Rainer Alajuuma

UNITY-PELIMOOTTORILLA TOTEUTETTU KUVAPOHJAINEN RENDERÖINTI RAKENNUSTEKNISESSÄ SOVELLUKSESSA

UNITY-PELIMOOTTORILLA TOTEUTETTU KUVAPOHJAINEN RENDERÖINTI RAKENNUSTEKNISESSÄ SOVELLUKSESSA

Rainer Alajuuma
Opinnäytetyö
Kevät 2016
Tietotekniikan koulutusohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietotekniikan koulutusohjelma, ohjelmistokehitys

Tekijä: Rainer Alajuuma
Opinnäytetyön nimi: Unity-pelimoottorilla toteutettu kuvapohjainen renderöinti rakennusteknisessä sovelluksessa
Työn ohjaajat: Risto Korva ja Marko Savolainen
Työn valmistumislukukausi ja -vuosi: Kevät 2016 Sivumäärä: 52 + 1

Opinnäytetyö toteutettiin tilaajan, Group Builder Oy:n tarpeesta saada 3D-mallinnetun talon visuaalinen näyttävyys parhaimmalle mahdolliselle tasolle.

Työssä rakennettiin yrityksen tarvitsemille 3D-objekteille erilaisia shadereita eli eräänlaisia grafiikkaan liittyviä kooditiedostoja, joiden avulla näyttävyyttä saatiin tehostettua mm. kiiltävyyden ja heijastavuuden osalta. Käytettyinä työkaluina olivat monialustainen pelimoottori Unity3D ja siihen Asset Storesta saatavilla ollut maksullinen lisäosa Shader Forge. Työssä eteneminen tapahtui vaiheittain. Aluksi tutkittiin työhön soveltuvia työkaluja ja valinnan jälkeen tutustuttiin työkalun ominaisuuksiin ja mahdollisuuksiin. Rakentaminen tapahtui erilaisten kokeilujen ja yhdistelmien kautta. Internetissä olevia ohjeita ja valmiita malleja pyrittiin hyödyntämään tarpeiden mukaan.

Lopputuloksena tuotteelle eli talojen 3D-mallinnusohjelmalle saatiin näyttävä ulkoasu erilaisten shaderien kokonaisuudella. Työn tilaaja oli tyytyväinen tuotteen laatuun.

Asiasanat: Unity3D, Shader, Shader Forge, renderöinti, 3D-mallinnus

ALKUSANAT

Haluan kiittää työharjoittelupaikan ja opinnäytetyöaiheeni mahdollistanutta GroupBuilder Oy:tä ja sen koko henkilökuntaa. Työn mahdollistamiseksi yritys tarjosi käyttööni tietokoneen ja kustansi työkalun käyttöön tarvittavan lisenssin. Aihe oli mielestäni mielenkiintoinen ja uudenlainen toteutustapa graafisen ulkoasun yksinkertaiseen ehostamiseen. Haluan lisäksi kiittää opinnäytetyöni ohjaajaa Risto Korvaa useiden palaverien aikana esiin tulleista hyvistä ideoista ja jäsentelystä työhöni liittyen. Kiitokset myös Tuula Hopeavuorelle, joka tarkisti työn useaan otteeseen sekä kertoi virhekohdista ja parannusehdotuksista.

Oulussa 17.5.2016

Rainer Alajuuma

SISÄLLYS

SANASTO

API	Ohjelmointirajapinta, jonka mukaan eri ohjelmat suorittavat pyyntöjä ja keskustelevat keskenään.
Asset	Asset Storesta ilmaiseksi tai maksullisena ladattava Unityn lisäosa, koodi, äänitiedosto tai muu taiteellinen luomus.
Asset Store	Asset Store on Internetissä toimiva markkinapaikka Unityn käyttäjille, jossa voidaan myydä ja ostaa taiteellisia luomuksia, valmiita koodeja, äänitiedostoja jne. Tarjolla on myös paljon ilmaisia asseteja.
CG	Tulee sanoista C for Graphics. Toisin sanoen C-kieli Graafiseen toteutukseen. Nvidian vastine Microsoftin HLSL:lle.
Cubemap	Ympäristön kartoittamistapa. Kuution kuusi eri sivua yhdistetään kyseisen objektin kuvakulmasta otetuksi 360 asteen muodostamaksi kokonaiskuvaksi.
GB-CAD	CAD Tulee sanoista Computer-aided Design. Tietokoneohjelma, jonka avulla voidaan piirtää asuntojen pohjapiirustuksia. GB-CAD on yrityksen oma CAD-versio.
GB4D	CAD-ohjelmalla piirretyn pohjakuvan perusteella muodostettu 3D-malli asunnosta.
GB-CORE	GB-ohjelman selainympäristö.
Geometry Shader	Shader, jolla voi suorittaa erinäisille 3D-objektin muodoille laskutoimituksia, lisätä uusia pisteitä tai

	liikutella niitä. Voidaan lisätä tai poistaa dynaamisesti meshin yksityiskohtia.
HLSL	High Level Shading Language, eli ns. korkean tason varjostus kieli. Shaderille kehitetty oma koodikieli.
IBL	Lyhenne sanoista Image-based Lighting. 3D-renderöintitekniikka, joka mahdollistaa oikean maailman (real-world lighting) tasoisen objektin valaistuksen käyttämisen.
Lightmap	Työkalu, jolla virtuaalisen kohtauksen pintojen kirkkaus esilasketaan ja tallennetaan myöhempään käyttöön. Yleisimmin käytetään paikallaan oleviin objekteihin.
Light Probe	Työkalu, jolla muodostetaan kolmiulotteisen objektin ja tilan heijastus. Mahdollistaa lisäksi liikkuvien objektien heijastukset ja valoon reagoinnin reaaliajassa.
Mesh	Peliobjektit koostuvat lukuisista kolmioista järjestettynä 3D-tilaan, aikaan saaden vaikutelman kiinteästä esineestä. Ikään kuin verkko, jossa ilmenee vain objektin muodon tieto.
Node	Shader Forgessa käytettyjä "solmukohtia" eri ominaisuuksien yhdistelemiseksi.
Normal map	Tekniikka, jolla 3D-mallinnuksessa saadaan enemmän yksityiskohtia objektin tekstuuriin ikään kuin huijaamalla töyssyjen ja epätasaisten kohtien valaistusta. Tekniikan avulla objektille saadaan yksityiskohtia käyttämättä enempää polygoneja kuin alkuperäisessä objektissa.

Pixel Shader	Käsittelee kaikista tarkimmin objektin aluetta. Määrittää pikselin tarkkoja tietoja 3D-objektille, esimerkiksi valaistukseen ja väriarvoon liittyen.
Polygoni	Monikulmio eli polygoni on geometriassa tasokuvio, jonka reuna on suljettu murtoviiva. Murtoviivaa seuraamalla päätyy lopulta takaisin lähtöpisteeseen.
Post-Process	Shaderille annettava ominaisuus, jolla saadaan aikaiseksi koko ruudun alueella vaikuttavia ominaisuuksia, kuten mustavalkoisuus, seepia jne. Renderöidään yleensä vasta lopuksi.
Renderöinti	Kuvan mallintamista tietokoneohjelman avulla. Laskee shaderien ohjelmakoodin perusteella halutut ominaisuudet. Viimeistelee erikoistehosteet ja muotoilun antaen mallille viimeisen silauksen.
Shader	(Suomeksi ns. varjostin). Määrittää tarkasti, miten valo, varjot, värit ja heijastukset piirretään tietokoneohjelman graafisessa objektissa.
Shader Forge	Asset Storesta maksullisena saatava asset, jolla voidaan rakentaa Unityssä näyttäviä varjostuksia pelimaailman objekteille.
Tekstuuri	Tekstuuri on tekniikka tuoda 3D-grafiikkaan lisää todenmukaisuutta ja näyttävyyttä ilman, että polygonirakennetta tarvitsee muokata. Käytännössä mallin perusmuoto pinnoitetaan bittikarttakuvalla eli tekstuurilla.
Unity inspector	Unityn päänäkymä, jossa kaikki ominaisuudet ja valikot sijaitsevat.

1 JOHDANTO

Rakennusala monien muiden alojen ohella on nykypäivänä suuresti teknillistynyt ja tullut entistä riippuvaisemmaksi teknisistä laitteista ja sovellutuksista. Nykypäivän tekniikka mahdollistaa jopa jo kokonaisten talojen 3D-tulostamisen. Tämän opinnäytetyön aiheeksi muodostui 3D-ympäristön eri objektien ehostaminen shaderien eli eräänlaisten grafiikkaan keskittyvien kooditiedostojen avulla.

Yksi tämän hetken monista teknisen ja visuaalisen rakentamisen apuvälineitä tarjoavista uusista yrityksistä on GroupBuilder Oy, joka perustettiin vuonna 2012. Yrityksen ohjelmisto on ainutlaatuinen ja se yhdistää kolme täysin omaa elementtiä: GB-Cadin, GB-Coren ja GB3D:n. Toisin sanoen ohjelmisto hyödyttää useita eri ihmisryhmiä rakentamisen alkutaipaleelta viimeiseen vaiheeseen asti, niin suunniteltaessa rakennusta kuin varsinaisessa rakentamisvaiheessa ja lopulta valmiin rakennuksen virtuaalisessa sisustusvaiheessa. Viimeisen vaiheen lopullinen tavoite on aikaansaada asiakkaalle paras mahdollinen kuva tuotteesta ja helpottaa asunnon ostopäätöstä. Tätä varten tarvittiin mahdollisimman hienoja shadereita.

GB3D-nimellä toiminut projektin vanhempi versio ei ollut täysin GroupBuilderin omaa käsialaa, vaan osa tuotteen toiminnoista oli tuotettu muissa yrityksissä, kuten VividWorks Oy:ssä. Nykyisessä GB4D-projektissa tuotetut vaiheet ovat täysin GroupBuilderin omia. GB4D-tiimi sai alkunsa maaliskuussa 2015, kun yritys kävi esittelemässä toimintaansa OAMK Tekniikan yksikössä. Haastattelujen kautta yritys poimi muutamia opiskelijoita uuteen kehitystiimiinsä mukaan ja kaikki aloitettiin puhtaalta pöydältä.

Opinnäytetyö sai alkunsa 3D-sovelluksen visuaalisen näyttävyyden tehostamisen tarpeesta. Visuaalinen puoli on yksi tuotteen tärkeimmistä asioista. Sovellusta kehitetään Unity3D-ympäristössä, johon on saatavilla useita eri asseteja, toisin sanoen lisäosia sekä kehitystyökaluja. Opinnäytetyöni tärkein työkalu Shader Forge mahdollisti kehittää yrityksen tarvitsemille 3D-objekteille erilaisia shadereita.

2 UNITY3D-SOVELLUSKEHITYS

2.1 Unity3D

Unity3D-pelimoottorin kehitys sai alkunsa vuonna 2001, ja Unity on nykyisin yksi suosituimmista ja käytetyimmistä pelien sekä sovellusten kehitysalustoista. Unityn maailmanlaajuinen markkinaosuus on 45 prosenttia ja sillä on 4,5 miljoonaa rekisteröitynyttä kehittäjää ja 600 miljoonaa alustalla kehitettyjen pelien pelaajaa. Nykyisistä markkinoista useat mobiilipelit on kehitetty juuri Unityn avulla ja tuettuina alustoina ovat mm. iOS, Android ja Windows Phone. Unityllä on yhteistyökumppaneina monia suuria yrityksiä, kuten Microsoft, Sony, Qualcomm, Samsung, Nintendo, Oculus VR ja Intel. (1.)

Unity Technologies on monikansallinen yritys, jonka päätoimisto sijaitsee San Franciscossa, Yhdysvalloissa. Lisäksi sillä on toimistoja myös Euroopan alueella esimerkiksi Tanskassa, Ruotsissa ja Suomessa. Unity Technologies kehitti Unityn monialustaiseksi pelimoottoriksi, jolla voidaan rakentaa kaksi- ja kolmiulotteisia sovelluksia vaikkapa yrityksien käyttöön, niin virtuaalisten harjoitussimulaattoreiden kuin pelienkin muodossa. Alustoina voivat toimia PC-laitteet, mobiililaitteet sekä useat eri konsolit. Lisäksi pelien ja sovellusten toimiminen on mahdollistettu Internet-selaimissa Web Playerin tai WebGL:n avulla ilman, että laitteelle tarvitsee asentaa mitään. (1; 2.)

Unitystä on saatavilla niin ilmainen (Unity 5 Personal Edition) kuin maksullinenkin versio (Unity 5 Professional Edition) yksityiseen tai yrityskäyttöön. Maksullinen Unity Pro maksaa n. 70 euroa/kk:ssa tai kertasuorituksella 1393,53 euroa. Unityyn on saatavilla myös muita maksullisia lisäosia eri mobiilialustoilla kehittämisen tueksi (esim. iOS Pro, Android Pro ja Team license). (1.)

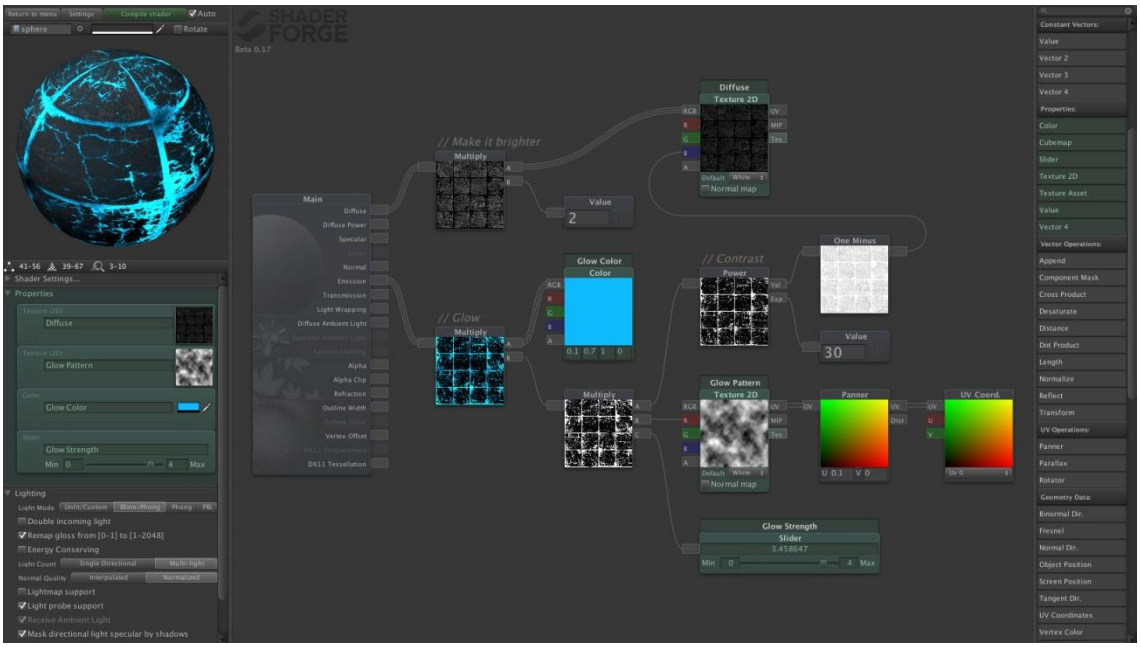
Unity tekee pelien kehittämisestä helppoa ja monipuolista helppokäyttöisen ja loogisen käyttöliittymänsä avulla. Tuettuina ohjelmointikielinä ovat JavaScript, Boo ja C#, joista käytetyin on C# (2).

2.2 Asset Store

Marraskuussa 2010 julkaistu Unity Asset Store tuo yhteen sovelluskehittäjät ja asiakkaat sekä helpottaa sovelluskehitystä. Se on markkinapaikka kaikille Unityyn saatavilla oleville lisäosille (asseteille), jotka voidaan luokitella mm. graafisiksi taidonnäytteiksi, valmiiksi ohjelmointikoodeiksi, äänitiedostoiksi jne. Kuka tahansa voi laittaa omia tuotoksiaan tarjolle joko maksullisena tai ilmaisena. Kukin tuote käy läpi tarkat vaatimuskriteerit ennen hyväksymistä. (3; 4.)

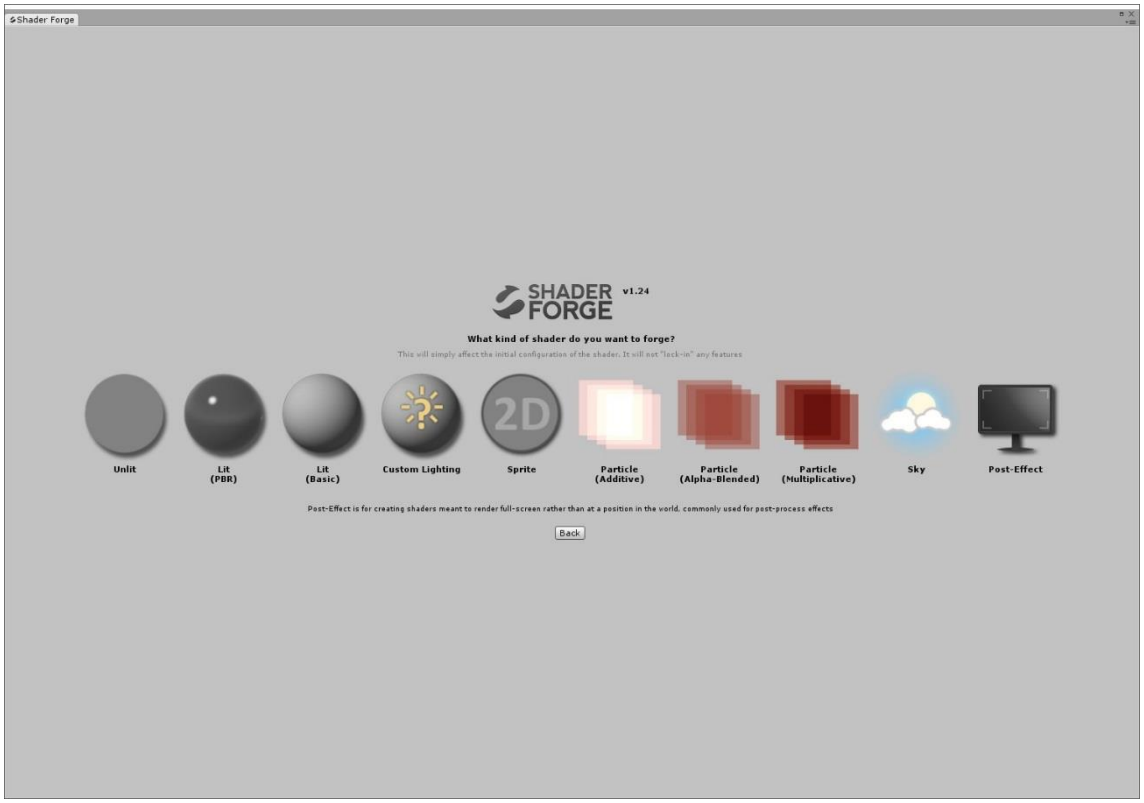
2.3 Shader Forge

Shader Forge on yksityishenkilön, Joachim Holmérin (Neat Corporation) kehittämä palkittu Unity-lisäosa, joka mahdollistaa shaderien, eli ns. varjostinten suunnittelutyön ilman minkäänlaista koodaustaitoa. Shaderit pitävät sisällään esimerkiksi valon sijaintiin ja väriin liittyviä tietoja. Shader Forge on käytettävissä niin ilmaisessa Unityn versiossa kuin maksullisessa Pro-versiossakin. Yleensä shadereita suunniteltaessa ne aikaansaadaan nimenomaan kirjoittamalla koodia, joka pitää sisällään monimutkaisia laskutoimitusten yhtälöitä. Shader Forgessa kaikki rakentuu erilaisia nodeja (solmuja) visuaalisesti yhdistelemällä (kuva 1). Shader Forge on myös monien tunnettujen peliyritysten suosiossa (esim. Valve, Blizzard, DICE, Ubisoft, Bungie, EA, Avalanche Studios, Rovio). (5.)



KUVA 1. Shader Forge (5)

Kukin Shader Forgen node vastaa jotakin laskutoimitusta varsinaisessa kooditiedostossa. Lisäosan parhaimpia puolia on sen helppokäyttöisyys. Työkalu käynnistetään klikkaamalla Unityn ylävalikosta Window -> Shader Forge. Avautuvassa ikkunassa voidaan avata valmis työ tai luoda uusi. Uutta luotaessa aukeaa näkymä, jossa on jo valmiita pohjaratkaisuja erilaisin kiilto- ja valaistusominaisuuksin (kuva 2). Uuden shaderin voi tietysti aloittaa puhtaalta pöydältäkin valitsemalla avautuvasta näkymästä ensimmäisen pohjaratkaisun, unlit eli valaisematon.



KUVA 2. Kuvakaappaus Shader Forgen pohjaratkaisuista

Kuvan 1 vasemmassa ylälaidassa näkyvillä on ns. preview-näkymä, josta voidaan tutkia reaaliajassa työn alla olevan shaderin näkyvyyttä objektissa. Objektia voidaan pyöritellä tai antaa sen pyöriä itsekseen. Esikatselun alapuolella on useita valikkoja erilaisten asetusten säätämiseen. Keskellä on työpöytänäkymä, jossa erilaisia nodeja yhdistellään isoon pääkytkentäalustaan muodostaen lopulta laajemman node-rakenteen. Oikeassa laidassa näkyvissä olevasta vierityslaatikosta voidaan valita erilaisia nodeja ja vetää ne työpöydälle. Shader Forgessa on myös monenlaisia pikanäppäinyhdistelmiä, joilla saadaan haluttu komponentti esimerkiksi painamalla jokin kirjain pohjaan ja valitsemalla kirjaimella alkava tietty node. Kaikkia nodeja on mahdollista liikutella ja raahata haluttuun kohtaan. Lisäksi monen noden valinta ja siirto onnistuu Alt-näppäin pohjassa ja hiirellä ylivetämällä. Nodepuu on toisin sanoen helppo pitää juuri siinä loogisessa järjestyksessä kuin käyttäjä itse haluaa.

Shaderin rakentaminen tapahtuu yksinkertaisesti vetämällä viiva kunkin noden laatikon reunoissa olevista liitäntäkohdista toiseen ja yhdistämällä lopulta rakenne pääkytkentäalustan halutun ominaisuuden liitäntään. Kussakin node-laatikossa on input- ja output lähdöt eli sisääntulo- ja ulosmenoportti. Jokainen node antaa siihen oman vaikutuksensa, joko värin, tekstuurin tai vaikkapa valon oikean sijaintitiedon muodossa. Tulos on näkyvissä välittömästi preview-näkymässä, mikäli shaderin päivittävä Auto-valinta on rastitettuna. Käyttäjä voi myös itse päivittää manuaalisesti painamalla Compile shader -nappia.

3 SHADER FORGEN SISÄLTÖ

Projekti lähti liikkeelle tilaajan tarpeesta saada visuaalista näyttävyyttä 3D-malleille. Kun kyseessä on asunnon myymistä edistävä sovellus, on graafisen ulkonäön oltava hiottua, jotta ostajalle välittyy mahdollisimman hyvä ja viimeistelty kuva hänelle saatavilla olevista eri vaihtoehdoista ja valinnoista.

3.1 Shaderin määritelmä

Shaderista puhuttaessa tarkoitetaan eräänlaista tietokoneohjelmaa, jolla jonkin graafisesti mallinnetun esineen pinta saadaan oikeaoppisesti varjostettua. Toisin sanoen valaistus määritetään oikein perustuen esineen kulmaan suhteessa aurinkoon tai muuhun valonlähteeseen. Ohjelma kertoo, mille esineen alueelle kuuluu tietyn kirkkausasteen taso ja missä se lähtee himmentymään. (6.)

Pääasiassa shaderit on tarkoitettu tietokoneen graafisen prosessointiyksikön (GPU = Graphics Processing Unit) suoritettavaksi, jossa laskentateho saadaan maksimoitua. GPU:sta puhuttaessa tarkoitetaan tietokoneen grafiikkaprosessoria, jossa vaativimmat grafiikkaan liittyvät laskutoimitukset mm. shaderien ohjelmakoodin perusteella suoritetaan. (6.) Shadereita voidaan kuitenkin suorittaa myös prosessorin (CPU = Central Processing Unit) avulla, jolloin GPU:lla suorittaminen ei ole pakollista.

Shaderit ja erilaiset varjostinkielet ovat syrjäyttäneet vanhemman mallisen toteutustavan fixed-function pipelinestä eli ns. muunneltavasta prosessointitilasta, jota käytettiin ennen shaderien tuloa. Vanhassa toteutuksessa graafinen toteutus perustui käyttäjän tuottamaan muunneltavuuteen, kun taas nykyisin grafiikka toteutetaan käyttäjän tuottamissa ohjelmissa, eli shadereissa. (7.)

Renderöinti on tärkeä sana puhuttaessa 3D-mallinnuksesta, grafiikasta ja shadereista ylipäätään. Shaderit tarjoavat grafiikkaan liittyvän ohjelmakoodin renderöitäväksi. Renderöitäessä esineen geometria, katselukulma, tekstuuri,

valaistus ja varjostus yhdistetään yhdeksi kokonaisuudeksi erinäisten kuvien avulla ja lopputuloksena tavoitellaan mahdollisimman aitoa eli fotorealistista tuotosta. Joskus renderöinnissä saattaa kestää huomattavankin kauan, koska ohjelma käy koodin läpi jopa yksittäisten pikselien tarkkuudella ja laskee jokaisen määritetyn värin ja valoisuuden määrän. (8.)

3.2 High-Level Shading Language (HLSL)

Shader Forge, kuten muutkin vastaavat node-pohjaiset shaderien luomiseen tarkoitetut lisäosat, pohjautuu HLSL:ään, eli ns. korkean tason varjostuskieleen. HLSL on Microsoftin kehittämä koodikieli, jonka avulla voidaan kehittää C-kielen tapaisia ohjelmoitavia varjostimia (9). Se esiteltiin samassa yhteydessä DirectX 9:n kanssa (9; 11). HLSL:n suurin etu on, että se tuo shaderit ikään kuin käyttäjän ulottuville ja helpommaksi muokata edeltäjiinsä, esimerkiksi Assemblyyn verrattuna (9). Lisäksi yhteensopivuus eri laitteiden ja alustojen välillä on paljon parempi. Assembly oli koodikielenä ns. symbolinen konekieli ja huomattavasti vaikeampi toteutustapa nykyisin tunnettuun C-kieleen nähden. Myös virheiden mahdollisuus oli merkittävästi suurempi Assemblyllä kirjoitettaessa. (12.)

3.3 HLSL-osiot

HLSL sisältää pääsääntöisesti samoja kielen ominaispiirteitä kuin C-kieli. Alla on englanniksi tiivistetysti luokiteltuna tärkeimmät osiot:

- Language Syntax (DirectX HLSL)
- Shader Models vs. Shader Profiles
- Intrinsic Functions (DirectX HLSL)
- Asm Shader
- D3DCompiler
- Inline Format Conversion
- Appendix (DirectX HLSL)
- HLSL errors and warnings.

Language Syntax, eli ohjelmointikielen syntaksi. HLSL vaatii ohjelmointikielen syntaksin ymmärtämistä. Tässä osiossa on määritelty mm. koodi muuttujien määrittelemiseen ja alustamiseen, käyttäjäkohtaisten shaderien funktioiden määrittely sekä funktioiden vahvistaminen määrittämällä eräänlaisia valintoja, mitä polkua funktion tulisi seurata milloinkin. (10.)

Shader Models vs. Shader Profiles määrittää DirectX-ohjelmointirajapinnan eri versioille soveltuvat säännöt ja rajoitteet varjostimen mallista riippuen (Shader Model 1–5). Kaikki se koodi mitä käyttäjä Vertex-, geometria- tai pikselishaderille asettaa, vahvistetaan tietyn Shader Modelin kautta ohjelmakoodin kääntämisvaiheessa. (10.)

Intrinsic Functions eli toisin sanoen vapaasti käännettynä valmiit funktiot, joiden toiminta on jo testattu hyvin toimivaksi ja ne on käännetty käyttäjän vapaasti käytettäväksi omien määritettävien funktioiden lisäksi (10).

Asm Shader pitää sisällään Assembly-määritelmiä ohjelmoinnin ja shaderien kääntämistä varten (10).

D3DCompiler kääntää raa'an HLSL-lähdekoodin, kuten funktiot, tietueet, listat, luettelot ja vakiot (10).

Inline Format Conversion osio pitää sisällään D3DX_DXGIFormatConvert.inl-tiedoston, joka käsittelee formaatin muutosfunktioita ja rakenteita, joita tarvitaan shaderin satunnaisten tietojen laskemiseen ja mm. pikselishaderien käyttöön. Tällöin em. tiedosto täytyy sisällyttää ohjelmaan ja laitteiston tulee tukea DirectX11 3D:tä. (10.)

Appendix on sisällytetty ohjelmiin ja muodostaa viimeisen osan kokonaisuutta, se sisältää listauksen sekä avain- että varatuista sanoista. Kyseisiä sanoja ei voida käyttää tunnisteina käyttäjän omissa ohjelmakoodeissa. Appendix sisältää myös koodin kieliopin, eli käytännössä tietyt säännöt HLSL:n lauseiden muodostamiseen. (10.)

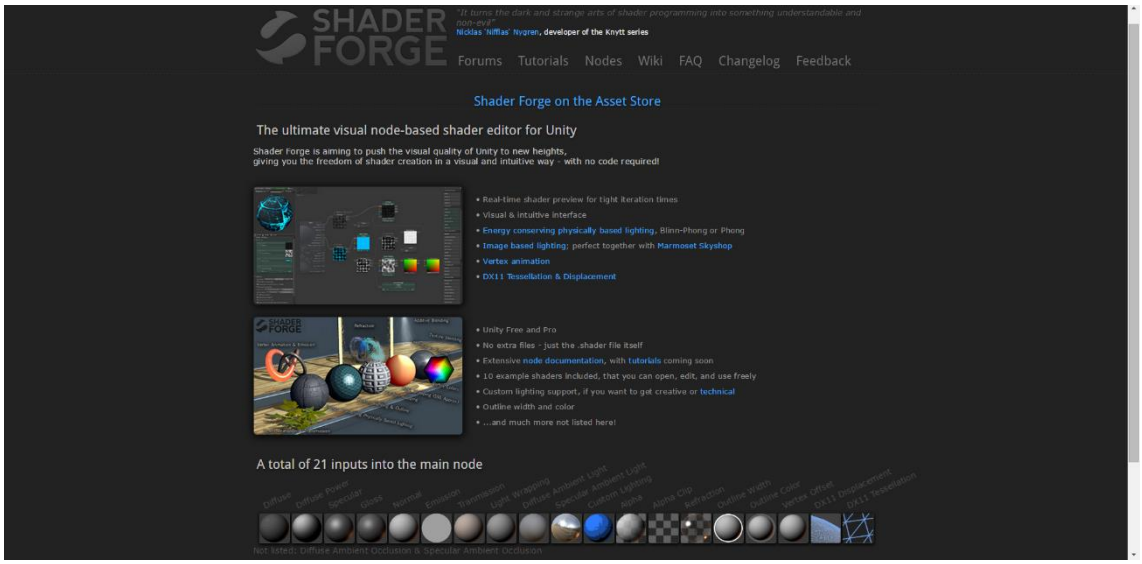
HLSL errors and warnings osio tuottaa virhe- ja varoituskoodit, jotka shader voi näyttää käyttäjälle. (10.)

3.4 Shader Forgeen tutustuminen

Opinnäytetyön merkityksellisyys muodostui uudenlaisesta ja mahdollisesti helppokäyttöisemmästä toteutustavasta rakentaa erilaisia näyttäviä shadereita. Työtä lähdettiin taustoittamaan tutustumalla Asset Storessa tarjolla oleviin erilaisiin shaderin mallinnustyökaluihin, joista Shader Forge havaittiin monipuolisimmaksi ja parhaimman näköiseksi käytettävyydeltään. Kyseinen lisäosa Unityyn ei ollut ilmainen. Hinnaksi muodostui 83,61 euroa. Tuotteen ostamisesta päätettiin ja tutustuminen saattoi alkaa.

Ennen tuotteen ostamista oli mahdollista tutustua siihen erilaisten kuvakaappausten ja videoiden avulla, joskaan tarjolla olevia videoita ei ollut opinnäytetyön suorittamisen ajankohtana vielä kovinkaan paljoa tuotteen uutuudesta johtuen. Shader Forgen toteuttanut Joachim Holmér pitää myös yllä tuotteen ominaisuuksia havainnollistavia verkkosivuja, Wiki-sivua ja foorumia erilaisten ongelmien ilmoittamiseen ja muuhun keskusteluun.

Tärkeimpänä tietolähteenä tuotteen käyttämiseen luonnollisesti olivat kehittäjän omat nettisivut (5) (Kuva 3). Niiden kautta pääsi tutustumaan Shader Forgen keskeisimpiin ominaisuuksiin, yhteensopivuuteen mahdollisten muiden Assetien kanssa ja itse tuotteen käyttämiseen. Sivustolla selitettiin jokaisen käytettävissä olevan noden merkitys ja toiminta, joskin aloittelijan näkökulmasta edellä mainitut asiat tuntuivat hieman vaikeilta ja totuttelua vaativilta. Osa nodeista oli jokseenkin puutteellisesti selitetty ja ikään kuin oletettu käyttäjän jo entuudestaan tietävän tiettyjä asioita. Hyvänä lisävaihtoehtona työkalun käytön harjoitteluun olivatkin videot.



KUVA 3. Shader Forgen nettisivut

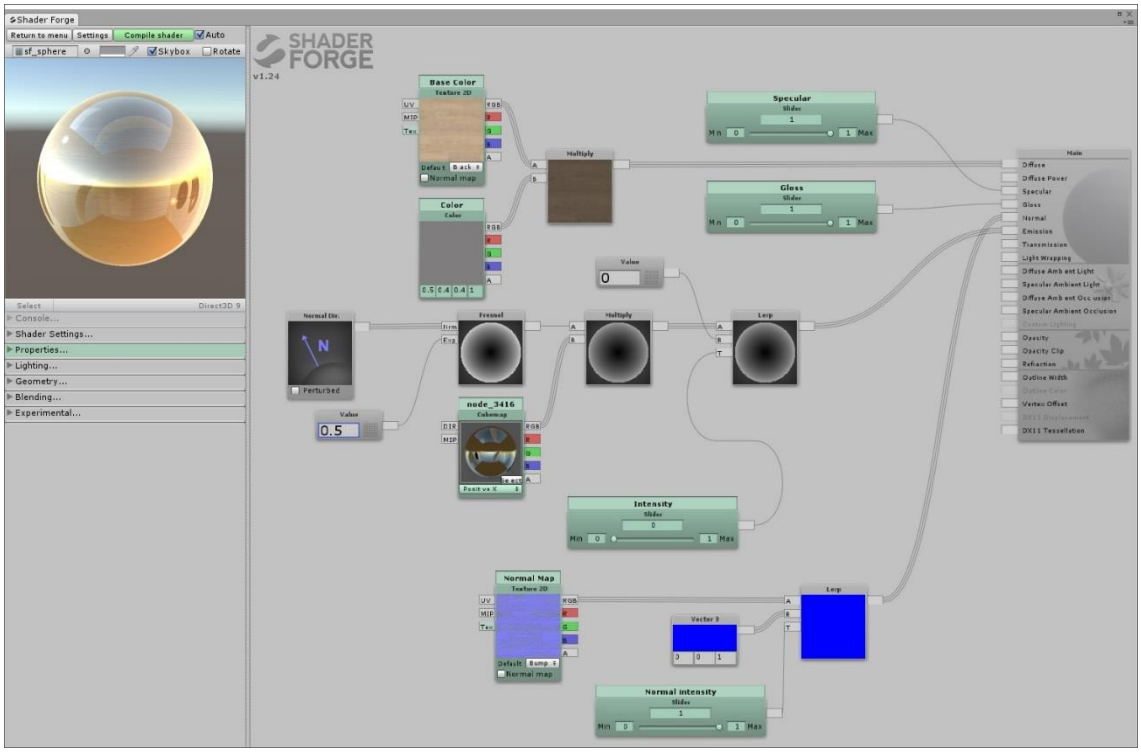
Foorumi kävi tarpeelliseksi monessa opinnäytetyön vaiheessa. Sieltä oli mahdollista lukea ongelmista ja ohjelman mahdollisista virheistä, sekä kysyä apua tietynlaisen toiminnon luomiseen. Ohjelman kehittäjä Holmér vastaili kysymyksiin ja ehdotuksiin parhaansa mukaan ja mielestäni tuotteen kehitys ja tuki olivat hyvällä tasolla ottaen huomioon kehittäjän tekevän toistaiseksi yksin kaiken.

3.5 Shader Forgen käyttäminen ja ominaisuudet

Käyttö aloitettiin valmiilla pohjilla, joissa oli jo joitain nodeja lisättynä. Valmiisiin pohjiin oli videoita, ja erilaisia kuvia katsomalla helppo muokata omia haaroja eli liittää yhä useampia nodeja toisiinsa ja katsoa samalla jatkuvasti päivittyvää preview-ikkunaa kulloisestakin tuotoksesta. Käytännössä käyttöliittymä mahdollistaa rajattoman määrän toteutustapoja, joskaan kaikki kytkennät nodejen välillä eivät ole mahdollisia. Sovellus ilmoittaa mahdollisesta kytkennästä vihreällä värillä tai tietyn olemassa olevan kytkennän korvaamisesta punaisella värillä. Mikäli kytkentä ei ollut ollenkaan mahdollinen tiettyyn porttiin, oli se himmeänä ja poissa käytöstä siinä vaiheessa, kun kytkentä aloitettiin.

Shaderien rakentelu itsessään oli kohtuullisen helppoa ja mielenkiintoista. Erilaisia Internetistä poimittuja ratkaisuja pystyi yhdistelemään ja saamaan vielä hienompia kokonaisuuksia aikaiseksi. Shadereihin oli myös mahdollista yhdistää joko itse luotu cubemap ohjelmakoodin toteuttamana tai hankkia cubemapin luontiin valmis sovellus vaikkapa Asset Storesta. Cubemapin avulla shaderille saatiin lisäarvoa realistisen kaltaisilla ympäristön heijastusominaisuuksilla (kuva 4).

Joissain tapauksissa preview-ikkunan näkymä saattoi hävitä kokonaan tai muuttaa värinsä täysin pinkiksi. Silloin oli helppo arvata tehneensä jonkin virheellisen kytkennän. Kytkentä voitiin peruuttaa viemällä hiiri noden tai pääkytkentäalustan liitäntäkohtaan ja painamalla näppäimistöltä Alt + hiiren oikea painike. Kyseinen lanka meni tuolloin punaiseksi ja ilmoitti käyttäjää hyvin, mikä osa oltiin aikeissa poistaa. Käytännössä ohjelma ei millään muulla tavoin ilmoittanut virheellisestä toteutuksesta. Mitään koodausohjelmille tyypillistä teksti-ikkunaa virheiden ilmoittamiseen ei toisin sanoin ollut.



KUVA 4. Kuvakaappaus ympäristöä peilaavasta shaderista

3.6 Shader Forgen koodi

Unityn alkuperäiset shaderit muodostetaan ns. toteavalla koodikielellä, joka on nimetty ShaderLabiksi. Se pohjautuu kaikesta huolimatta samaan koodikieleen, jota esimerkiksi CG ja HLSL ovat. Shader Forge käyttää päällisin puolin samaa HLSL-kieltä, mutta sisältää joitakin eroavia piirteitä. (25.) Shader Forgen HLSL sisältää kuitenkin seuraavat shaderin tyypilliset ominaispiirteet:

Syntax eli shaderin nimi, joka on lainausmerkkien sisällä ja aaltosulkujen sisällä varsinainen sisältö, joita ovat mm. properties, subshaders, fallback ja shader pass. Shaderin kooditiedostossa voi olla ainoastaan yksi shaderin määrittävä rivi (kuva 5). (25.)

```
Shader "name" { [Properties] Subshaders [Fallback] [CustomEditor] }
```

KUVA 5. HLSL Syntax

Properties muodostaa shaderin varsinaisen näkyvän sisällön, johon käyttäjä voi vaikuttaa Unityn päänäkymässä asettaessaan shaderin jollekin materiaalille. Propertyjä ovat esimerkiksi väri, tekstuuri tai jokin arvo esimerkiksi kiillolle. (25.)

Subshaders ja fallback ovat käytännössä joukko aliohjelmia, jotka käydään läpi shaderin latausvaiheessa. Niistä käytetään vain sitä, jota käyttäjän laitteisto tukee. Mikäli mitään subshadereista ei tueta, ohjelmakoodin suoritus siirtyy shaderin loppuun, jossa fallback shader on määritetty. Sen avulla käyttäjälle asetetaan jokin Unityn vakioshadereista, kuten VertexLit. Tällä taataan kaikille käyttötapauksille soveltuva ratkaisu ja ei tarvita kirjoittaa useita shadereita erikseen. (25; 26.)

Shader pass on subshaderin sisälle kirjoitettu koodipätkä, joka yleensä sisältää valon tai tekstuurin asettamiseen liittyvää tietoa.

Pääasiallisin ero Shader Forgen automaattisesti generoimassa koodissa verrattuna alkuperäiseen HLSL-koodiin, on kooditiedoston ylimmäisin rivi heti kommenttirivien jälkeen (kuva 6). Se on ohjelman kehittäjän itse määrittelemiä true-, ja false-arvoja sekä dataa, joka on tärkeä sisällyttää ohjelmakoodiin toimivuuden takaamiseksi. Datalla luultavimmin sinetöidään luotujen shaderien alkuperä. Yläosan kommenttiriveistä ilmenee myös Shader Forgen versionumero, kehittäjän nimi, nettisivun osoite ja varoitus: *”Note: Manually altering this data may prevent you from opening it in Shader Forge”*. Tämä tarkoittaa, että mikäli käyttäjä muuttaa valmista generoitua ohjelmakoodia, ei shaderia välttämättä enää saada auki Shader Forgessa. Loppuosassa shader-tiedostoa on myös koodipätkä:

```
CustomEditor "ShaderForgeMaterialInspector"
```

Tällä määritetään Shader Forgen yhteydessä toimiva preview-näkymä, joka päivittyy aina automaattisesti tai käyttäjän itse nappia painamalla, kun ohjelmakoodia on muutettu.

```
1  // Shader created with Shader Forge v1.26
2  // Shader Forge (c) Neat Corporation / Joachim Holmer - http://www.acegikmo.com/shaderforge/
3  // Note: Manually altering this data may prevent you from opening it in Shader Forge
4  /*SF_DATA;ver:1.26;sub:START;pass:START;ps:flbk:,iptp:0,cusa:False,bamd:0,lico:1,lgpr:1,limd:3,spmd:1,
5
6  Shader "Shader Forge/pbr" {
7      Properties {
8          _BumpMap ("Normal Map", 2D) = "bump" {}
9          _Color ("Color", Color) = (0.5019608,0.5019608,0.5019608,1)
10         _MainTex ("Base Color", 2D) = "white" {}
11         _Metallic ("Metallic", Range(0, 1)) = 0
12         _Gloss ("Gloss", Range(0, 1)) = 0.8
13     }
14     SubShader {
15         Tags {
16             "RenderType"="Opaque"
17         }
18         Pass {
19             Name "FORWARD"
20             Tags {
21                 "LightMode"="ForwardBase"
22             }
23
24
25             CGPROGRAM
26             #pragma vertex vert
27             #pragma fragment frag
28             #define UNITY_PASS_FORWARDBASE
29             #define SHOULD_SAMPLE_SH ( defined (LIGHTMAP_OFF) && defined(DYNAMICLIGHTMAP_OFF) )
30             #define _GLOSSYENV 1
```

KUVA 6. Kuvakaappaus Shader Forgen koodista

4 SHADERIEN LUOMINEN

4.1 Aloittaminen

Shader Forge -lisäosan hankkimisen jälkeen voitiin avata Unityssä haluttu projekti, jonka yhteyteen tuote päätettiin asentaa. Sen jälkeen lataus aloitettiin joko Unitystä suoraan valitsemalla *Window -> Asset Store* ja hakemalla Shader Forge, jonka jälkeen sivulla painettiin ”Install”. Toinen vaihtoehto oli avata Internet-selaimen kautta Asset Store ja hakea asennettava lisäosa. Tiedostojen latauduttua Unityssa avautuvasta ikkunasta valittiin ”Import”. Shader Forgeen tarvittavat tiedostot liitettiin tällöin mukaan projektiin.

Shader Forge avattiin valitsemalla Unityn ylävalikosta *Window -> Shader Forge*. Avautuvasta päänäkymästä (kuva 7) lähdettiin yleisimmin liikkeelle. Toinen tapa oli valita suoraan Unityn päänäkymää tarkasteltaessa jokin peliobjekti, joka sisälsi materiaalin yhteydessä aiemmin tehdyn shaderin. Tällöin voitiin suoraan painaa ”Open in Shader Forge”.

KUVA 7. Kuvakaappaus Shader Forgen päänäkymästä

Lisäosan päänäkymästä valittavissa olivat seuraavat painikkeet:

- **New Shader** = Luo uusi shader
- **Load Shader** = Lataa aiemmin luotu shader
- **Polycount thread** = Pelien grafiikan mallintamiseen liittyvä sivusto, jossa Shader Forgella oma osionsa kysymyksiä ja mallien esittelemistä varten
- **Unity thread** = Unity3D-foorumin Shader Forge -osio ideoita, pohdintaa, avun kysymistä yms. keskustelua varten
- **Node Documentation** = Joachim Holmérin omat kotisivut, joissa dokumentoituna kaikkien Nodet ja niiden käyttötarkoitukset
- **Wiki** = Holmérin oma Wiki-osio Shader Forgelle, jonne myös kuka tahansa muukin voi kirjoittaa tietoa sekä lisätä valmiita shadermalleja
- **Credits** = Näyttää Shader Forgen kehitystyössä mukana olleiden henkilöiden nimet ja muuta tietoa
- **Post bugs & ideas** = Avaa User Echon Shader Forge -osion, jossa voidaan kertoa ohjelmavirheistä, äänestää asioista ja pyytää apua ongelmiin
- **Forums** = Avaa Holmérin omien verkkosivujen yhteyteen luodut foorumit, jossa voi myös kysyä apua ongelmiin tai erilaisiin shaderien toteutustapoihin, katsoa tiedotteita ja keskustella asioista yleisesti ottaen

4.2 Shader Forgen asetukset

Shader Forge sisältää suurehkon valikoiman asetuksia. Asetukset sijaitsevat käyttöliittymän vasemmassa palkissa, jossa näkyy myös 3D-näkymä parhaillaan työn alla olevasta shaderista. Näkymän 3D-objektia voidaan vaihtaa haluttaessa vieressä olevasta Mesh-valikosta. Lisäksi myös tausta voidaan ottaa pois, laittaa objekti pyörimään tai vaihtaa taustan väriä shaderin hahmottamisen helpottamiseksi. Ylälaidasta valittavissa ovat myös ”Return to menu”, josta päästään takaisin päävalikkoon sekä ”Settings”, joka avaa pienen lisäasetuksia sisältävän näkymän. Settings-valikon avulla voidaan vaikuttaa Shader Forgen

käytettävyyteen ja työpöydän näkymiin. Viimeisenä on ”Compile shader” -nappi, jota painamalla voidaan itse ajaa läpi viimeisimmät muutokset ja tallentaa shader. Vieressä on myös rastitettava valinta ”Auto”, joka on oletuksena päällä ja shaderin ohjelmakoodi käännetään jokaisen muutoksen jälkeen automaattisesti.

3D-näkymän alapuolella alkavat varsinaiset asetukset, jotka aukeavat pienestä väkäsestä klikattaessa.

Console ilmoittaa joistakin selkeästi virheellisistä node-kytkennöistä sekä ehdottaa mahdollista korjausta tarjoamalla suoraan painikkeen toiminnon suorittamiseksi. Unityn omaan Consoleen verrattuna Shader Forgen vastine ei tarjoa läheskään yhtä tarkkoja virhe- tai varoitusilmoituksia. (Kuva 8.)

Shader Settings sisältää joitakin laatuun ja kohdealustaan liittyviä asetuksia. **Path** antaa käyttäjän päättää polun, mistä kyseinen shader löytyy. **Fallback** määrittää käyttöön Unityn oman ns. varashaderin, jota käytetään, mikäli käyttäjän laitteisto ei tue luodun shaderin asetuksia. **LOD** eli Level Of Detail-arvo määrittää shaderin laadun toisin sanoen yksityiskohtien tarkkuuden (21). Vakiona se on ns. rajattomassa tilassa. Tyypillisesti lukuarvon 100–600 väliltä antamalla voidaan rajoittaa shaderin laatua, jos halutaan sen toimivan paremmin vaikkapa heikommalla näytönohjaimella varustetulla laitteistolla (21). **Allow using atlased sprites** -valinta rastitettuna sallii useita eri kohtaan objektia sijoitettavia kuvia sisällään pitävien tekstuurien käytön. **Draw call batching** -valikosta on valittavissa ”Enabled”, ”Disabled” tai ”Disabled during LOD fade”. Käytännössä valinta mahdollistaa tai estää objektin piirtämiseen liittyvän pyynnön grafiikkaan liittyvälle API:lle (22). **Inspector preview mode** antaa valittavaksi, onko näkymässä 2D- vai 3D-objekti. **Target renderers** sisältää shaderin kohdealustat: Direct3D 9, Direct3D 11, OpenGL, OpenGL ES 2.0, OpenGL ES 3.0, iOS Metal, Direct3D 11 for Windows RT/Phone, Xbox 360, Xbox One, Playstation 3, Playstation 4 ja Playstation Vita. (Kuva 8.)

Properties näyttää shaderin yhteyteen luodut valinnat, jotka tulevat yleensä näkyviin Unity-käyttäjälle inspector-näkymässä, ellei shaderin luoja ole

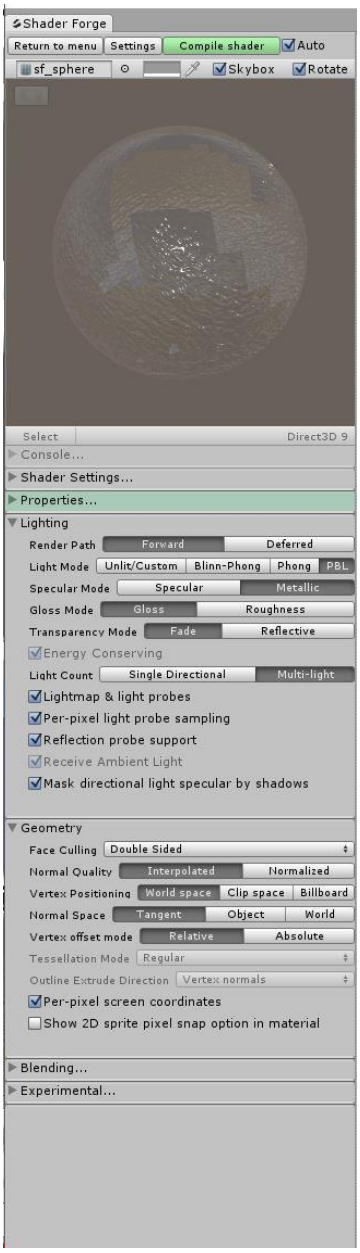
päättänyt piilottaa niitä. Valinnat tuovat käyttäjälle Unityn päänäkymään tekstuurien valitsemisen, värin vaihtamisen joko Color- tai Vector 4 -valinnan avulla, cubemapin lisäämisen sekä erilaisten valaistusasetusten vaihtamisen liukuvalitsinta tai lukuarvoa muuttaen. (Kuva 8.)

KUVA 8. Asetukset: Console, Shader Settings, Properties

Lighting käsittelee täysin valaistukseen liittyviä asetuksia. **Render Path** -kohdasta valitaan renderöintipolku joko Forward tai Deferred. **Light Modella** voidaan vaikuttaa nodejen pääkytkentäalustan valintamahdollisuuksiin. Valon asetukseksi voidaan valita joko Unlit/Custom, Blinn-Phong, Phong tai PBL. **Specular Mode** -kohdasta voidaan valita joko Specular tai Metallic, riippuen minkä tyyppistä shaderia valoisuuden suhteen ollaan tekemässä. **Gloss Mode** antaa valittavaksi joko Gloss tai Roughness eli voidaan määrittää, kuinka kiiltävästä tai ”karkeasta” shaderista on kyse. **Transparency Modessa** valittavana on Fade tai Reflective sen mukaan, halutaanko shaderin olevan valoa himmentävä vai heijastava. Lisäksi voidaan rastittaa **Energy Conserving**, joka ikään kuin yhdistää Glossin ja Specularin arvot, jolloin Glossin lisääminen tai vähentäminen tekee saman Specularin arvolle (14). **Lightmap & light probes** -rasti täytyy olla valittuna Unity 5 -versiossa. Tällöin shader tukee valaistuksen hoitavaa Unityn Enlighten-engineä (15). Vanhalla Beast-enginellä jokainen property täytyi olla oikeaoppisesti nimettynä, muutoin esilasketut valaistukset eivät menneet oikein (15). **Per-pixel light probe sampling** -valinta antaa mahdollisuuden renderöidä pikselintarkkuudella light proben tuottama ympäristönheijastuma kyseiselle pinnalle, kuten esimerkiksi liikkuvalle pelihahmolle (23). Valinnan tekeminen saattaa lisätä näytönohjaimelta vaadittavaa suorituskykyä. **Reflection probe support** -valinta antaa luodulle shaderille tuen reflection proben käyttöön. **Receive Ambient Light** on yleensä automaattisesti valittuna ja antaa shaderin ottaa vastaan ympäristönvaloa eli vaikkapa Scenen Directional Lightin valoa, jota usein käytetään ns. aurinkona. **Mask directional light specular by shadows** naamioi directional lightin specular-arvon varjojen mukaan. (Kuva 9.)

Geometry-asetusvalikko käsittelee pelkästään shaderin geometriaa. Ensimmäinen pudotusvalikko **Face Culling** antaa valittavaksi, onko objektin etu- vai takapuoli näkyvillä tai kummatkin puolet, jos ”Double Sided” on valittuna. Hyödyllinen esimerkiksi kangastyyppisissä objekteissa, jossa molemmille puolille halutaan jotakin näkyvää shaderin avulla. **Normal Quality** korjaa tietyn tyyppisten shaderien vektoreiden pituutta, jos ne ovat jollain tapaa vääristyneet. **Vertex Positioning** muuttaa shaderin katselukulmaa verteksien

osalta. **Normal Space** varmistaa, että vektorit ovat samassa tilassa ja objektin valaistuminen tapahtuu oikein. Valittavissa on ”Tangent”, ”Object” ja ”World”. Yleisesti Tangent on valittuna. (27.) **Vertex offset mode** sallii verteksin etäisyyden vaihtamisen suhteellisen (relative) ja tietyn asetetun pisteen (absolute) välillä. **Tessellation Mode** valintaa voidaan käyttää vain, kun pääkytkentäalustan DX11 Tessellation-porttiin on kytketty esimerkiksi jokin arvo. Valinta antaa mahdolliseksi tavallisen DX11-ominaisuuden käyttöön perustuvan kolmioiden ruudutuksen (regular) sekä reunan pituuteen perustuvan ruudutuksen (edge length based). (16.) **Outline Extrude Direction** määrittää, minkä pisteen kautta objektin ympäröivä ulkoviiva menee. Outline width ja color täytyvät olla kytkettyinä shaderin pääkytkentäalustaan, jotta asetusvalinnat ovat käytettävissä. Valitsemalla **Per-pixel screen coordinates** voidaan käyttää pikselintarkkoja määrityksiä shaderin pinnoille, kun käytössä on Screen Position -node. Tällöin näytetään ruudun senhetkinen kohta meshistä. Valinnan poistamalla käytössä on kokonaiset verteksit objektista**. Show 2D sprite pixel snap option in material** näyttää Unityn inspector-näkymän materiaalissa valinnan, josta voidaan laittaa sprite shaderille pikselintarkka määritys käyttöön tai pois käytöstä. (Kuva 9.)

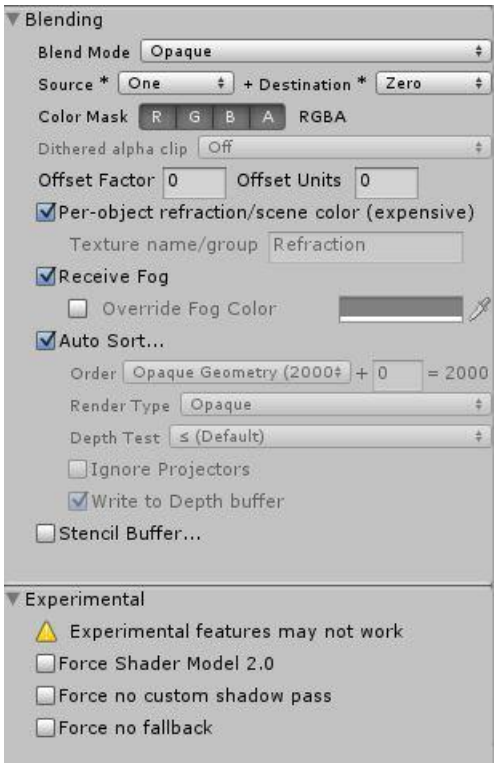


KUVA 9. Asetukset: Lighting, Geometry

Blending-asetukset antavat erilaisia mahdollisuuksia objektin läpinäkyvyyteen liittyen. **Blend Mode** antaa valittavaksi monenlaisia arvoja erityyppistä läpinäkyvyyttä tarvitsevien efektien luomiseen liittyen, joita ovat: Opaque, Alpha Blended, Alpha Blended (Premultiplied), Additive, Screen ja Multiplicative. Niille voidaan määrittää alempana olevista valikoista lähtö- (Source) ja kohdearvot (Destination). Väriarvo voidaan vaihtaa **Color Mask** -valinnan R-, G-, B- ja A-painikkeista. **Dithered alpha clip** antaa mahdolliseksi alpha-arvon

sekoitussävytteisen värin. **Offset Factor** -kenttään voidaan syöttää arvo, joka ulontaa samassa sijainnissa olevaa geometriaa. **Offset Units** on käytännössä sama asia kuin offset factor, mutta ulontaa tietyn yksikön (unit) verran. **Per-object refraction/scene color (expensive)** -valinta rastitettuna antaa joko objekti- tai ryhmäkohtaisen tarkistuksen valon taittumiseen liittyen. Mikäli valinta on päällä, laitteiston resursseja kuluu renderöitäessä enemmän ja tarkistus tapahtuu objektikohtaisesti. Alempana olevaan tekstikenttään voidaan syöttää ryhmän tai tekstuurin nimi. **Receive Fog** -valinta tulee olla rastitettuna, mikäli shader keskittyy sumumaisen efektin luomiseen. **Auto Sort** -valinta on yleensä rastitettuna. Pois otettaessa voidaan manuaalisesti päättää läpinäkyvyyden järjestykseen, renderöinnin tyyppiin ja syvyyteen liittyvistä arvoista. **Stencil Buffer** -valinta rastittamalla voidaan manuaalisesti määrittää eri arvoja ja luoda vaikkapa maski eli peittoalue tietylle osalle objektia tai määrittää, millä alueella shaderin ominaisuus vaikuttaa. (Kuva 10.)

Experimental on ns. ”kokeellisia” asetuksia, jotka eivät välttämättä toimi odotetulla tavalla kaikissa tilanteissa. **Force Shader Model 2.0** pakottaa shaderin käyttämään vanhemman mallista shadermallinnusta, jota käytetään yleisesti suurimmassa osassa Unitya tuetuista mobiililaitteista. Pienentää shaderin resurssien kulutusta. Uudemman tason laitteet tukevat nykyisin myös Shader Model 3.0:aa. (28.) **Force no custom shadow pass** kirjoittaa koodiin kustomoidun varjon ohjauksen, jos shader on asetettu alpha clipped- tai vertex offset-tilaan. **Force no fallback** käyttää luodulle shaderille varjoja vain, mikäli asetuksissa määritetty fallback shader sattuu niitä käyttämään. (Kuva 10.)



KUVA 10. Asetukset: Blending, Experimental

4.3 Shader Forgen valmiit pohjaratkaisut

Shader Forge pitää sisällään kymmenen valmista pohjaratkaisua, jotka tulevat näkyviin heti käyttäjän valittua päävalikosta ”New Shader” (kuva 2). Viemällä hiiren kunkin pohjaratkaisun päälle alla näkyy informaatiota kyseisen tyyppisestä shaderista. Mikään valinta ei lukitse mitään ominaisuuksia varsinaisesta työvaiheesta pois, vaan asetusvalikosta voidaan muuttaa milloin tahansa shaderin tyyppiä. Alla on lueteltuna ja selitettynä pohjaratkaisut.

Unlit

Valonlähteet eivät vaikuta Unlit-shaderiin. Väri perustuu käyttäjän määritykseen ja mikään Unityn Scenen asetus ei vaikuta siihen.

Lit (PBR)

Pyrkii vastaamaan Unityn omaa Physically Based shaderia eli ympäristöön perustuvaa shaderia. Siihen vaikuttavat lightmapit, light probet, reflection probet ym. asetukset.

Lit (Basic)

Vastaa vanhanaikaisena pidettyä tyyliä Blinn-Phong-valaistusmallista, joka mahdollistaa sitä kirkkaamman specularin arvon, mitä lähempää kyseistä objektia katsotaan. Ottaa vastaan vain suoraa valoa, ei lightmappeja tai minkäänlaista dataa probeilta.

Custom Lighting

Mahdollistaa käyttäjän itsemäärittelemiä valaistusmalleja shaderille. Heti pohjaratkaisun valittua Shader Forge rakentaa automaattisesti perinteisen Blinn-Phong-mallisen shaderin, jota voi vapaasti lähteä muokkaamaan.

Sprite

Käytetään 2D-shaderien kehittämiseen esimerkiksi Unityn käyttöliittymän tekstiä ja kuvia varten. Kyseisillä shadereilla on pixel-perfect-valinta eli pikselin tarkkuus. Toimivat hyvin myös muiden 2D Spritejen kanssa.

Particle (Additive)

On yleisesti partikkeliefektejä varten tehty shader-pohjaratkaisu. Mahdollistaa mm. hehkuefektit, valonsäteet, kipinöintiefektit jne.

Particle (Alpha-Blended)

On yleisesti partikkeliefektejä varten tehty shader-pohjaratkaisu. Sisältää Additiveen eroten läpinäkyvyyttä. Mahdollistaa mm. pirstale-efektit, sumun tyyppisen efektin jne.

Particle (Multiplicative)

On yleisesti partikkeliefektejä varten tehty shader-pohjaratkaisu. Tarkoitettu pääasiassa mm. pimennysefekteille, mustalle savulle, eräänlaiselle vastahehkulle jne.

Sky

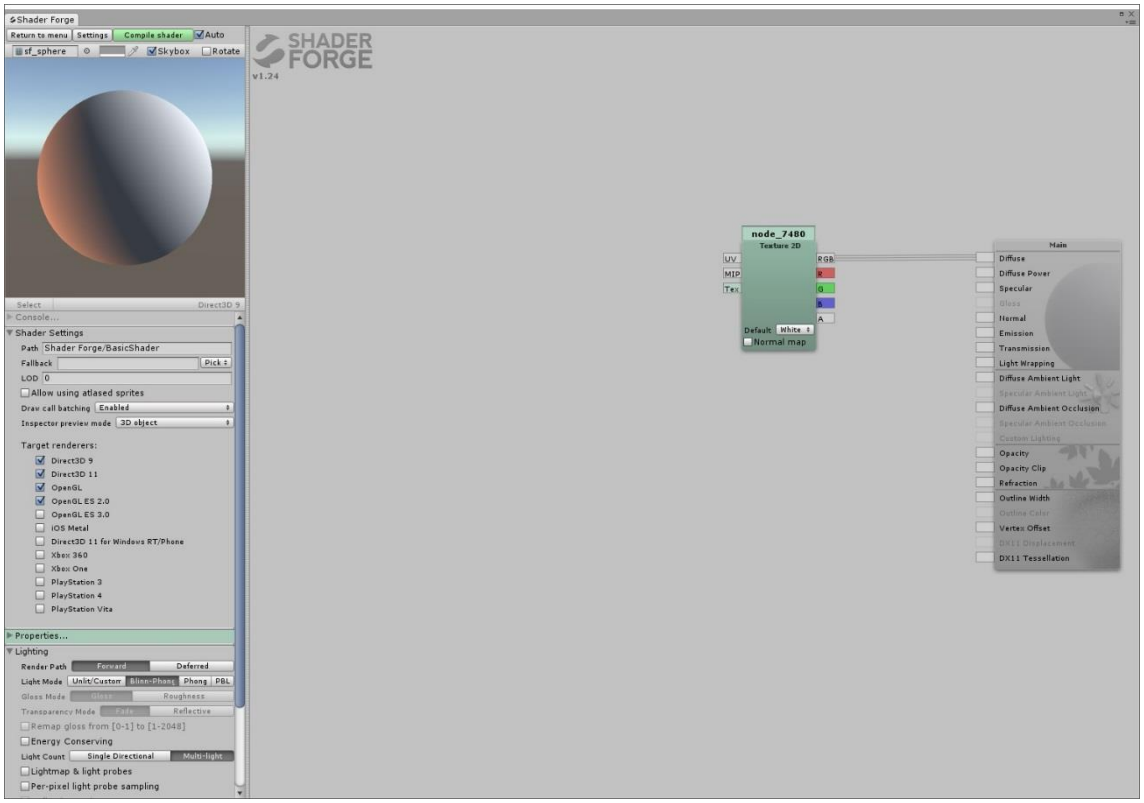
Sky-pohjaratkaisu on tehty Unityn eri taivasmateriaalien yhteydessä käytettäville shadereille. Se renderöidään kaiken muun takana.

Post-Effect

Tarkoitettu koko ruudulla esitettäville shadereille eli ns. filtteriefekteille. Pääasiallisesti käytetään post-process-efektejä varten eli vaikkapa yleisesti koko scenelle lisättyä mustavalkoisuutta tai seepiaefektiä varten. Renderöidään lopuksi kaiken muun jälkeen. (17.)

4.4 Shader Forgen peruskäyttö

Käyttäminen lähtee liikkeelle uuden shaderin luomisesta. Valitaan mieleinen pohjaratkaisu, jolloin aukeaa varsinainen työpöytänäkymä (kuva 11).



KUVA 11. Kuvakaappaus Shader Forgen työpöytänäkymästä

Aiemmin esitellyn asetus- ja esikatselunäkymän vieressä oleva harmaa tila on varattu kokonaan shaderin rakentelua varten. Työpöydän näkymää voidaan myös loitontaa tai lähentää kohtuullisen paljon vierittämällä hiiren rullaa. Alueella liikkuminen tapahtuu hiiren vasen painike pohjassa alustaa siirtelemällä. Näkymän oikeassa laidassa on varsinainen kytkentäalusta, iso harmaa laatikko. Tietyt nodet ja kytkennät liitetään jossain vaiheessa sen input, eli sisääntuloliitäntöihin. Laatikon nimi on Main.

Pääkytkentäalusta pitää sisällään seuraavat kytkentämahdollisuudet:

Diffuse

Shaderin pääväri, joka ottaa vastaan valoa ja muodostaa valon himmentymän ja varjot riippuen valon normaalikulmasta (16).

Diffuse Power

Valon himmentymän normaalikulman eksponentti. Käytetään ns. ekstrametallisen ulkoasun luomiseen silloin, kun arvo on yli 1. (16.)

Specular

Käytetään korostuksena kirkkaan valopilkun luomiseksi kiiltävän esineen pintaan. Mitä korkeampi arvo annetaan, sitä kirkkaampi valopilkku on. Musta ei vaikuta shaderiin millään tavalla. (16.)

Gloss

Gloss eli kiilto on spekulaaristen korostusten eksponentti. Korkeammat arvot tekevät shaderista kiiltävämmän, kun taas nollaa lähestyvät arvot antavat mattapintaisen ulkoasun. Huom. Jos ”gloss remapping” ei ole rastitettuna asetuksista, tulisi välttää alle yhden arvoja. (16.)

Normal

Normal on tangenttiavaruuden normaalisuunta, johon voidaan kytkeä normalmap-tekstuurikarttoja tai käyttäjäkohtaisia normaalivektoreita (16).

Emission

Emission on valoa, jota on käytännössä aina lisättynä shaderiin riippumatta valaistuksen olosuhteista (16).

Transmission

Kontrolloi kuinka paljon valoa pääsee läpi, kun valonlähde on parhaillaan renderöidyn kohteen takana. Voidaan käyttää esimerkiksi ohuisiin materiaaleihin, kuten vaatetukseen tai kasvillisuuteen. (16.)

Light Wrapping

Saadaan aikaan “subsurface scatteringia” muistuttava efekti eli valo voi mennä tietynlaisesta ohuesta pinnasta läpi. Toimii parhaiten tasaisille objekteille.

Punaista väriä annettaessa kietoo väriarvon ympäri objektia saaden sen näyttämään, kuin valo olisi objektin muodon eli meshin sisällä. Valo tulee ulos punaisen eri aallonpituuksilla samoin, kuten esimerkiksi iho on varjostettu. (16.)

Diffuse Ambient Light

Lisää valoa shaderille vaikuttaen diffuseen. Voidaan käyttää esimerkiksi normal directionia hyödyntävän cubemapin yhteydessä, kun halutaan Image Based Lighting- tai Ambient light -tyyppistä efektiä. (16.)

Specular Ambient Light

Lisää valoa shaderille vaikuttaen speculariin. Voidaan käyttää esimerkiksi view directionia hyödyntävän cubemapin yhteydessä, kun halutaan Image Based Lighting -tyyppinen efekti. Käytetään usein samassa yhteydessä Diffuse Ambient Lightin kanssa. (16.)

Diffuse Ambient Occlusion

Heikentää epäsuoraa diffusen valoa, kuten light probeja ja Diffuse Ambient Lightia (16).

Specular Ambient Occlusion

Heikentää epäsuoraa specularin valoa, kuten reflection probeja ja Specular Ambient Lightia (16).

Custom Lighting

Tämä tuloliitäntä on käytössä vain, kun shader on asetettu unlit-tilaan sallien kustomoitua valon käyttäytymistä. Custom Lightingiin kytketyt nodet ovat kukin valokohtaisia. (16.)

Opacity

Kontrolloi pikselin tai palasen tarkkuudella läpinäkyvyyttä. Osittainen läpinäkyvyys on yleisesti pikkutarkkaa saada kohdalleen, varsinkin jos renderöintipolkuna käytetään deferred renderingiä. (16.)

Opacity Clip

Opacity Clipin avulla kontrolloidaan, pitäisikö tietyn pikselin tai palasen piirtyä vai ei. Opacity Clipin käyttöä suositellaan peliobjekteille, jotka eivät tarvitse osittaista läpinäkyvyyttä. Komponentin avulla läpinäkyvyys voidaan jaotella helposti toisin kuin pelkän Opacityn avulla. (16.)

Refraction

Refraction eli valon taittuminen on näyttöruudun alueelle sijoittuva offsetti eli ulontuma heijastamaan takana olevat pikselit. Alpha-arvo suositellaan asetettavaksi alle yhden ennen käyttöä, jolloin heijastusefekti on nähtävissä. (16.)

Outline Width

Lisää ympärysviivan peliobjektille, jossa shader on kiinnitettynä. Huom. kovat reunat rikkovat ympärysviivan. (16.)

Outline Color

Lisäämällä Color -komponentin lisää värin ympärysviivalle, jos Outline Width on kytkettynä (16).

Vertex Offset

Tämän avulla shaderia voidaan animoida tai vaikuttaa peliobjektin muotoon monin tavoin. Käytetään asettamalla XYZ-koordinaatti perustuen siihen, kuinka paljon halutaan yksittäisen verteksin ulontuvan. (16.)

DX11 Displacement

Toimii suurelta osin samalla tavalla kuin Vertex Offset, mutta käytetään DX11-ruudutuksen yhdistämiseen. Huom. DX11 on vain Windows-käyttöön ja tarvitsee sitä tukevan näytönohjaimen ja lisäksi ominaisuus täytyy olla valittuna käyttöön Unityssä. (16.)

DX11 Tessellation

Kontrolloi kuinka moneen alaosastoon kolmiot ruudutetaan. Huom. DX11 on vain Windows-käyttöön ja tarvitsee sitä tukevan näytönohjaimen. Lisäksi ominaisuus täytyy olla valittuna käyttöön Unityssä. (16.)

5 SHADER FORGEN KÄYTTÖTAPAUKSIA

Group Builder Oy:ssa tehtäväkseni muodostui rakentaa 3D-maailman eri objekteille shadereita, joille haluttiin Unityn Standard-shadereista poikkeavaa ulkonäköä tai efektejä. Pääasiassa keskityttiin kiilto-ominaisuuteen, peilaavuuteen, normalmappien rakentamiseen ja asettamaan ne osaksi kokonaisuutta. Shaderin kohdealustaksi asetettiin poikkeuksetta lähes kaikki valinnat. Vähintäänkin Open GL ES 2.0 ja 3.0 tuli olla valittuina, koska projektimme tähtäsi Unity WebGL -pohjaiseen sovellukseen ja ne olivat sen yhteydessä tarvitut rajapinnat.

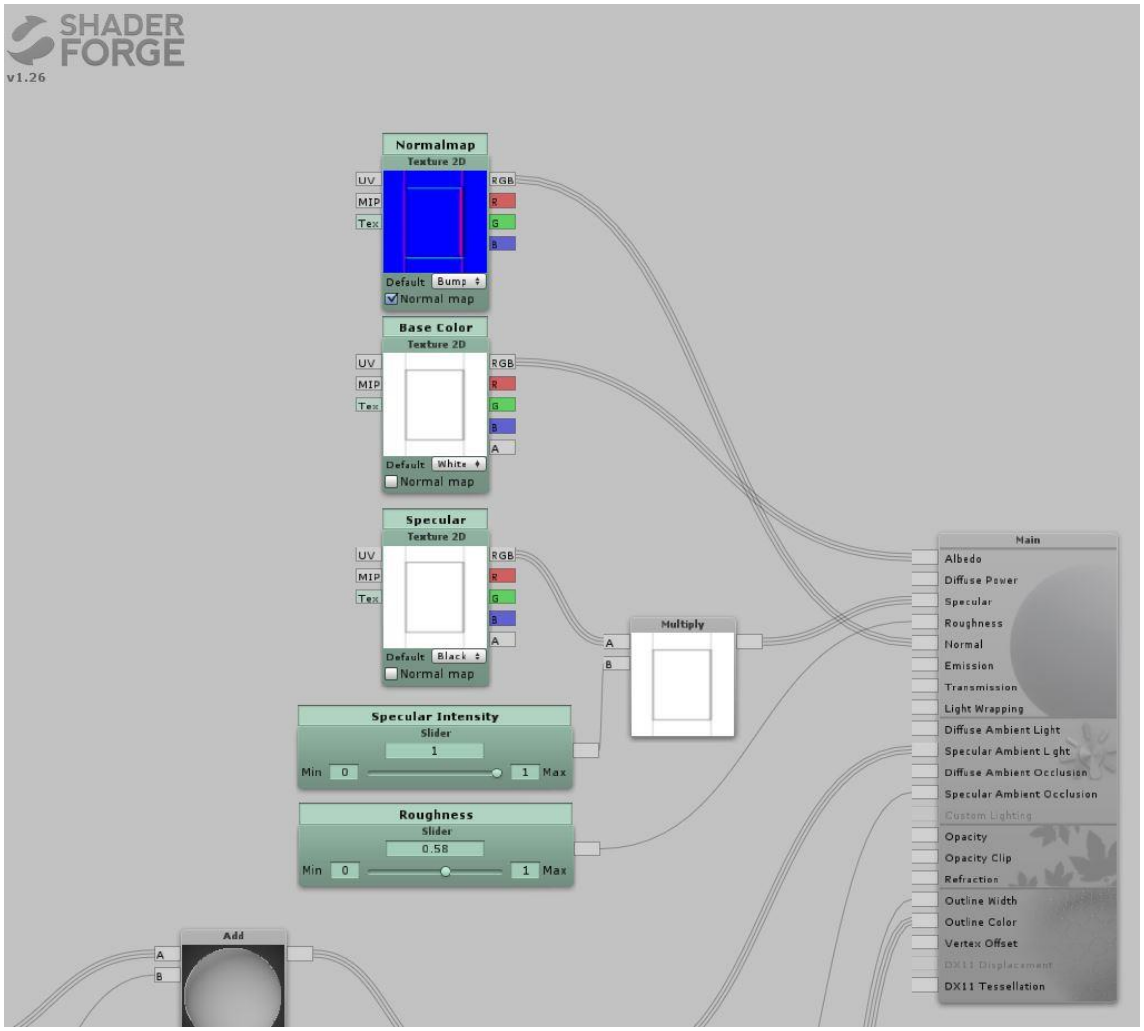
Shadereita rakennellessa huomionarvoiseksi asiaksi paljastui, että Internetistä suoraan otettua HLSL-koodia jollekin valmiille shaderille ei voinut hyödyntää Shader Forgessa täysin kopioituna. Shader Forgen ohjelmakoodi oli likipitäen samankaltaista, mutta siinä oli kuitenkin joitakin omia määrityksiä ja ne piti olla sisällytettynä oikein, jos shaderin halusi saada avattua kustomoitavaksi työpöytänäkymälle. Useimmiten tämä ei kuitenkaan onnistunut ja shader näkyi pinkkinä, toisin sanoen jokin oli virheellisesti.

5.1 Käyttötapaus 1: Shader kaapin oville

Kun kyseessä on asuntojen mallintamiseen ja sisustukseen keskittyvä sovellus, oli mukana lukuisia kaappeja, tasoja ja ovia. Tarpeeksi muodostui saada erinäisille pinnoille erityyppisiä ominaisuuksia esimerkiksi kiiltoa tai mattapintaa.

Shaderin luominen aloitettiin valitsemalla valmiista pohjaratkaisuista tarkoitukseen sopiva lähtökohta. Kaapin ovia varten luoduissa shadereissa käytössä oli valon asetus PBL eli Physically Based Light. Kyseinen shader koostui pääasiassa kolmesta tekstuurikartasta: colormapista, normalmapista ja lisäoptiona specularmapista (kuva 12). Colormapissa määriteltiin käytännössä tarvittu tekstuuri, joka sisälsi väritiedon tietyille kohdille materiaalia ilman valo- tai varjostustietoa (20). Normalmap on yksi monista tekstuurin korkeuteen liittyvistä kartoista, joka keskittyi lähinnä mihin suuntaan geometrian normaalit osoittavat (20). Specular map ei ollut pakollinen läheskään kaikilla pinnoilla,

mutta se toi tiettyihin kiiltävyyttä hakeviin shadereihin lisäarvoa kertomalla, kuinka kirkkaana tai tummana tietyn kohdan mallista pitää esiintyä (20). Normalmapin luomiseen käytin useista ilmaisena tarjolla olevista ratkaisuista NormalMap-Onlinea, jossa tekstuurikartan luomiseen riitti alkuperäisen tekstuurin (kuvatiedoston) pudottaminen ruudun alueella olevaan laatikkoon. Mahdollisuutena oli muokata myös tekstuurikartan korkeutta ja joitain arvoja mm. sävyyn ja tarkkuuteen liittyen ennen tiedoston tallentamista tietokoneelle.



KUVA 12. Tekstuurimapit

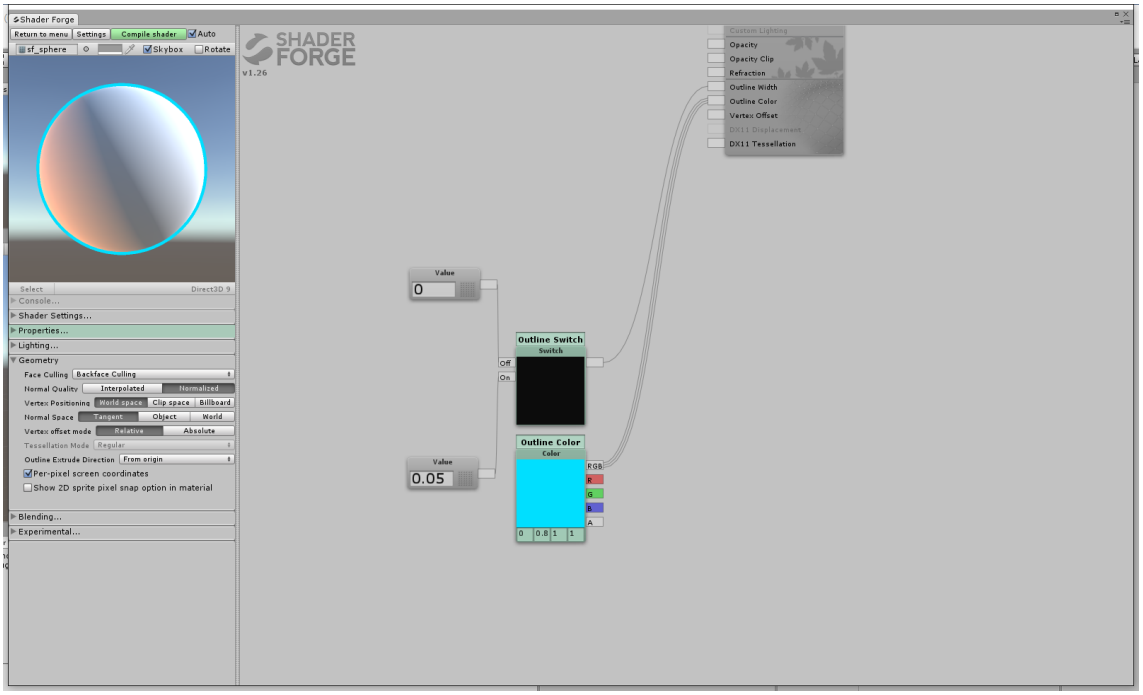
Tekstuurikartat luotiin Shader Forgessa lisäämällä tarvittu määrä Texture2D-Propertyjä. Kyseinen node voitiin luoda valitsemalla vasemman asetuspaneelin ylälaidasta *Settings -> Show node browser panel* ja hakemalla oikeaan laitaan

ilmestyvästä paneelista tarvittu node. Toinen vaihtoehto lisätä node Shader Forgessa oli painaa halutun noden aloittava kirjain näppäimistöltä pohjaan ja valita ilmestyvästä valikosta tarvittu node.

Luotiin kolme Texture2D-propertyä, jotka nimettiin asianmukaisesti klikkaamalla ylhäällä olevaa tekstikenttää. Normalmap kytkettiin pääkytkentäalustan Normal-inputtiin. Base color kytkettiin shaderin asetuksista riippuen, joko nimellä Diffuse tai Albedo esiintyvään ylimpään porttiin pääkytkentäalustasta. Tässä tapauksessa PBL-asetusta käytettäessä se esiintyi nimellä Albedo. Specular kytkettiin myös nimensä mukaiseen porttiin pääkytkentäalustalla. Sitä ennen rakenteeseen lisättiin Slider-property ja Multiply-node. Multiply toimii nimensä mukaisesti kertoimena ja tässä tapauksessa se mahdollistaa haarautuvan kytkennän. Kytkentä tapahtui liittämällä Texture2D-property ja Specular Intensityksi nimetty Slider-property Multiply-nodeen, josta se lopulta liitettiin Specularin pääkytkentäalustalla (kuva 12). Muita shaderiin haluttuja kytkentöjä olivat mm. Specular ambient light ja occlusion. Niiden pääkytkentäalustalla vastaaviin portteihin voitiin liittää Custom Lightingistä otettuja valmiita pohjaratkaisuja mm. valon suuntaan, katselukulmaan ja normaalien kulmaan liittyen sekä voitiin vaikuttaa shaderin kirkkauteen. Slider-propertyjä käyttämällä voitiin asettaa arvot käyttäjän helposti muokattaviksi.

Kun kyseessä on 3D-sovellus, jossa käyttäjä voi klikkailla erinäisiä pintoja ja tehdä valintoja, täytyi esineen valinnan indikoinnille olla jokin säväyttävä efekti. Kokeilin monen tyyppisiä Outline-ratkaisuja eli objektille lisättävää reunaviivaa. Kaapin ovien tapauksessa shader lisättiin vain kaapin oven materiaalille. Yksinkertaisimmillaan em. ominaisuutta tavoitellessa shaderiin lisättiin Switch-property, Color-property ja kaksi Value-nodea. Switch eli kytkin toi Unityn Inspectoriin mahdolliseksi rastittaa valinnan päälle tai pois päältä. Toisin sanoen käyttäjä pystyi määrittämään, milloin halusi asettaa reunaviivan päälle. Kytkentä tapahtui liittämällä Switch-propertyyn kummatkin Value-nodet, toinen On- ja toinen Off-Inputiin. Off sai arvon 0 ja On-arvo esimerkiksi 0.05 riippuen halutun reunaviivan paksuudesta. Switch kytkettiin pääkytkentäalustan Outline Width -porttiin. Color-property määritti viivan värin ja se kytkettiin Outline Color

-porttiin. Geometria-asetuksien Outline Extrude Direction -valintaan asetettiin From origin. (Kuva 13.)



KUVA 13. Kuvakaappaus Outlinen toteutuksesta

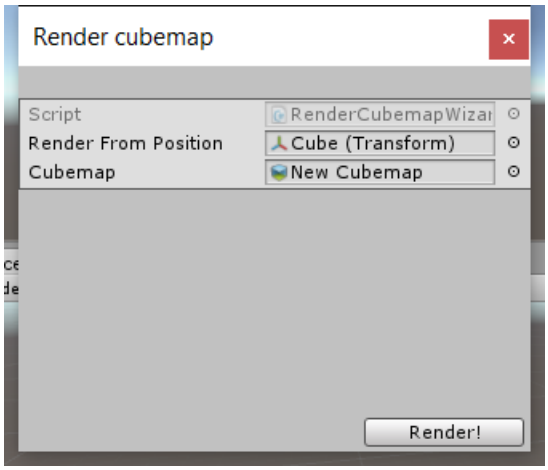
5.2 Käyttötapaus 2: Shader keittiöaltaalle

Kyseinen shader oli käytännössä samantyyppinen kuin aiemmassa käyttötapauksessa. Luotiin PBL-shader, jolle lisättiin perustekstuurin määrittävä arvo. Texture2D-property liitettiin Base Coloriin pääkytkentäalustalla. Muita Slider-propertyjä liittämällä käytettyjä arvoja olivat mm. Metallic ja Gloss. Haluttaessa käyttää Metallic-valintaa shaderin Lighting asetuksista tuli valita Specular Modeksi Metallic.

Poikkeuksena tämä shader vaati ympäristöä peilaavaa ominaisuutta aivan, kuten oikeassakin maailmassa keittiöallas peilaisi. Tämän toteuttamista varten tuli luoda cubemap. Cubemapin luominen tapahtui valitsemalla Unityssä: *Assets -> Create -> Legacy -> Cubemap*. Luodulle cubemapille voitiin antaa nimi ja vaihtaa tiettyjä arvoja, joista tärkeimpänä oli laatu eli Face size. Asetettavia resoluution arvoja olivat mm. 64, 128, 256, 512, 1024. Isompi arvo

tarkoitti parempaa laatua, mutta myös suurempaa tiedostokokoa luodulle cubemapille.

Lisäksi tarvittiin apuvälineeksi RenderCubemapWizard.cs-skripti (liite 1), jonka avulla voitiin luoda cubemapin tarvitsemat kuvat. Skripti löytyi valmiina Unityn omilta ohjesivuilta. Se tuli lisätä Unity projektin Assets-kansioon toimiakseen oikein. Tällöin valitsemalla Unityssä ylävalikosta: *GameObject -> Render into Cubemap* aukesi valintaikkuna, johon voitiin asettaa haluttu cubemap. Render From Position -kenttään lisättiin haluttu peliobjekti, jonka katselukulmasta jokaiseen ympäröivään suuntaan tarvittavat kuvat otettiin. (Kuva 14.)

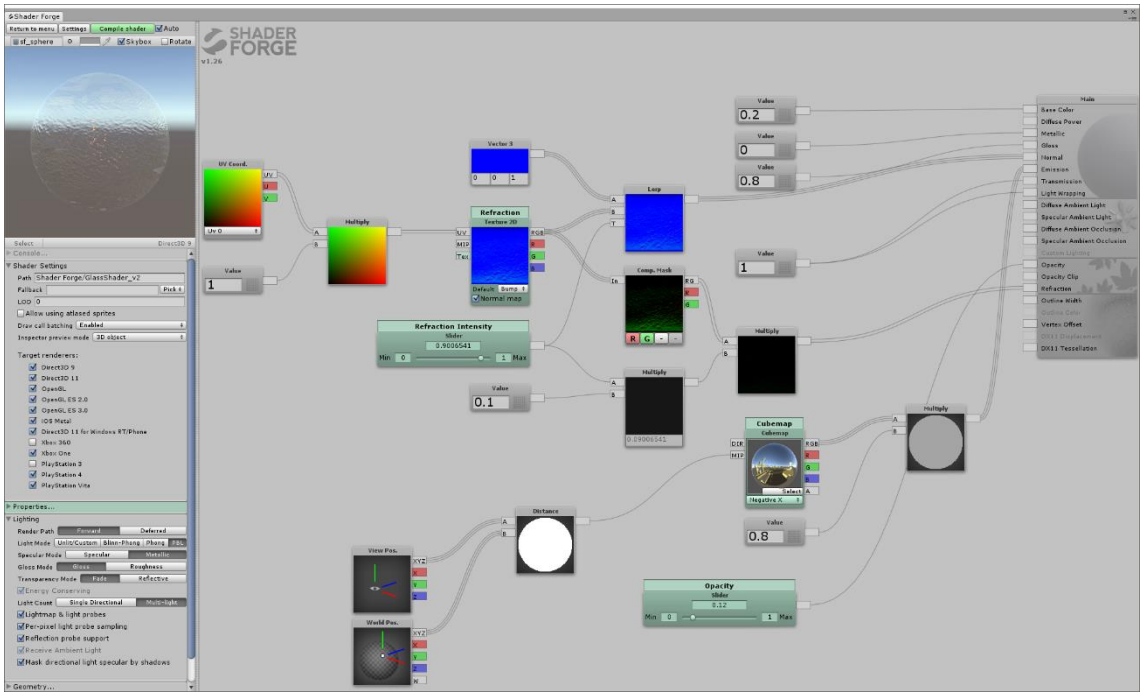


KUVA 14. Kuvakaappaus cubemapin kuvien luomisesta

Kun cubemap ja haluttu peliobjekti oli lisätty skriptin valintaikkunaan, voitiin painaa ”Render!”. Tällöin kuvat rakentuivat automaattisesti ja ne lisättiin cubemapille. Luotu cubemap voitiin lopulta lisätä Shader Forgeen luomalla cubemap-property, johon kyseinen cubemap asetettiin Select-valikosta. Property kytkettiin Multiply-nodeen, johon voitiin myös kytkeä Value-node tai vaikkapa Slider-Property helppokäyttöisen arvon säätämisen avuksi. Kolmen noden muodostama kokonaisuus liitettiin pääkytkentäalustan Specular Ambient Light -liitäntään, jolloin Valuen tai Sliderin avulla voitiin vaikuttaa cubemapin kirkkauteen. Lopputuloksena saatiin hienosti kiiltävä sekä peilaava pinta keittiön altaalle.

5.3 Käyttötapaus 3: Shader lasipinnalle

Perustui myös PBL-valonasetukseen ja suurimmalta osin Shader Forgen Refraction shaderiin, joka oli yksi mukana tulevista kymmenestä esimerkki shaderista. Niitä sai käyttää vapaasti ja muokata tarvittaessa lisää. PBL:ää käytettäessä oli aina kytkettävä joko Metallic- tai Specular-porttiin jokin arvo. Tässä tapauksessa valmiiksi kytkettynä oli lukuarvo 0 ja Metallic. Arvoa suurentamalla saatiin metallimaisempaa pintaa. Base Color-inputtiin oli kytkettynä arvo 0.2, joka vaikutti shaderin kirkkautena (suurempi arvo kirkkaampi). Gloss-porttiin oli kytketty arvo 0.8, mikä aikaansai auringon suuremman heijastuvuuden materiaalin pinnasta. Pääkytkentäalustan Normal-portista Refraction-porttiin päättyvä haara piti sisällään samat piirteet kuin alkuperäisessä Refraction-shaderissa. Kytkennässä oli mm. Texture2D-property yhdistettynä Vector3-nodeen, joka antoi sinisen värin. Texture2D piti sisällään normalmapin. Edellä mainitut nodet olivat yhdistettynä UV-koordinaattitiedot sisällään pitäneen noden ohella Component Mask-nodeen, joka erittelee halutut väriarvot tekstuurista. Lisäksi Slider-property oli liitettynä. Lopulta päädyttiin Refraction-inputiin pääkytkentäalustalta. Sliderin avulla voitiin muuttaa shaderin sisältämän normalmapin kuvion näkyvyyttä. Normalmapiksi täytyi asettaa jokin kuvion sisältämä tekstuuri ja asettaa se Unityn inspector-näkymässä normalmapiksi, jotta shader toimi oikein. Kaiken muun lisäksi kytkettiin Slider-property Opacity-porttiin ja nimettiin se Opacityksi. Tällä tavoin voitiin hallita objektin läpinäkyvyyttä. (Kuva 15.)



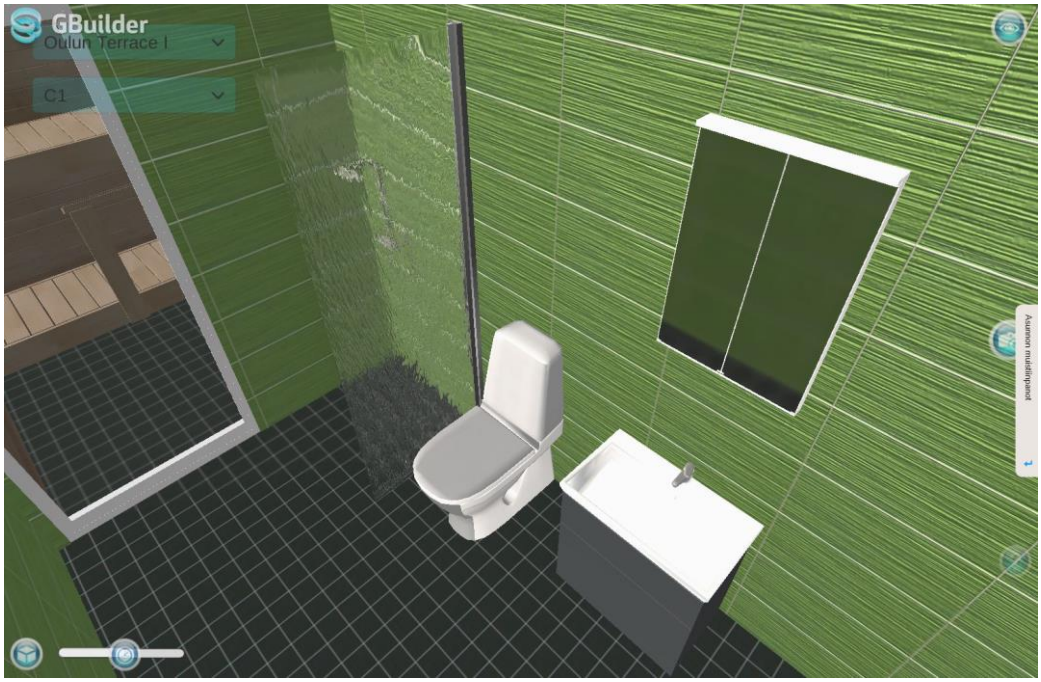
KUVA 15. Kuvakaappaus shaderista lasipinnalle

Kytkentään päätettiin vielä liittää alkuperäisien ominaisuuksien lisäksi cubemap samalla tavoin kuin käyttötapaus 2:ssa. Sen avulla haluttiin lasille sille ominainen ympäristöä heijastava ominaisuus.

Cubemapin käytössä ongelmalliseksi muodostui sen heijastaman kuvan näkyminen juuri sen kokoisena, kuin se objektin näkökulmasta on. Toisin sanoen, jos shaderia käytettiin vaikkapa asuntoon tulevassa peilissä tai ikkunoissa, heijasti se kuvan yhtä suurena kaukaa sekä läheltä katsottuna. Tälle ratkaisuksi tai eräänlaiseksi ohitukseksi huomasin lisätä cubemap-noden MIP-porttiin Distance-noden, jonka A- ja B-portteihin oli kytkettynä View Position- ja World Position -nodet niiden XYZ-outputeista (kuva 15). MIP-portti liittyy Mipmappeihin eli alkuperäisen kuvan esilaskettuihin pienempiresoluutioisiin versioihin (24). Mipmapia luotaessa ensimmäisenä kuvana on korkeimmalla resoluutiolla oleva. Se on havaittavissa lähimmässä pisteessä objektia katsottaessa (24). Cubemap kytkettynä etäisyyden määrittävään nodeen, jolle on kerrottu katselukulma ja objektin meshin sijainti pelimaailmassa mahdollisti ominaisuuden, jossa cubemapin tarkkuus pieneni kauempaa katsottuna.

Luodun cubemapin asetuksissa täytyi olla myös valittuna MipMaps, jotta kytkentä toimi shaderissa. Käytännössä ominaisuus oli eräänlainen kuvan blur-efekti. Lisäksi se toimi teho-optimointina shaderille sumentaessaan kuvan tietyltä etäisyydeltä, koska katsoja ei joka tapauksessa sitä olisi nähnyt kaukaa tarkasti.

Rakennettu shader oli monikäyttöinen ja sitä pystyi pienillä muokkauksilla käyttämään esimerkiksi ikkunassa, peilissä, suihkun väliseinän tai saunanoven lasissa. Luotua Refraction-Intensityksi nimettyä Sliderin arvoa muuttamalla lasille saatiin rypyläinen pinta, joka toimi hienosti vaikkapa suihkun väliseinänä. (Kuva 16.)



KUVA 16. Kuvakaappaus GBuilder-ohjelmistosta

Yhteenvetona voitaisiin sanoa, että opinnäytetyössä käytetyllä työkalulla saatiin hienosti toimiva kokonaisuus yrityksen ohjelmiston 3D-osuudelle. Shaderit toivat tyylikkyyttä lukuisille eri pinnoille. Yrityksen tulevaisuutta ajatellen virtuaalitodellisuus yleistyy ja shaderit tulevat entistä enemmän tarpeellisiksi käyttäjän voidessa kävellä virtuaalilasien avulla tulevassa asunnossaan ja tutkia paikkoja aivan kuten oikeassa maailmassa. Ominaisuus on jo nyt käytössä.

6 POHDINTA

Työn päätarkoituksena oli saavuttaa 3D-mallinnetuille taloille visuaalisesti mahdollisimman näyttävä ulkoasu Unity3D-pelimoottorin ja siihen saatavilla olleen maksullisen lisäosan, Shader Forgen avulla. Työssä rakennettiin erilaisia shadereita, joiden avulla pääasiassa parannettiin 3D-objektien kiiltävyyttä ja heijastusominaisuuksia.

Työn eri vaiheissa törmättiin moniin ongelmiin, joista suurimpana oli luotujen shaderien siirtäminen varsinaiseen yrityksen projektiin. Shaderit näkyivät jostain syystä virheellisesti täysin mustina ja eivät ikään kuin ottaneet valoa vastaan ollenkaan. Lopulta tämä ongelma kuitenkin saatiin ratkaistua tiimin eri jäsenten yhteistyöllä. Lisäongelmia aiheutti myös Shader Forgen ohjemateriaalien ja opastusvideoiden vähäinen määrä tuotteen ollessa vasta kehitysvaiheessa.

Shaderien yhteyteen usein liitetyt cubemapit havaittiin jossain vaiheessa kohtuullisen suuriksi tilanviejiksi. Tiedostokoko saattoi olla jopa 32 MB yhdeltä cubemapilta. Projektin sisältäessä paljon objekteja, joille cubemap luotiin, ohjelmakoodin käännösvaiheessa laatua päätettiin pienentää. Tällöin päästiin huomattavasti alhaisempaan tiedosto- ja projektikokoon. Jatkossa moni shader olisi luultavasti optimoitavissa entistä kevyemmäksi varsinkin mobiilikäyttöä ajatellen.

Lopputuloksena saatiin näyttävä ja hienosti toimiva kokonaisuus, kuten tavoitteenakin oli. Shaderit antoivat asuntojen 3D-malleille ja objekteille viimeisen silauksen. Työn tilaaja Group Builder Oy oli tyytyväinen työn laatuun.

LÄHTEET

1. Company Facts. 2015. Unity Technologies. Saatavissa: http://unity3d.com/public-relations. Hakupäivä 7.11.2015.

2. Unity (pelimoottori). 2015. Wikipedia. Saatavissa: https://fi.wikipedia.org/wiki/Unity_(pelimoottori). Hakupäivä 7.11.2015.

3. Freeman, Will. 2010. Unity launches Unity Asset Store. Develop. Saatavissa: http://www.develop-online.net/news/unity-launches-unity-asset-store/0108154. Hakupäivä 7.11.2015.

4. Frequently asked questions. 2015. Unity Asset Store. Saatavissa: https://unity3d.com/asset-store/sell-assets/faq. Hakupäivä 7.11.2015.

5. Holmér, Joachim. 2015. Shader Forge. Saatavissa: http://acegikmo.com/shaderforge/. Hakupäivä 8.11.2015.

6. Working with Shaders. 2016. Microsoft. Saatavissa: https://msdn.microsoft.com/en-us/library/hh873117.aspx. Hakupäivä 23.11.2015.

7. Fixed Function Pipeline. 2015. OpenGL. Saatavissa: https://www.opengl.org/wiki/Fixed_Function_Pipeline. Hakupäivä 26.11.2015.

8. Birn, Jeremy. 2002. 3D Rendering. Saatavissa: http://www.3drender.com/glossary/3drendering.htm. Hakupäivä 6.12.2015.

9. St-Laurent, Sebastien. 2005. The COMPLETE Effect and HLSL Guide, s. 215. Saatavissa: https://books.google.fi/books?id=5FAWBK9g-wAC&pg=PA215&lpg=PA215&dq=hlsl+reference&source=bl&ots=UXRh3xvsnr&sig=FjIE53N64gvscVbit7qImC6OYIw&hl=fi&sa=X&sqi=2&ved=0ahUKEwi9h_DO68zLAhXBNJoKHVfVC8AQ6AEITzAI#v=onepage&q=hlsl%20reference&f=false. Hakupäivä 8.1.2016.

10. Reference for HLSL. 2016. Microsoft. Saatavissa: https://msdn.microsoft.com/en-us/library/windows/desktop/bb509638(v=vs.85).aspx. Hakupäivä 10.1.2016.

11. HLSL. 2016. Microsoft. Saatavissa: https://msdn.microsoft.com/en-us/library/windows/desktop/bb509561(v=vs.85).aspx. Hakupäivä 10.1.2016.

12. Assembly (ohjelmointikieli). 2015. Wikipedia. Saatavissa: https://fi.wikipedia.org/wiki/Assembly_(ohjelmointikieli). Hakupäivä 14.1.2016.

13. Brown, Dan. 2012. What are Vertex and Pixel shaders? Stackoverflow. Saatavissa: http://stackoverflow.com/questions/832545/what-are-vertex-and-pixel-shaders. Hakupäivä 6.2.2016.

14. Holmér, Joachim. 2014. Examples for PBL/Energy Conservation. UserEcho. Saatavissa: https://shaderforge.userecho.com/topics/205-examples-for-pblenergy-conservation/. Hakupäivä 12.3.2016.

15. Acegikmo. 2015. Lightmapping. Saatavissa: http://acegikmo.com/shaderforge/wiki/index.php?title=Lightmapping. Hakupäivä 13.3.2016.

16. Acegikmo. Nodes. Saatavissa: http://acegikmo.com/shaderforge/nodes/. Hakupäivä 2.4.2016.

17. RB Whittaker. Post-Processing Shaders. Saatavissa: http://rbwhitaker.wikidot.com/post-processing-effects. Hakupäivä 3.4.2016.

18. Zucconi, Alan. 2015. Screen shaders and image effects in Unity3D. Saatavissa: http://www.alanzucconi.com/2015/07/08/screen-shaders-and-postprocessing-effects-in-unity3d/. Hakupäivä 3.4.2016.

19. Specular highlight. 2015. Wikipedia. Saatavissa: https://en.wikipedia.org/wiki/Specular_highlight. Hakupäivä 4.4.2016.

20. Russell, Eddie. 2014. Understanding the Difference between Texture Maps. Saatavissa: http://blog.digitaltutors.com/understanding-difference-texture-maps/. Hakupäivä 13.4.2016.

21. Unity. 2016. Shader Level of Detail. Saatavissa: http://docs.unity3d.com/Manual/SL-ShaderLOD.html. Hakupäivä 16.4.2016.

22. Draw Call Batching. 2016. Unity. Saatavissa: http://docs.unity3d.com/Manual/DrawCallBatching.html. Hakupäivä 16.4.2016.

23. Cupisz, Robert. 2011. Advanced shading and lighting for mobile. Unity. Saatavissa: http://blogs.unity3d.com/2011/06/08/advanced-shading-and-lighting-for-mobile/. Hakupäivä 17.4.2016.

24. Microsoft. 2016. Texture Filtering with Mipmaps (Direct3D 9). Saatavissa: https://msdn.microsoft.com/en-us/library/windows/desktop/bb206251(v=vs.85).aspx. Hakupäivä 20.4.2016.

25. Unity. 2016. ShaderLab Syntax. Saatavissa: http://docs.unity3d.com/Manual/SL-Shader.html. Hakupäivä 2.5.2016.

26. Unity. 2016. Shaders: Vertex and Fragment Programs. Saatavissa: http://docs.unity3d.com/Manual/ShaderTut2.html. Hakupäivä 2.5.2016.

27. Nitschke, Benjamin. 2007. Professional XNA Game Programming: For Xbox 360 and Windows. Saatavissa: https://books.google.fi/books?id=YsNptpNlfCUC&pg=PA187&dq=shader+normal+space&hl=fi&sa=X&ved=0ahUKEwi29ePdu8DMAhXLDZoKHY4KD3wQ6AEIIzAB#v=onepage&q=shader%20normal%20space&f=false. Hakupäivä 4.5.2016.

28. Acegikmo. 2014. Using Shader Forge for Mobile. Saatavissa: http://acegikmo.com/shaderforge/wiki/index.php?title=Using_Shader_Forge_for_Mobile. Hakupäivä 4.5.2016.

```
using UnityEngine;
using UnityEditor;
using System.Collections;

public class RenderCubemapWizard : ScriptableWizard {

        public Transform renderFromPosition;
        public Cubemap cubemap;

        void OnWizardUpdate () {
                string helpString = "Select transform to render from and cubemap to render into";
                bool isValid = (renderFromPosition != null) && (cubemap != null);
        }

        void OnWizardCreate () {
                // create temporary camera for rendering
                GameObject go = new GameObject( "CubemapCamera");
                go.AddComponent<Camera>();
                // place it on the object
                go.transform.position = renderFromPosition.position;
                go.transform.rotation = Quaternion.identity;
                // render into cubemap
                go.GetComponent<Camera>().RenderToCubemap( cubemap );

                // destroy temporary camera
                DestroyImmediate( go );
        }

        [MenuItem("GameObject/Render into Cubemap")]
        static void RenderCubemap () {
                ScriptableWizard.DisplayWizard<RenderCubemapWizard>(
                        "Render cubemap", "Render!");
        }
}
```