, jokaiseen 4 eri tekstuuria eli ~240. Godsbane peliprojekti kuitenkin keskeytettiin Rockodilen todettua se kannattamattomaksi. Siinä oli vai-

keuksia niin projektin skaalassa kuin omaperäisyydessäkin. PBR-renderöinti on kuitenkin yksi niistä suurista asioista joita projektista jäi käteen.

Kuva 13. Godsbane pelin early alpha kuvaa, juuri ennen projektin loppumista.

Godsbanen jälkeen aloitimme projektin Hear No Evil protyypin kehittämisen. Tässä vaiheessa mukaan tuli valmis pelimoottori Unreal Engine 4, jossa PBR-teknologia oli sisään rakennettuna. Unreal Engine 4:n hyödyt kääntyivät nopeasti eduksi vaikka uusi pelimoottori toi paljon uutta opeteltavaa. Pelimoottoria vaihdettaessa työn kulku ei kuitenkaan muuttunut Godsbane-projektista, vaan 3d-mallit sekä tekstuurit tehtiin samoilla työkaluilla samassa järjestyksessä. Uuden pelimoottorin ansiosta 3d-mallien testaaminen pelin sisällä oli kuitenkin huomattavasti nopeampaa ja suoraviivaisempaa valmiiden työkalujen ansiosta.

Hear No Evil –peliprojekti oli kaikilta osin paljon suunitellumpi kokonaisuus kuin Godsbane. Godsbane projekti oli lopulta minulle enemmän PBR-teknologian testailua, sillä

työskentelytapa vaihdettiin muutaman kerran projektin aikana. Unreal Engineen valmiiksi rakennetut valon käyttäytymismallit sekä eri PBR-työskentelytavat mahdollistivat meille nopeamman testailun sekä näyttävempien tilojen suunnittelun.

Kuva 14. Hear No Evil pelin päävalikkoruudun taustakuva. Rakennettuna meidän omista 3d-malleistamme.

Nopean iteroinnin ansiosta saimme Unreal Enginellä aikaan PBR-teknologiaa tukevien efektien hyödyntämisen. Visuaaliset efektit johin emme olisi omalla pelimoottorilla pystyneet kuin vasta pitkän ajan päästä, olivat nyt meillä vain muutaman tunnin työn takana. PBR-teknologia toimi hyvin yhteen Unreal Enginen tarjoamien valo- sekä partikkeliefektien kanssa.

Kuva 15. Pelikuvaa Hear No Evil –projektista suoraan Unreal Enginestä 4:stä

Unreal Engine 4:n tuomat työkalut nopeuttivat ja ohjasivat pienen tiimimme työmäärää kaivatulla tavalla. Pääsimme Hear No Evil –projektissa kolmen kuukauden työn aikana pitemmälle kuin olimme päässyt koko Godsbanen kahdeksan kuukautta kestäneen kehityksen aikana. Vaikka oman pelimoottorin rakentamisen ansiosta olimme saaneet syvempää ymmärtämistä PBR-teknologiasta kuin olisimme suoraan valmiiseen moottoriin hypätessä saaneet, olivat hyödyt kuintekin niin moninkertaiset, että oman pelimoottorin rakentamisen yrittäminen tuntui melko hullunkuriselta. Hear No Evil –projekti on prototyyppi asteella ja visuaaliselta tyyliltään usean ihmisen kehuma. Kaivattu lopputulos saatiin lopulta aikaan valmiilla, pienelle tiimille soveltuvalla pelimoottorilla.

7 YHTEENVETO

PBR-teknologia on uutta pelimaailmassa ja sen omaksuminen aikaisessa vaiheessa tulee olemaan peliyritykselle eduksi. Rockodilen kaksi peliprojektia tarjoili oivan tilaisuuden opetella PBR-renderöinnin tarjoamia mahdollisuuksia. Tekstuurien laaja-alaisempi käyttö tuli tutuksi sekä useat ohjelmat joita uuden tyyppiset tekstuurit pakottivat käyttämään. Näiden kahden projektin aikana opettelin seistemän uutta ohjelmistoa joko vain perusteiltaan tai vaadittavalle ammattilaistason asteelle. Uusi teknologia toi myös uusia haasteita, sillä siitä löytyi todella vähän virallista informaatiota. Useimmat asiat tuli opittua yksittäisten henkilöiden tekemistä youtube-tutoriaaleista tai foorumi viesteistä.

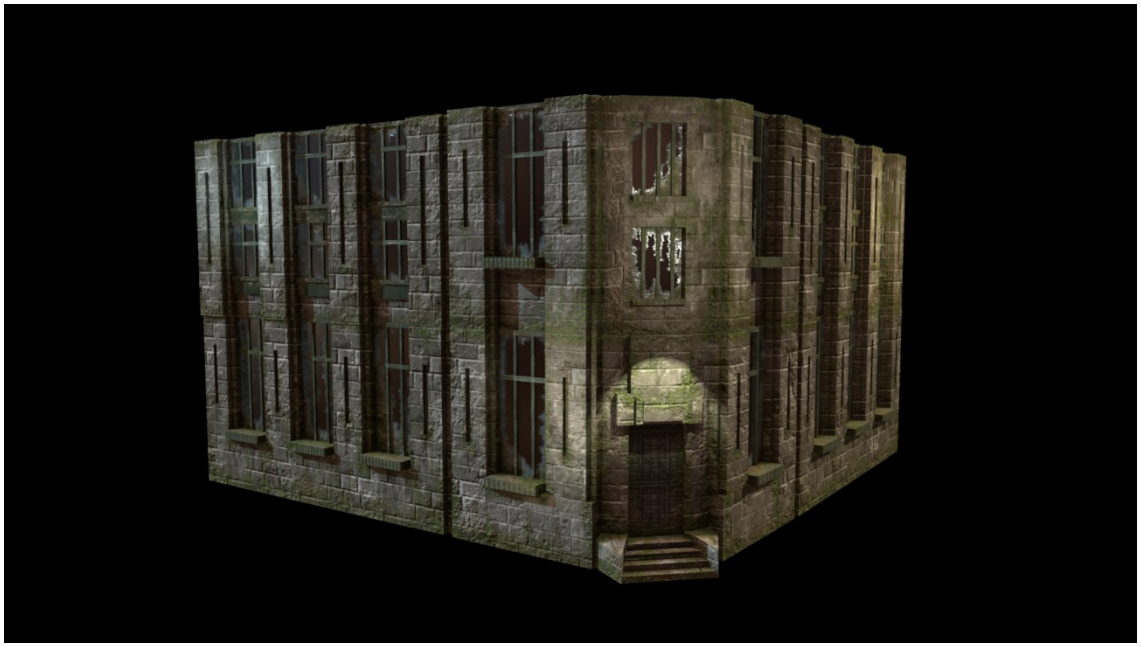
Rakensimme pelimoottorin ensimmäiseen projektiimme itse. Tämä tarkoitti PBR-teknologian osalta sitä, että meidän tuli implementoida PBR-renderöinnin vaatimat shaderit pelimoottoriimme itse. Vaikka oma pelimoottori myöhemmin hylättiinkin sen työmäärän vuoksi, opimme PBR-renderöinnistä paljon syvällisemmin, kuin valmiissa pelimottorissa sen käyttämisestä oppii. Näin jälkikäteen katsottuna ensimmäinen projektimme oli minulle enemmän syvällinen oppimistilaisuus pelimoottoreihin, renderöintiin, tekstuureihin ja 3d-mallinnukseen kuin se oli yritys tehdä myyvä peli. Vaikka projekti oli alusta alkaen tuhoon tuomittu, opimme siitä tiiminä niin paljon, että valmiiseen pelimoottoriin siirtyminen sujui kitkatta.

Unreal Engine 4:ään siirtyminen tuntui samalta kuin siirtyminen kynästä ja paperista tietokoneella kirjoittamiseen. Moni asia oli niin paljon yksinkertaisempaa, että se jätti tilaa suunittelulle ja iteroinnille. Tämä mahdollisti seuraavan projektimme Hear No Evil:n harppauksen grafiisessa laadussa verrattuna Godsbaneen. Hear No Evil projektissa siirryimme täysin metalness-työskentelytapaan PBR-renderöinnissä.

LIITTEET

Työnäytteitä

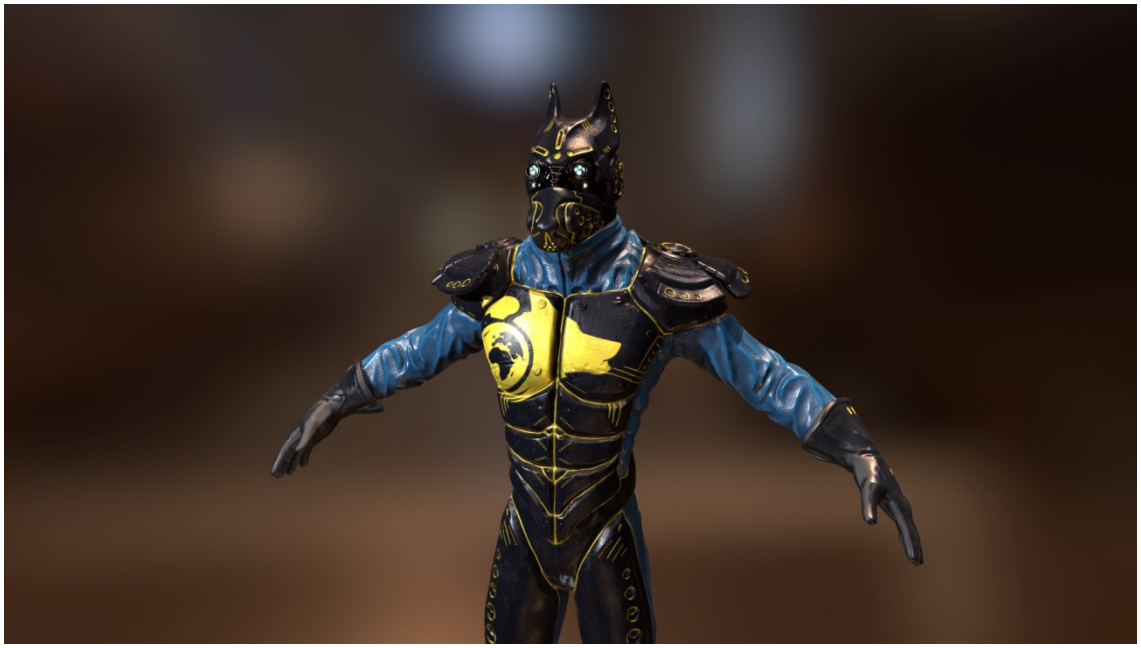
Näiden työnäytteiden tarkoitus on tuoda esille, miten PBR-teknologia soveltuu niin hahmojen, rakennusten kuin tavaroidenkin renderöintiin. Kuvissa vaihtelee valoasetukset sekä taustat, mutta PBR-renderöinnin ottaessa vaikutteita aidosta valon käyttäytymisestä 3d-mallit eivät menetä yksitysikohtaisuuksiaan missään valaistusolosuhteissa. Kaikki kuvissa näkyvät elementit ovat minun tekemiäni kaikkia niitä vaiheita soveltaen mitä tässä opinnäytetyössä esiteltiin.



(Vanha harmaa kivitalo joka tehty PBR-renderöintiä käyttäen)

(Punatiilinen talo joka tehty PBR-renderöintiä käyttäen)

(Thompson M1A1 jäljitelmä PBR-renderöintiä käyttäen)



(Scifi hahmo jossa käytetty myös valoa simuloivaa Emission mappia)

(Kokonainen 3d-tila PBR-elementeistä kasattuna)

LÄHTEET

Wilson, J. 2015. PBR Texture conversion
Referenced 24.10.2016
http://www.marmoset.co/toolbag/learn/pbr-conversion

Russel, J. 2015. Basic theory of physically-based rendering
Referenced 25.10.2016
https://www.marmoset.co/toolbag/learn/pbr-theory

Wilson, J. 2015. Physically-based rendering, and you can too!
Referenced 25.10.2016
http://www.marmoset.co/toolbag/learn/pbr-practice

Disqus. 2015. PBR Theory
Referenced 25.10.2016
https://learnopengl.com/#!PBR/Theory

Epic Games, Inc. 2016. Physically based materials
Referenced 4.4.2017
http://www.marmoset.co/toolbag/learn/pbr-practice

Wikipedia. 2017. 3d-modeling
Referenced 27.10.2016
https://en.wikipedia.org/wiki/3D_modeling

Wikipedia. 2017. Frame rate
Referenced 27.10.2016
https://en.wikipedia.org/wiki/Frame_rate

Wikipedia. 2017. Rendering
Referenced 28.10.2016
https://en.wikipedia.org/wiki/Rendering_(computer_graphics)

Wikipedia. 2017. UV mapping
Referenced 27.10.2016
https://en.wikipedia.org/wiki/UV_mapping