Katja Kaikkonen

Volumetriset efektit ja käyttö Unreal Enginessä

Tradenomi

Tietojenkäsittely

Syksy 2021

KAMK • University of Applied Sciences

KAMK • University of Applied Sciences

Tiivistelmä

Tekijä(t): Kaikkonen Katja

Työn nimi: Volumetriset efektit ja käyttö Unreal Enginessä

Tutkintonimike: Tradenomi

Asiasanat: volumetrinen, efekti, Unreal Engine 4, optimointi, volyymi, tilavuus, sumu, valo

Opinnäytetyö käsittelee volumetristen efektien teoriaa ja implementointia Unreal Engine 4:ssä. Tarkoituksena oli oppia volumetristen efektien tekniikoita, jotta pystytään soveltamaan näitä efektejä käytännössä ja tarvittaessa ratkomaan mahdollisia ongelmatilanteita.

Volumetriset efektit ovat valon havaitsemista sen kulkiessa kolmiulotteisen väliaineen läpi. Volumetrisiä efektejä on hyödynnetty elokuvataiteessa paljon ennen 3D-grafiikan yleistymistä. Volumetristen efektien pohjalta kehitettiin algoritmi lääketieteen tarkoituksiin, jossa se on edelleen käytössä magneetti- ja tietokonekerroskuvantamisessa. Volumetristen efektien tekemiseen on useita erilaisia tekniikoita, ja ne voidaan jakaa kahteen pääkategoriaan: suoraan ja epäsuoraan. Tekniikat eroavat toisistaan datan käsittelyn tavassa: epäsuorat tekniikat luovat datasta polygoniverkon, kun taas suorat tekniikat eivät muunna dataa toiseen muotoon, vaan käyttävät esimerkiksi 3D-tekstuureita. Epäsuorat tekniikat ovat paljon kevyempiä käyttää ja ne on helppo implementoida, mutta ne eivät ole yhtä hyvälaatuisia kuin suorat tekniikat. Suorat tekniikat ovat taas raskaampia, ja niiden implementointi voi olla hankalampaa, koska kaikki järjestelmät eivät tue 3D-tekstuureita. Suorien tekniikoiden laatu on paljon parempi kuin epäsuorissa tekniikoissa.

Unreal Engine 4 käyttää suoria tekniikoita volumetristen efektien toteuttamiseen. Monet komponentit pohjautuvat fysiikan lakien imitointiin, ja ymmärtämällä valon käyttäytymistä komponentteja pystytään hallitsemaan tehokkaammin. Käytännön osuudessa rakennettiin volumetriset efektit pelimoottorin sisällä ja tarkasteltiin ratkaisuja, joilla saadaan efektit toimimaan halutulla tavalla. Osuudessa otettiin huomioon keinot, joita voidaan käyttää optimointiin, ja ratkaistaan ongelma resoluution ja näköetäisyyden välillä. Lisäksi tutustuttiin, miten paikallinen modulaarinen efekti luodaan materiaalin avulla ja miten sitä hyödynnetään erilaisissa ympäristöissä. Volumetrinen efekti toimii niin sumuna kuin hiekka- ja lumimyrskynä. Opinnäytetyö tehtiin toimeksiantona Kajaanin Ammattikorkeakoulun yhteydessä toimivalle Clever Simulation Entertainment -kehitystiimille, jossa pyrittiin hyödyntämään Unreal Enginen volumetrisiä efektejä varsinkin virtuaalitodellisuuden parissa.

Opinnäytetyössä havaittiin, että on mahdollista luoda yhä näyttävämpiä volumetrisiä efektejä, kun ymmärtää, miten eri tekniikat ja valon käyttäytyminen vaikuttavat lopputulokseen. Teknisten rajoitteiden tiedostaminen ei suinkaan rajoita tekemistä, vaan antaa uusia keinoja hyödyntää ja implementoida efektejä erilaisille digitaalisille alustoille.

KAMK • University of Applied Sciences

Abstract

Author(s): Kaikkonen Katja

Title of the Publication: Volumetric Effects and Use in Unreal Engine 4

Degree Title: Bachelor's Degree in Business Information Technology, Bachelor of Business administration.

Keywords: volumetric, effect, Unreal Engine 4, optimization, volume, fog, light

This Bachelor's thesis studies the theory and implementation of volumetric effects in Unreal Engine 4. The purpose was to learn techniques of the volumetric effects in order to apply them in practice and if necessary, be able to solve potential problem situations.

Volumetric effects have been utilized in cinematography long before 3D graphics became common, and an algorithm was developed for medical purposes where it is still in use in the MRI (Magnetic Resonance Imaging) and CT (Computed tomography) scanning. There are several different techniques for creating volumetric effects and they can be divided into two main categories: Direct and indirect techniques. The techniques differ in the way data is handled: The indirect techniques create a polygon mesh from data, while the direct techniques do not convert the data into another format, but use, for example, 3D textures. The indirect techniques are much faster and easier to implement, but not as visually good looking as, the direct techniques. The direct techniques are more expensive and can be more difficult to implement because not all systems support 3D textures. However, the direct techniques have become standard method nowadays because the quality is more visually attractive than what you can achieve with the indirect techniques.

Unreal Engine 4 uses direct techniques to implement volumetric effects. Many components are based on the laws of physics, and understanding how light behaves, it is possible to have more effective control. In the practical part, the volumetric effects are built inside the game engine and solutions are considered that provide an efficient way for the effects to work as desired. Same effects do not necessarily work in different situations, and that is why it is good to know in advance for what purpose the effect is created for. The practical part considers means that can be used for optimization and solves a problem between resolution and view distance. In addition, it was explored how the local modular effect is created with material and how it is utilized in different environments. The volumetric effect acts as a mist as well as a sand and snowstorm. The thesis was commissioned by the Clever Simulation Entertainment development team at Kajaani University of Applied Sciences, where the aim was to utilize Unreal Engine's volumetric effects, especially in virtual reality.

The following observation was considered as the most important point of the thesis: It is possible to create increasingly spectacular volumetric effects when one understands how different techniques and the behavior of light affects the result. Awareness of the technical restrictions by no means limits the work progress but provides new ways to utilize and implement the effects.

Sisällys

Termiluettelo

Artefakti

Käytetystä tekniikasta johtuva säännöllinen virhe tietotekniikassa.

Assetti

Pelimoottorin sisäisiä käyttövalmiita sisältöpaketteja.

Etupertoinen renderöinti

Volumetristen efektien objektinjärjestysalgoritmeihin perustuva renderöintitekniikka (Forward rendering). Ei vakiintunutta suomenkielistä termiä.

Frokseli

Kameran suuntainen vokseli.

Fysiikkaperusteinen renderöinti

Physical based rendering, PBR. Perustuu kuvien luomiseen todellisen valon käyttäytymisen perusteella.

Implementointi

Pelimoottorin sisältöpaketin käyttöönotto.

Noodi

Unreal Enginen visuaalisessa ohjelmoinnissa käytettävä graafin vaihe.

Pikseli

Kuvapiste, bittigrafiikassa kuvan pienin osa-alue.

Polygoniverkko

Polygonikappaleista koostuva malli (Mesh).

Polunseuranta

Renderöintitekniikka (Path tracing), jatkettu versio säteenseurannasta. Hyvin tarkka jäljitystekniikka, joka seuraa luotua sädettä kohteiden välillä.

Rasterointi

Tapa esittää kuvia digitaalisessa muodossa.

Renderöinti/3D-piirto

Digitaalisen tiedon muuntaminen näytölle sopivaan esitysmuotoon.

Säteen marssitus

Säteen suuntauksen kaltainen renderöintitekniikka (Ray marching), jossa sädettä marssitetaan pisteiden välillä. Ei vakiintunutta suomenkielistä termiä.

Säteenseuranta/säteenjäljitys

3D-grafiikassa käytettävä renderöintitekniikka, joka jäljittelee virtuaalisten valonsäteiden kulkua (Ray tracing).

Säteen suuntaus

3D-grafiikassa käytettävä renderöintimenetelmä (Ray casting).

Takaperoinen renderöinti

Volumetristen efektien renderöintitekniikka (Backward rendering), joka pohjautuu kuvatila- tai kuvajärjestysalgoritmeihin. Ei vakiintunutta suomenkielistä termiä.

Vokseli

Vokseli edustaa arvoa ruudukossa 3D-avaruudessa.

Volyymi

Tilavuus 3D-grafiikassa.

Volyymitekstuuri

Unreal Enginen käyttämä komponentti ”Volume texture”.

Väliaikainen uudelleenheijastaminen

Unreal Enginen tapa käyttää reunojen pehmennystä (anti-aliasing), jonka tarkoituksena on hävittää grafiikassa ilmeneviä artefakteja. Englanninkieliseltä termiltään myös ”Temporal reprojection” ja ”Temporal anti-aliasing”.

1 Johdanto

Volumetriset efektit ovat valon havaitsemista kolmiulotteisen väliaineen läpi. Niillä on jo kahden vuosikymmenen ajan luotu tunnelmaa, immersiota ja syvyyttä peleihin. Viime vuosina efektit ovat alkaneet olla yhä näyttävämpiä tekniikan ja suorituskyvyn parantuessa. Volumetriset efektit ovat kuitenkin edelleen raskaita ja resursseja vieviä, ja niiden implementointi vaatii tietoa efektien toiminnasta ja optimoinnista.

Oikeassa elämässä näemme volumetrisiä efektejä päivittäin. Aamu-usva, valonsäteet ikkunaverhon välistä tai jopa höyry aamukahvista ovat volumetrisiä efektejä. Valon hajonnan puuttuminen herättää huomiota ja saa esimerkiksi pelin pahimmillaan näyttämään viimeistelemättömältä ja epäaidolta, mikäli tavoitellaan realistisuutta. Sumuefektejä on toki tehty myös muilla tekniikoilla kuin tilavuutta mallintamalla, mutta tässä työssä perehdymme volumetrisiin tekniikoihin.

Opinnäytetyö on tehty toimeksiantona Kajaanin Ammattikorkeakoulun yhteydessä toimivalle Clever Simulation Entertainmentille. Clever SE tuottaa peli- ja virtuaalitodellisuusratkaisuja lähinnä asiakaslähtöisesti, mutta myös Kajaanin Ammattikorkeakoulun erinäisiin hankkeisiin. Tarkoituksena oli avata volumetristen tekniikoiden taustoja ja rakentaa valmis modulaarinen efekti, jota voidaan soveltaa eri ympäristöissä mahdollisimman kustannustehokkaasti.

Opinnäytetyö jakaantuu teoria- ja käytännönosuuteen. Teoriaosuudessa avataan, mistä volumetriset efektit muodostuvat ja miten valon fysiikka voi vaikuttaa efektin toimintaan. Lisäksi käydään läpi tekniikoita, miten efektejä on mahdollista toteuttaa, ja eroja tapojen välillä. Teoriaosuuden toinen vaihe käsittelee efektien toimintaa Unreal Engine 4.26:ssa ja mitä teknisiä toteutustapoja pelimoottori käyttää. Lisäksi sivutaan efektien toimivuutta virtuaalitodellisuudessa. Opinnäytetyön käytännön osuudessa hyödynnetään teoriaosuudessa opittuja tietoja ja rakennetaan pelimoottorin sisällä volumetriset efektit ja tutkitaan, mitä eri komponentteja on käytettävissä. Lopuksi rakennetaan modulaarinen volumetrinen efekti Clever SE:n kehitystiimin käyttöön ja tutkitaan, miten se käyttäytyy eri ympäristöissä. Käytännön osuudessa ratkotaan erilaisia teknisiä ongelmatilanteita.

Opinnäytetyön päätarkoituksena on antaa lukijalle kokonaisvaltainen kuva volumetrisistä efekteistä ja rakentaa valmis paikallinen efekti helposti implementoitavaksi eri tilanteisiin. Volumetristen efektien aihealue on laaja, ja työ raapaisee vain osittain tekniikkojen ja komponenttien käyttöä eri tarkoituksissa.

Vaikka työssä perehdytään hieman matemaattisiin kaavoihin, pyritään kuvien ja visualisoinnin kautta avaamaan toimintaa jokaiselle. Tämän opinnäytetyön lopussa pitäisi kyetä luomaan Unreal Enginessä volumetriset efektit siten, että pystyy ottamaan huomioon niiden vaikutuksen suorituskykyyn ja ratkomaan yleisimpiä ongelmia volumetrisiä efektejä tehdessä. Opinnäytetyö keskittyy yhteen peligrafiikan osa-alueeseen ja menee syvemmälle tekniikoihin. Siksi lukija tarvitsee perustietämyksen 3D-grafiikasta ja Unreal Enginen käytöstä ennen työhön perehtymistä.

2 Volumetriset efektit

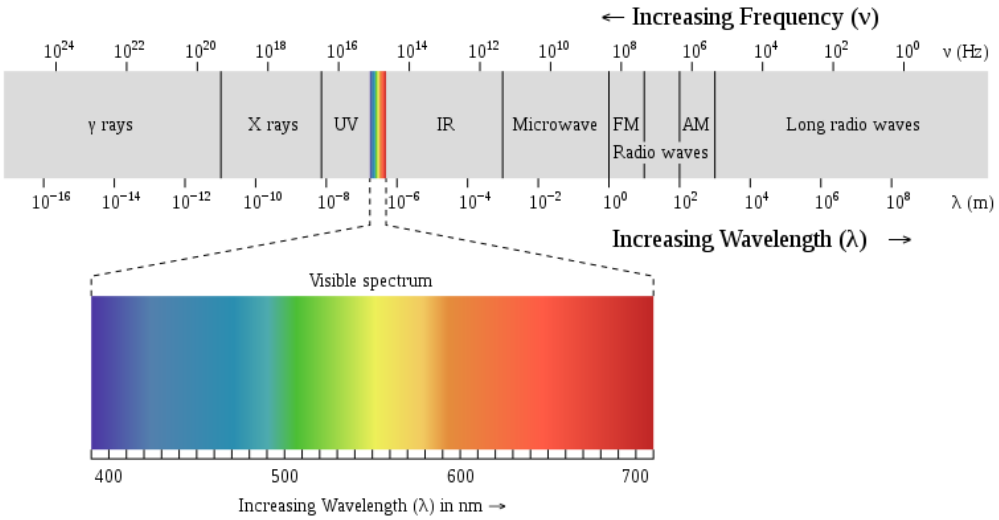
2.1 Volumetristen efektien konsepti

Volumetrisen efektin voi yksinkertaisimmillaan kuvailla näin: Valo on nähtävissä kulkiessaan todellisen kolmiulotteisen väliaineen (kuten sumu, pöly, savu tai höyry) tilavuuden läpi, aivan kuten todellisessa maailmassa [1].

Perimmäisin idea tilavuuden mallintamisen taustalla on arvioida valonsäteet, kun ne kulkevat tilavuuden läpi ja antaa jokaiselle leikkaavalle pikselille opasiteetti ja väri. Toisin sanoen tietokonemallintamisessa pyritään simuloimaan valon käyttäytymistä tilavuudessa (kuva 1). Valon käyttäytymisen mallintaminen ei aina ole helppoa primitiivisillä muodoilla, kuten tullaan myöhemmin toteamaan [2, 3].

Kuva 1. Volyymi jäljittelee oikeaa sumua, peittäen hahmoa enemmän, mitä syvemmällä se on volyymin sisällä, koska valo hajaantuu yhä vahvemmin.

Koska volumetriset efektit ovat pohjimmiltaan valon käyttäytymistä eri aineissa, täytyy ymmärtää valon fysiikkaa. Valo on sähkömagneettista säteilyä, ja silmä havaitsee aallonpituudet eri väreinä, kuten kuvassa 2 on esitetty. Mitä lyhyempiä sähkömagneettiset aallot ovat, sitä sinisempää valo on. Jos kaikkia aallonpituuksia on saman verran, valo on valkoista.



Kuva 2. Sähkömagneettinen spektri, jossa näkyvä osa korostettuna. [4.]

Valon hajoaminen voidaan jakaa karkeasti kahteen elastiseen tyyppiin: Rayleigh-sirontaan ja Mie-sirontaan. Sironta on fysikaalinen prosessi, jossa valo muuttaa suuntaansa kohdatessaan esteen tai tiheyden muutoksen, kuten esimerkiksi sumun. Rayleigh-sironnassa hiukkasten koko on huomattavasti aallonpituutta pienempi ja sironta on verrannollinen aallonpituuteen, aiheuttaen erilaisia väriefektejä. Mie-sironta on taas vastakohta: hiukkaset ovat aallonpituutta vastaavia tai suurempia, ja sironta on suurempaa kuin Rayleigh-sironnassa. Koska Mie-sironta on riippumaton aallonpituudesta ja tapahtuu valon etenemissuunnassa, valo näyttää lähes valkoiselta. [5, 6.]

2.2 Käyttötarkoitukset

Kuten todettiin, valo käyttäytyy tilanteissa eri tavoin. Käyttäytyminen vaikuttaa efektien ulkonäköön ja tekotapaan, joten eri tekniikoiden ymmärtäminen ja käyttötarkoituksen rajaaminen on tärkeää, jotta saavutettaisiin toivottu lopputulos. Volumetrisillä efekteillä on mahdollista mallintaa paikallista sumua, usvaa, pilviä tai valon sirontaa horisontissa. Tietyllä sirontatyypillä voidaan myös mallintaa näkyviä valonsäteitä.

Volumetrinen valaistus on terminä sangen uusi, vaikka efektiä on käytetty taiteessa paljon pidempään. Termi itsessään vakiintui vasta 3D-grafiikan ja tietokonemallintamisen myötä, sillä elokuvataiteessa puhutaan savu- tai utuefekteistä [7, 8]. Volumetrisen renderöinnin konsepti syntyikin LucasFilmillä, jonka jälkeen Pixar kehitti konseptia ja hyödynsi sitä elokuvissaan [9]. Volumetrisiä efektejä käytetään niin animoiduissa kuin näytellyissä elokuvissa, kuten alla olevissa kuvissa 3 ja 4 voi nähdä.

Kuva 3. Mallinnettuja volumetrisiä efektejä elokuvassa Godzilla II: King of Monsters. [10.]

Kuva 4. Savukoneella luotu efekti kauhuelokuvassa Wendigo Carnage. [11.]

Nykyään volumetrisiä efektejä käytetään myös peleissä tuomaan syvyyttä ja tilan tuntua, mutta myös tunnelmaa (kuva 5). Volumetriset efektit ovat nykypäivänä tärkeä keino tuoda immersiivistä pelikokemusta. Ensimmäinen volumetrisen sumun käyttöönotto tosin tapahtui eri tarkoituksella. Vuonna 1999 julkaistu kauhupeli Silent Hill kärsi sen ajan teknisistä rajoitteista, ja piirtoetäisyys oli suuri ongelma. Piilottaakseen objektien yllättävän piirtymisen lähietäisyydellä kerrotaan, että kehittäjät päättivät peittää alueen paksulla sumulla. [12.]

Vaikka nykyajan tekniikat ovat kehittyneet, monien volumetristen mallien toiminta pohjautuu William Lorensenin ja Harvey Clinen kehittämään marssivat kuutiot algoritmiin, joka julkistettiin

vuonna 1987 [13]. Algoritmi oli alun perin tarkoitettu lääketieteen käyttöön, ja nykyäänkin algoritmia ja sen kehittyneempiä versioita hyödynnetään tietokonekerros- ja magneettikuvauksesta saadun datan mallintamisessa [14].

Kuva 5. Red Dead Redemption 2 on tunnettu volumetrisista efekteistään. Vokselipohjaiset ratkaisut ovat näyttäviä [15].

2.3 Tekniikat

Volumetrinen renderöinti on yleinen termi kaikkiin metodeihin, jotka ottavat 3D-dataa ja projektoivat sen 2D-pinnalle [16]. Tilavuuden dataa voidaan kuvata erilaisilla tavoilla, kuten laatikkona, perinteisellä vokselidataruudukolla, 3D-funktiolla (x, y, z) tai esimerkiksi pisteillä tilavuuden sisällä.

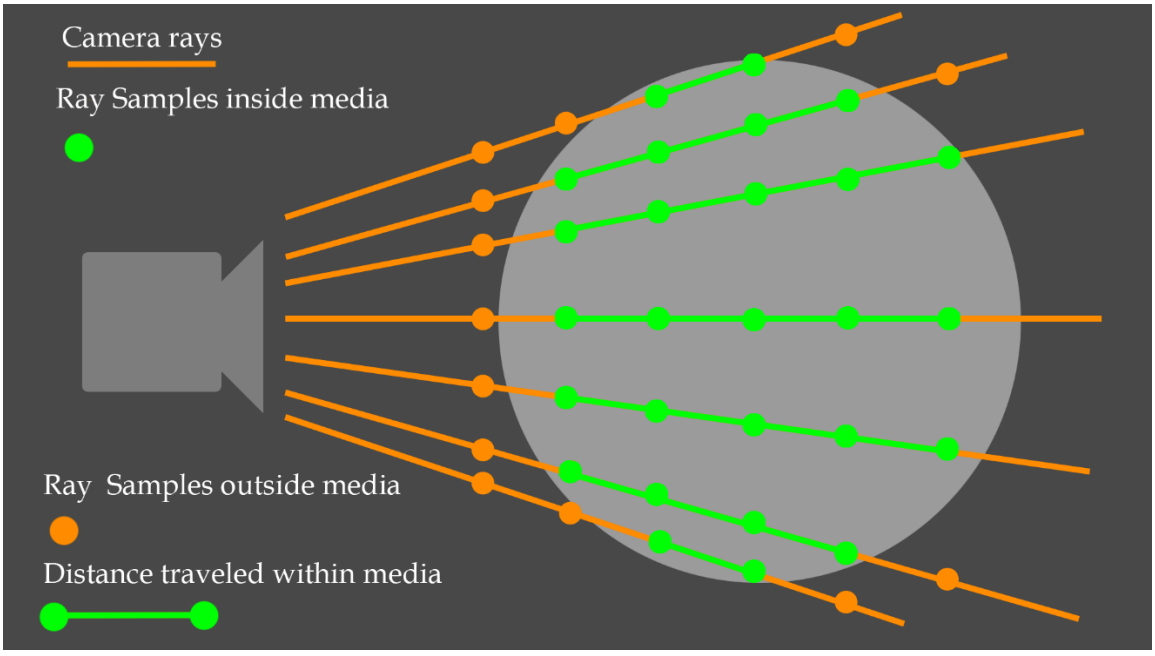
Volumetrisen renderöinnin tekniikat voidaan jakaa kahteen eri kategoriaan: suoriin ja epäsuoriin tekniikoihin, jotka molemmat käyttävät erilaisia algoritmeja valon visualisointiin tilavuudessa [17,18]. Suorat tekniikat hyödyntävät erilaisia säteenseuranta- ja vokselipohjaisia algoritmeja ja muodostavat datan pohjalta 2D-kuvia [19]. Epäsuorat tekniikat taas luovat läpinäkyvän esityksen polygoniverkosta [18].

2.3.1 Suora tilavuuden renderöinti

Suoran tilavuuden renderöinnin ideana on saada 3D-esitys suoraan tilavuuden datasta, kuten nimikin vihjaa. Koska datan katsotaan esittävän puoliläpinäkyvää, valoa lähettävää väliainetta, on mahdollista simuloida kaasumaisia ilmiöitä, kuten esimerkiksi tornadoja tai savua. Suora tilavuuden renderöinti voi käyttää kahdenlaisia tekniikoita: etu- ja takaperoisia. [20.]

Takaperoinen tekniikka käyttää kuvatila- tai kuvajärjestysalgoritmejä. Algoritmit suoritetaan jokaiselle pikselille erikseen, mistä hyvä esimerkki on tilavuuden säteenseuranta ja säteen marssittaminen. Tilavuuden säteenseuranta on jatkettu versio polunseurannasta [21]. Yksinkertaisimmillaan polunseurannassa seurataan luotua sädettä kamerasta pintoihin ja pinnoista valonlähteeseen [22]. Tilavuuden säteenseurannasta seurataan säteen kulkua objektien lävitse ja jokaisella kohtaamisella lasketaan, miten paljon säteestä hajautuu valoa kohtaamisen aikana [21].

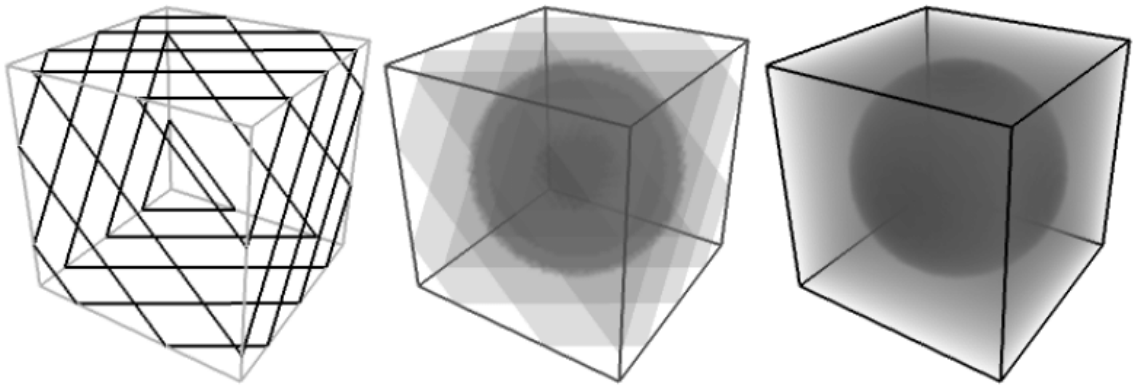
Säteen marssittaminen on myös kuvapohjainen tilavuuden mallintamistekniikka, ja se luo 2D-kuvia laskemalla 3D-dataa. Kuten tilavuuden säteenseuranta säteen marssitus puskee säteet tilavuuden läpi, luoden pisteitä matkalle. Marssituksen aikana lasketaan pisteiden välisiä etäisyyksiä (kuva 6). Säteen marssitus käyttää säteensuuntausta, mutta kuten muutkin volumetriset säteenseurantatekniikat, lasketaan pintadatan sijaan tilavuuksia. Volumetrisissä säteenseurannoissa ei myöskään luoda toissijaisia säteitä, koska tälle ei ole tarvetta. [14, 20, 23.]



Kuva 6. Säteen marssitus tilavuuden läpi.

Suoran tilavuuden renderöinnin heikkoja puolia ovat korkea ruudunpäivityksen kustannus ja yhdistäminen muihin polygoniverkkoihin, koska useat järjestelmät käyttävät kolmiopohjaisia esitystapoja. On kuitenkin olemassa erilaisia keinoja keventää täysiä volumetrisiä efektejä. Esimerkiksi korkeuskartoilla voidaan luoda kuvitteellisia varjostuksia, eivätkä ne ole läheskään niin suorituskyvylle raskas tapa kuin täysin volumetrisesti tehtynä. [18, 24.]

Etuperoinen renderöinti taas käyttää objektijärjestysalgoritmeja. Nämä algoritmit suoritetaan jokaiselle vokselille erikseen, ja esimerkiksi tekstuurin viipalointi käyttää tätä tekniikkaa. Tekstuurin viipalointi ottaa 3D-datan tilavuudesta, jakaa sen 2D-muotoon ruudukon avulla ja asettaa sen kameran suuntaisesti. Tekstuurin viipaloinnin hyviä puolia on, että se toimii vanhemmissa järjestelmissä, jotka eivät tue 3D-tekstuureita. 2D-tekstuurit ovat hyvin kevyitä ja nopeita, mutta artefaktien mahdollinen ilmaantuminen on todennäköistä tietystä kulmasta katsottuna ja kuvanlaatu ei ole paras mahdollinen. 3D-tekstuurien hyviä puolia on taas kuvanlaatu, mutta tekstuurit ovat hitaampia ja raskaampia kuin 2D:ssä. [20, 25.] Kuva 7 havainnollistaa tekstuurin viipaloinnin periaatetta.



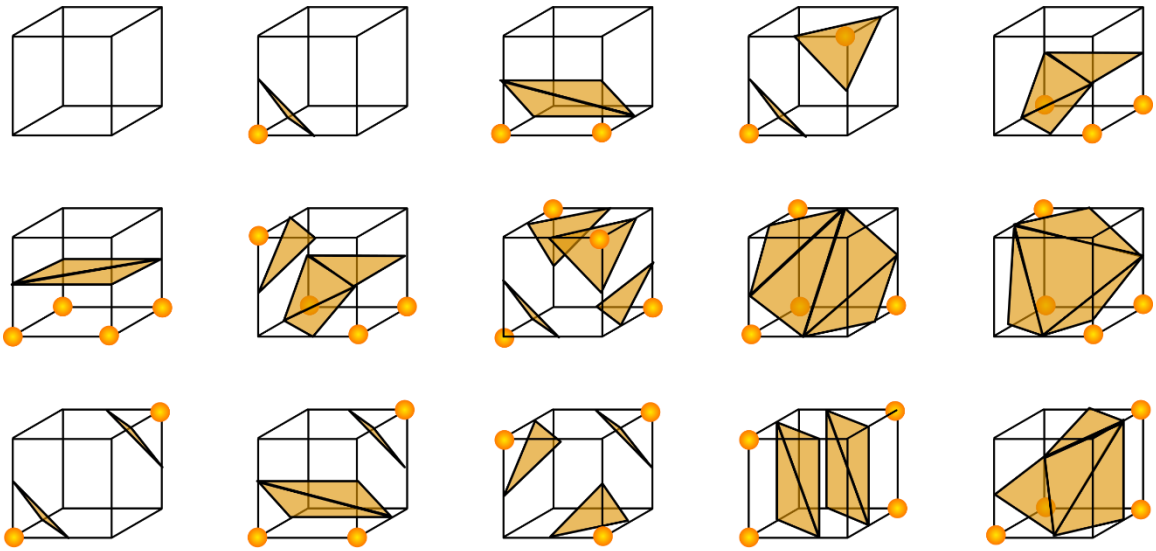
Kuva 7. Tilavuuden renderöinti 3D-tekstuuria viipaloimalla [25].

2.3.2 Epäsuora tilavuuden renderöinti

Epäsuora tilavuuden renderöinti muodostaa tilavuudesta polygoniverkon, ja kolmioiden käyttämisellä on paljon etuja. Grafiikkaprosessorit on suunniteltu toimimaan laskemalla kolmioita, joten siksi epäsuora tilavuuden renderöinti on tehokkaampi tapa kuin suora, jossa käytetään vokseleita. Vokselit ovat pikseleiden 3D-vastine. Polygonisointi tarkoittaa vokselien muuntamista kolmioiksi, ja se tapahtuu vain kerran, minkä jälkeen kolmiot piirretään joka ruudulla. Koska epäsuora tekniikka pohjautuu polygoniverkon esittämiseen, antaa se hyvän kontrollin lopputuloksesta. Tosin kuvanlaatu ei ole läheskään niin hyvä kuin monessa suorassa renderöintitekniikassa. [18, 26.]

Yksinkertaisin ja tunnetuin tapa polygonisoida vokseleita on aikaisemmin mainittu algoritmi marssivat kuutiot. Algoritmi mallintaa tilavuuden sijaintia pisteiden avulla ja luo kolmioita datan pohjalta. Tilavuuden sisällä jokaiselle vokselille luodaan kuvitteelliset kahdeksan pistettä reunoille ja algoritmi marssittaa verteksit sopiviin asemiin ja väliin luodaan pintoja. [13, 18, 27, 28.]

Tämä tapahtuu käytännössä siten, että luodaan hakemisto esilasketuista mahdollisista polygonin konfiguraatioista, joita on kuution sisällä yhteensä 256 (2^8=256). Näin suuri määrä erilaisia konfiguraatioita on kuitenkin altis virheille. Koska monet tuloksista vastaavat toisiaan, voidaan ne tiivistää 15 uniikkiin tapaukseen, jotka ovat näkyvissä alla (kuva 8). [13]



Kuva 8. Marssivat kuutiot -algoritmin 15 uniikkia tapausta. [29.]

Huonona puolena on, että algoritmi voi luoda paljon pieniä kolmioita, eikä välttämättä luo niin sanottua vedenkestävää polygoniverkkoa, vaan jättää reikiä. Puute on korjattu edistyneemmissä algoritmeissa. Marssivien kuutioiden algoritmissa täytyy muistaa, että generoitavalla objektilla on oltava tasainen pinta. Jos kuutiossa on teräviä kulmia, algoritmi tasoittaa ne automaattisesti ja toimii siksi esimerkiksi maan polygonisoinnissa hyvin. Algoritmia voidaan käyttää myös teräväkulmaisissa objekteissa nostamalla ruudukon resoluutiota, mutta tällä on suora vaikutus suorituskykyyn. Yksilölliset kuutiot mahdollistavat limittäisyyden, koska jokainen toimii itsenäisesti. [18.]

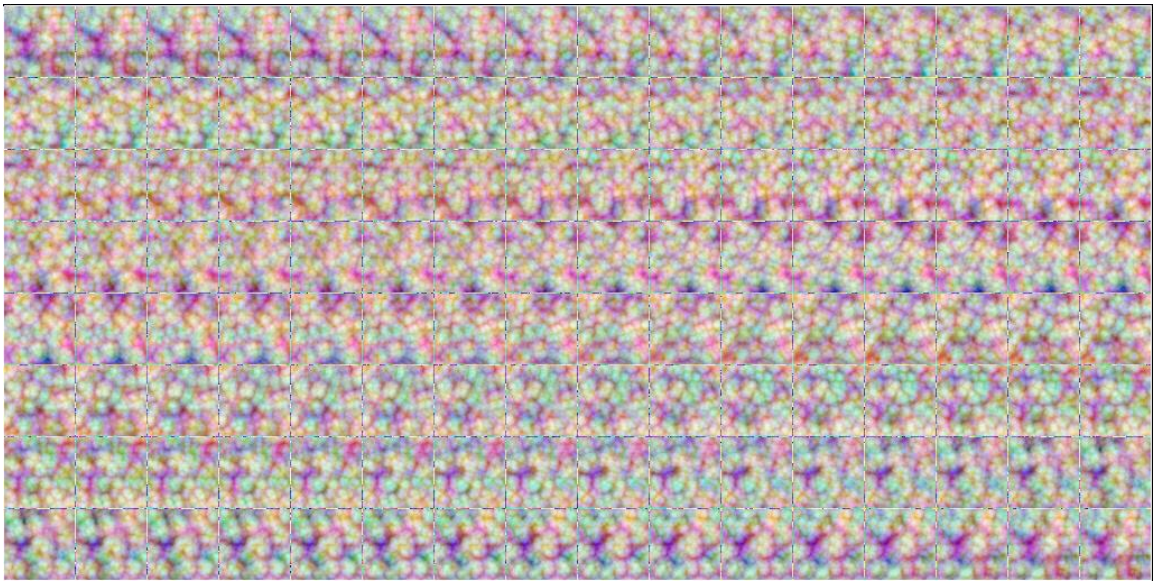
3 Unreal Engine ja efektit

3.1 Unreal Enginen komponentit efektien luomiseen

Unreal Engine sisältää useita eri komponentteja efektien luomiseen. Globaaleja eli ei tarkasti rajattuja sumuja ovat atmosfäärinen sumu ja eksponentiaalinen korkeussumu. Atmosfäärinen sumu jäljittelee partikkeleiden käyttäytymistä ilmakehässä, kun taas eksponentiaalinen korkeussumu mallintaa sumun tiiviyttä eri korkeuksilla. [30, 31]. Tämän lisäksi valon hajoamista voidaan hallita myös taivaan atmosfäärillä. Eksponentiaalisen korkeussumun asetusten kautta päästään myös volumetrisen sumun asetuksiin.

Unreal Enginen uudemmissa versioissa atmosfäärinen sumu on osittain korvautunut Sky Atmosphere -komponentilla, joka toimii suuntavalon lisänä taivaan generoinnissa. Atmosfääristä sumua ei voi käyttää uuden komponentin kanssa yhdessä, sillä taivaan atmosfääri mallintaa sumua itsestään. Komponentti mallintaa valmiiksi Mie-sironnalla korkeussumua, mutta tarvittaessa ympäristöön voidaan lisätä eksponentiaalinen korkeussumu vahvistamaan efektiä. Koska uusi komponentti voi keskustella päävalon ja korkeussumun kanssa, atmosfääri reagoi valon sijaintiin. Tämä tarkoittaa sitä että Rayleigh- ja Mie-sironnat määrittelevät itselleen arvot päivänajan mukaan, ja dynaamisen eli reaaliaikaisen valaistuksen rakentaminen on helpompaa kuin aiemmin. Jotta kaikki kolme komponenttia toimisivat yhdessä, on aktivoitava Atmosphere Sun Light -asetus suuntavalosta. Atmosfäärin mahdollisuus vaikuttaa eksponentiaaliseen korkeussumuun aktivoitava projektin asetuksista ja korkeussumun väriarvot oltava nolla, jolloin korkeussumu ottaa valaistuksen taivaan atmosfäärin komponentista. [32]

Näiden komponenttien lisäksi voidaan hyödyntää 3D-volyymitekstuureita, jotka käyttävät 2D-tekstuureja ja esittävät sen 3D-muodossa. Käytännössä volyymi viipaloi 2D-tekstuurin ruudukon avulla ja projisoi kuvat kahdessa suunnassa [33]. Ryan Brucksin tekemä Unreal Enginen liitännäinen, Volumetrics, sisältää algoritmin 3D-kohinatekstuurien generointiin. Tällä liitännäisellä voidaan generoida useita kohinatekstuureja päällekkäin käyttäen eri RGB-kanavia (kuva 9). Kanavien käytöllä mahdollistetaan dynaamisempi lopputulos, sillä yhden kohinatekstuurin sijaan voimme käyttää kolmea. [34]



Kuva 9. Kolme automaattisesti generoitua kohinatekstuuria tallennettuna RGB-värikanaville Unreal Enginessä.

Moottoriversio 4.26:ssa, Unreal Engine esitteli uuden komponentin nimeltään volumetriset pilvet. Toisin kuin aikaisemmat pilvielementit Unreal Enginessä, fysiikkaperusteiseen renderöintiin pohjautuva ratkaisu tuo lisää muokattavuutta. Nämä sädettä marssittamalla luodut 3D-tilavuudet toimivat myös erilaisilla valaistuksilla. Pilvet toimivat myös dynaamisen vuorokaudenvaihtelun kanssa, käyttäen taivaan atmosfääri- ja taivaan valokomponentteja. Kuten muitakin volumetrisiä efektejä, voidaan pilvien muotoa hallita volumetrisellä materiaalilla ja volyymitekstuureilla. [35]

Näiden lisäksi pilviä voidaan lisätä ympäristöihin yksittäisinä komponentteina, joita voidaan skaalata, pyöritellä ja siirrellä X-, Y- ja Z-akseleilla tai maalata suoraan kenttään halutuille alueille. Toimiakseen komponentin tarvitsevat kenttään myös generaattorikomponentin, joka löytyy samasta lisäosasta [35]. Kuvassa 10 on käytössä kaksi pilvikomponenttia ja molemmilla eri arvot pilvien piirtämiseksi. Unreal Enginen versio 4.26:ssa ei ollut vielä tukea volumetrisille pilville virtuaalitodellisuudessa, koska pilvet piirtyivät testivaiheessa vain toiselle silmälle.

Kuva 10. Pilviä Saami-opetuspelin Rastigaisa-kentässä. Näin tarkkojen ja tiiviiden pilvien näytemäärä on hyvin korkea ja sillä on suuri vaikutus kuvataajuuteen.

Volumetristen efektien kanssa voidaan käyttää volumetrisiä valaisukarttoja, jotka tallentavat epäsuoraa valaistusta itseensä. Volumetriset valaisukartat toimivat kuten normaalit UV-kartat, mutta 3D-tilassa [36]. Näin voidaan saada myös dynaamiset objektit toimimaan volumetrisen valaistuksen kanssa, ja yhdistettynä volumetriseen sumuun saadaan luotua oikeaa maailmaa jäljittelevä valaistus [37].

3.2 Komponenttien toimintaa teoriassa

Eksponentiaalinen korkeussumu on sovellettu versio Beerin ja Lambertin laista, ja kaikki kehittyneemmät versiot ovat muunnelmia tästä [23]. Beerin ja Lambertin laki määrittelee valon pidättäytymistä johonkin aineeseen, ja sen laskukaava voidaan esittää esimerkiksi seuraavasti [38].

$$A = \varepsilon\, b\, c \tag{1}$$

A on absorbanssi, ε on molaarinen absorptiokerroin, b on näytteen paksuus ja c on absorboivan aineen konsentraatio. Kaavaa käytetään yleensä kemiassa, mutta sen pohjaa voidaan hyödyntää peleissä, sillä se laskee, minkä verran valo heikkenee läpäistessään kohteen. [39.] Jotta ymmärtää, miten tämä laskukaava vaikuttaa volumetriseen valaisuun, voidaan käyttää esimerkkinä vanhaa Robert Drebinin, Loren Carpenterin ja Pat Hanrahanin laskukaavaa. Sillä voimme laskea, miten paljon valosäde läpäisee vokselia [40]. Laskukaavan voi esittää seuraavasti:

$$C_{out}(v) = C_{in}(v) * (1 - Opacity(x)) + Color(x) * Opacity(x) \quad (2)$$

Cin(V) on valon väri ennen kuin se läpäisee vokselin, Cout(V) on väri läpäisyn jälkeen. Tämä tarkoittaa, että joka kerta kun valonsäde läpäisee vokselin, valon väri kerrotaan käänteisellä vokselin opasiteetilla. Näin simuloidaan Beerin ja Lambertin lain valon absorboituminen [23]. Kuten huomataan, kaavan yksinkertaisuuden vuoksi Beerin lain soveltaminen sumujen laskentaan on hyvin kevyt ratkaisu.

Eksponentiaalisen korkeussumun ja volumetrisen sumun lisäksi Unreal Enginestä löytyy komponentti atmosfäärinen sumu ja taivaan atmosfääri, jotka jäljentävät ilmakehän valon hajoamista. Kuten aikaisemmin todettiin, valon väri ja käyttäytyminen riippuvat partikkelien koosta [6]. Nämä komponentit eroavat muista siinä, että ne jäljittelevät oikeaa sähkömagneettista sirontaa, muuttaen valon väriä sen kulman mukaan. Komponenttien toimintaperiaate on sama, jolloin niitä ei voida käyttää samanaikaisesti.

Pelimoottorissa voidaan säätää sumujen eri arvoja riippuen siitä, minkälaista lopputulosta haetaan, kuten esimerkiksi sirontaa, väliaineen heijastavuutta ja efektin valonläpäisevyyttä. Kun tiedetään, miten partikkelit käyttäytyvät fysiikassa, on mahdollista saavuttaa mahdollisimman autenttinen jäljitelmä.

Unreal Engine 4 käyttää volumetristen efektien tekemiseen suoraa tilavuuden renderöintiä ja säteen marssittamista. Volumetriset efektit koostuvat kameraan suuntautuvista vokseleita eli frokseleista. Volyymi itsessään koostuu vokseleista ja 2D-datasta 3D-avaruudessa.

Unreal Engine käyttää renderöinnissä lykättyä varjostusta. Lykätty varjostus eroaa etuperoisesta renderöinnistä siten, että se käyttää valaistusta laskemiseen, kun taas jälkimmäinen käyttää geometrian rasterointia [41]. Renderöinti itsessään tarkoittaa objektien kartoittamista ja niiden piirtämistä 3D-avaruuteen. Renderöinti vaikuttaa merkittävästi tapaan, miten varjostuksia ja efektejä käsitellään ja mitä pitää ottaa huomioon. Kun etuperoinen renderöinti tekee renderöinnin järjestyksessä, niin viivästetty renderöi ensin koko kentän ja lisää sitten varjostukset päälle [41]. Tällöin tulee ottaa huomioon, miten eri materiaalit ja varsinkin läpinäkyvyys saadaan piirtymään oikeassa järjestyksessä.

Unreal Engine 4 käyttää DirectX 11- ja DirectX 12 -rajapintoja, jotka sisältävät globaalin valaistuksen, valaistun läpikuultavuuden ja jälkikäsittelyefektit kuten myös GPU-partikkelisimulaatiot vektorikenttien avulla [42].

3.3 Suorituskyvyn parantaminen

Volumetriset efektit ovat hyvin raskaita efektejä käyttää. Ne maksavat yhden millisekunnin Playstation 4:n korkeilla asetuksilla ja kolme millisekuntia NVIDIA:n 970 GTX:llä paremmilla asetuksilla. Tämän vuoksi on tärkeää minimoida vokseleiden ylipiirron määrää ja valojen varjostamista. Esimerkiksi piste- ja spottivalot varjojen kanssa maksavat noin kolme kertaa enemmän kuin ilman varjoja. [31.]

Koska tilavuutta mallintava vokseliruudukko on hyvin matalaresoluutioinen suorituskyvyn parantamiseksi, Unreal Engine käyttää väliaikaista uudelleenheijastamista. Uudelleenheijastaminen ottaa edellisiä kehyksiä ja sekoittaa niitä nykyisiin, pehmentäen laskostumista. Kuvan 11 esimerkissä nähdään reunan pehmentämisen vaikutus efektiin. Vasemmalla on mahdollista nähdä efektin laskostuminen, kun reunan pehmennys ei ole päällä. [31.]

Kuva 11. Vasemmassa kuvassa uudelleenheijastaminen on pois päältä, jolloin voidaan nähdä volumetrisen efektin kerrokset. Oikeassa kuvassa päällä, jolloin laskostuminen pehmennetään.

Efektistä voidaan tehdä korkeampiresoluutioinen pienentämällä näköetäisyyttä eli rajaamalla, kuinka pitkälle sumu näkyy [43]. Periaatteessa pituuden lyhentäminen litistää laskoksia lähemmäksi toisiaan. Tällöin efekti ei ulotu niin kauas, mutta on tarkempi läheltä katsottuna.

Volumetristen pilvien optimointiin on useita tapoja. Volumetriset pilvet käyttävät varjojen laskemiseen joko säteen marssitusta tai Beerin varjokarttoja. Beerin varjokartat on tehokas sen nopeuden vuoksi, vaikka tarkkuus ei ole samaa tasoa marssituksen kanssa, mutta Beerin varjokartat kuitenkin toimivat kaukaa katsotuissa pilvissä. Pilville voidaan antaa mahdollisuus luoda varjoja

ja valonsäteitä, mutta koska varjojen luonti on kallista, on hyvä tarkistaa pilvien varjokartan resoluutio. [35.]

Aiemmin mainittujen lisäksi Unreal Enginen dokumentaatio sisältää useita muita optimointitapoja, lähtien volumetristen pilvien materiaalin käytöstä ja mahdollisista komentokehotteista [35]. Pilvien käyttötarkoitus ja ympäristö määrittelevät optimointitarpeen, ja näin ollen ei ole yhtä oikeaa tapaa tai keinoa optimoida. Volumetrisiä pilviä testatessa voidaan huomata, miten eri komponenttien arvot ja ympäristön laajuus vaikuttivat suorituskykyyn eri tavoin. Esimerkiksi alla kuvassa 12 laskettiin varjojen tarkkuutta, jotta saatiin lisää tiheyttä pilviin.

Kuva 12. Optimoiduillakin pilvillä on mahdollista saavuttaa näyttäviä lopputuloksia.

4 Volumetristen efektien lisääminen kenttään Unreal Enginessä

4.1 Volumetrinen efekti

Volumetrisen efektin suunnittelu on yhtä tärkeää kuin sen tekeminen, koska halutaan välttää pelin kuormittamista turhilla efekteillä. Kun tiedetään, mihin käyttötarkoitukseen volumetristä efektiä tarvitaan, voidaan miettiä sen rakennetta. Tarvitaanko globaalia, koko kentän kattavaa sumua vai riittääkö paikallinen, tarkasti rajattu alue? Halutaanko sumuun tekstuuria vai onko sumu tasaista massaa? Onko sumu pääasia vai sen läpi pääsevä valo?

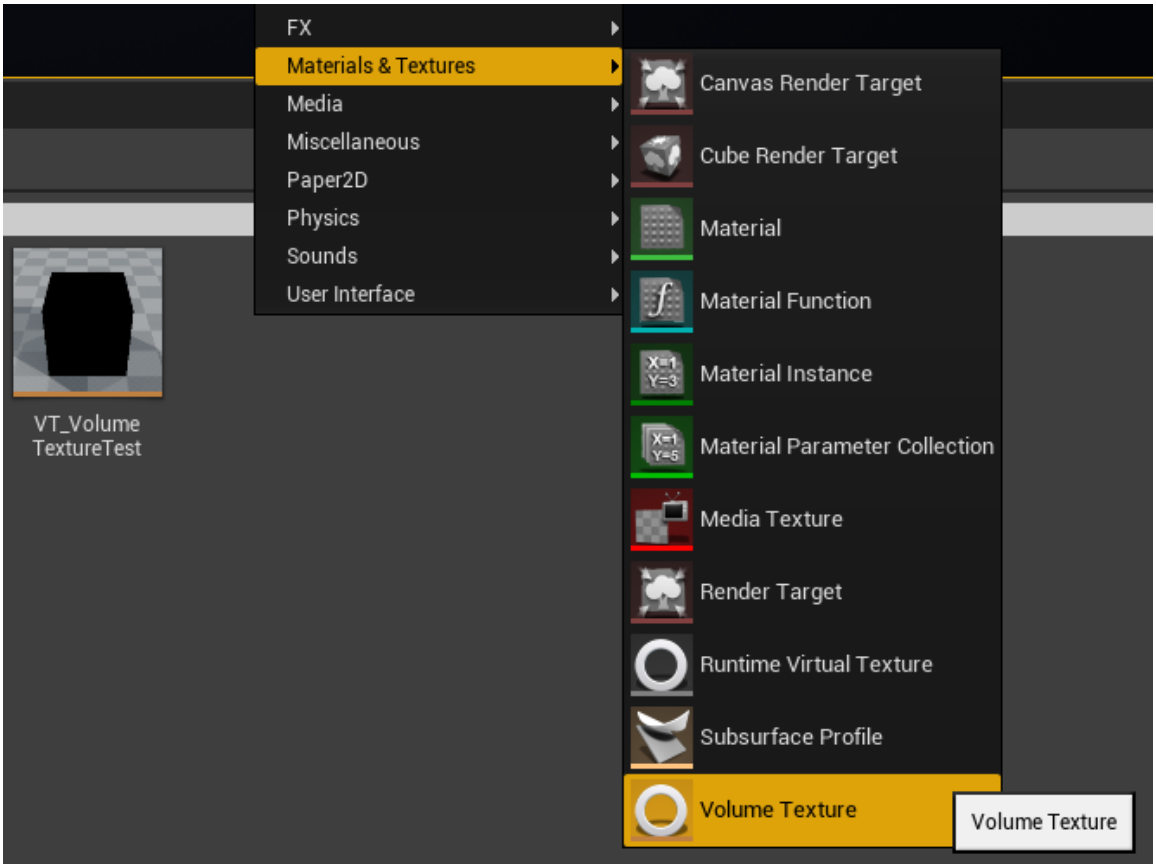
Seuraavassa osiossa käydään läpi Unreal Enginen volumetristen sumut ja tarkastellaan, miten komponentit käyttäytyvät yhdessä. Esimerkin komponentteina toimivat eksponentiaalinen korkeussumu ja taivaan atmosfääri, sekä polygoniverkko, johon lisäämme paikallisen sumun materiaalin. Lopuksi lisäämme esimerkkiin volumetriset pilvet. Lisäksi käytetään muutamaa konsolikomentoa, joilla voidaan säätää sumun laatua testausta varten. Esimerkki on tehty Unreal Engine 4:n versio 4.26:lla, ja efektit lisätään valmiiseen kenttään Pixel Perfect Polygonsin Dead hills landscape 2.0 -assettipaketista.

4.2 Volyymitekstuuri

Volume texture eli volyymitekstuuri on Unreal Enginen keino tallentaa 3D-volyymin informaatio 2D-tekstuuriin [44]. Ensimmäisessä materiaaliesimerkissä käytetään volyymitekstuurissa niin sanottua Perlin-Worley-kohinatekstuuria (kuva 13). Teksturoimalla luodaan sumuun eloa ja muotoa. Tämä on automaattisesti generoitu kohina, joka yhdistelee kahta muuta kohinatekstuuria. Kohinatekstuurista puhuttaessa tarkoitetaan 2D-tekstuuria, joka liitetään volyymitekstuuriin, joka käyttää kohinatekstuuria 3D-tekstuurin luomiseen.

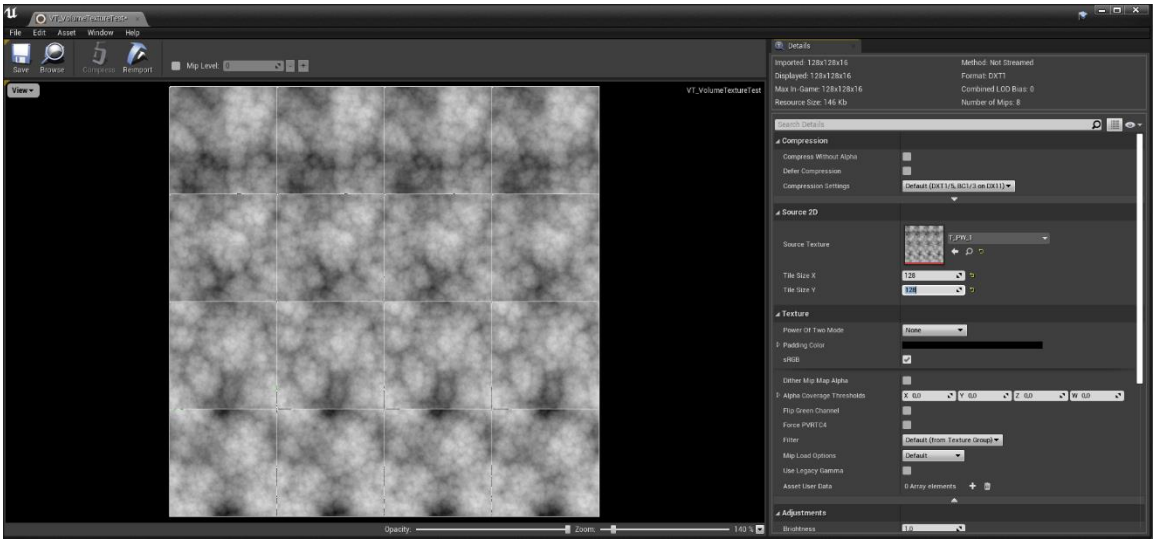
Kuva 13. Erilaisia kohinatekstuureita Sharedtoy-ohjelmalla generoituna. [45.]

Esimerkissä käytetään valmista Asher Zhun jakamaa 512x512 kokoista kohinatekstuuria, joka on valmiiksi paloiteltu 128 ruudukolle sopivaksi. Esimerkin yksinkertaistamiseksi kohinatekstuuri käyttää vain yhtä harmaakanavaa. Perlin-Worley-tekstuureita on mahdollista tehdä esimerkiksi Substance Painterissa yhdistämällä proseduraalisia Worley ja Perlin kohinatekstuureita [34]. Kun tekstuuri on tuotu pelimoottoriin, voidaan luoda volyymitekstuuri (kuva 14).



Kuva 14. Volyymitekstuurin luonti Unreal Engine 4:ssä.

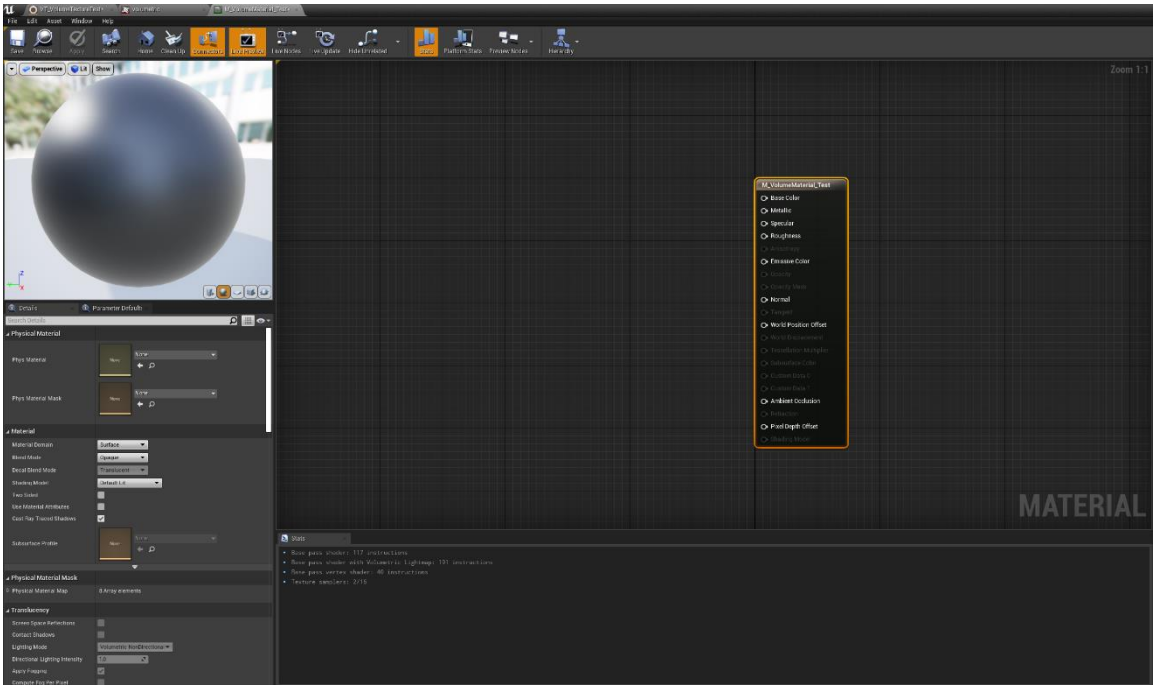
Volyymitekstuuriin lisätään aiemmin tuotu 2D-tekstuuri ja määritellään, minkä kokoisella ruudukolla tekstuuri viipaloidaan. Kuvassa 15 tekstuurille annetaan arvot 128x128, jolloin volyymi näyttää tekstuurin ruudukolla. Näkymän voi myös vaihtaa 3D-näkymäksi, jossa tekstuurin 3D-esitystä on mahdollista esikatsella. Volyymitekstuurin luonnin jälkeen siirrytään materiaaliin, jossa tekstuuria käytetään.



Kuva 15. Volyymitekstuuri ja kohinatekstuurin viipalointi.

4.3 Materiaali

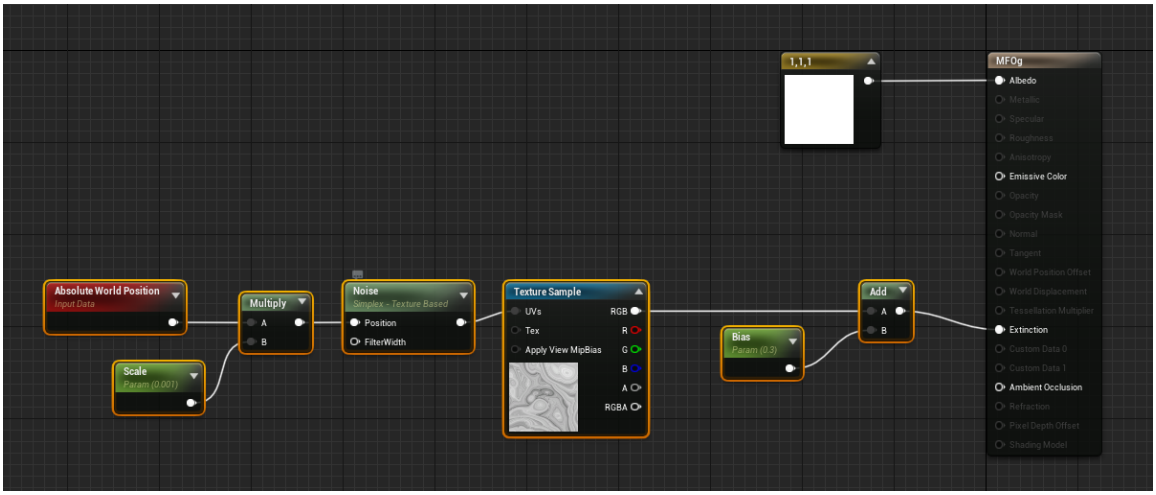
Unreal Enginen materiaalivalikosta luodaan materiaali. Unreal Enginessä materiaaleja hallitaan graafilla (Kuva 16), johon rakennetaan noodeilla halutut ominaisuudet. Vasemmassa yläkulmassa sijaitsee materiaalin esikatselu, sen alla materiaalin parametrivalikko ja asetukset ja keskellä graafi. Ennen kuin lisätään materiaaliin noodeja, säädetään materiaalialue. Materiaalialue määrittelee, miten Unreal käsittelee materiaaliin tulevia arvoja. Tässä tapauksessa materiaalialue tulee olla volyymi ja sen alla sekoitustila additiivinen. Sekoitustila määrää, miten materiaalin värit sekoittuvat taustan kanssa. Materiaalialue ja sekoitustila määrittelee, mitkä ulostulot materiaalissa ovat käytössä. Valaisutilaksi jätetään normaali valaistus, koska halutaan, että sumu reagoi valoihin.



Kuva 16. Materiaali-ikkunan graafi-editori Unreal Enginessä.

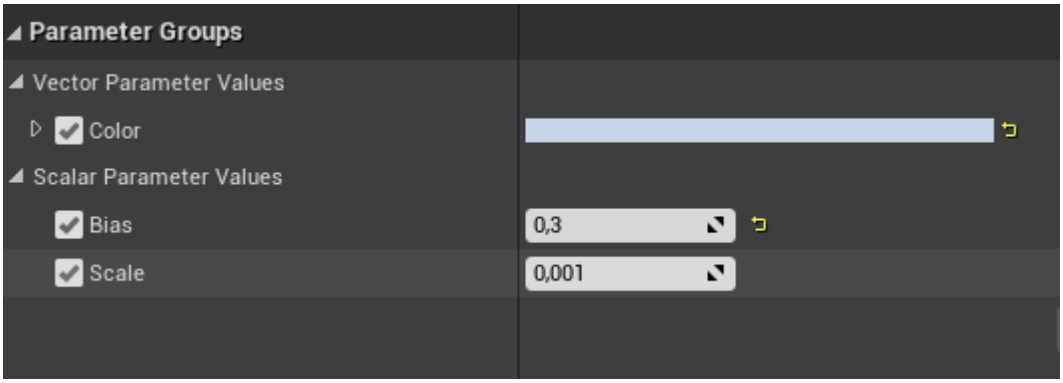
Seuraavaksi lisätään tarvittavat noodit graafiin. Tarvitaan vektorinoodi värille ja volyymitekstuuri omaksi noodikseen graafiin. Lisäksi annetaan parametrejä materiaalille, jotta voidaan myöhemmin vaihtaa arvoja materiaali-instanssissa. Materiaalissa käytetään kahta sisääntuloa: pääväriä ja katoamista. Pääväriin liitetään kolmiulotteinen vektori-noodi, joka muutetaan parametriksi. Parametrin nimeksi tulee Color. Tällä noodilla voidaan vaihtaa sumun väriä. Volyymitekstuuri muuntautuu Texture sample -nimiseksi noodiksi, kun sen pudottaa graafiin.

Ennen kuin tekstuuri liitetään materiaalin ulostulonoodiin ja lisätään skaalattava parametri nimeltä Bias eli vääristymä, jolla voidaan myöhemmin vaikuttaa sumun vahvuuteen (kuva 17). Ennen tekstuuria lisätään myös muutama noodi. Ensin lisätään absoluuttinen maailman sijainti etäisyyden laskemiseen kamerasta pikseleihin ja kerrotaan skaalattavalla parametrilla alaspäin, jotta saadaan tarkempi lopputulos pienemmälle alueelle. Lisätään lopuksi kohinanoodi ennen volyymitekstuuria, jotta saadaan lisää satunnaisuutta tekstuuriin. Kohinanoodin voi myös jättää pois ja vaikuttaa sumun muotoihin kohinatekstuuria vaihtamalla.



Kuva 17. Materiaalin noodit.

Seuraavaksi luodaan materiaalista instanssi. Materiaali-instanssi on esikäsitelty materiaali, jota voi muokata ilman kääntämistä. Instanssin sisällä pystytään muuttamaan skaalattavien parametrien asetuksia ja parametriksi muutettua värivektoria (kuva 18). Säädetään vääristymää ja skaalaa ja vaihdetaan materiaalin väriksi hieman sinertävä vivahde, jolloin sumu alkaa olla valmis kenttään testattavaksi.



Kuva 18. Materiaaliin asetetut skaalattavat parametrit löytyvät materiaali-instanssista.

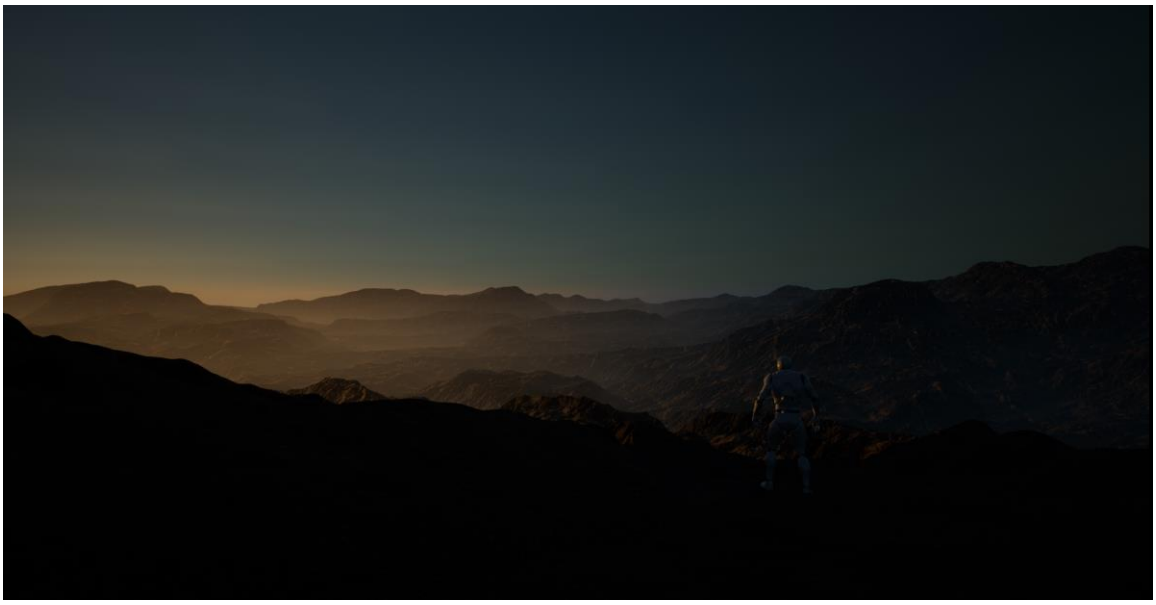
4.4 Kentän komponentit ja asetukset

Aloitetaan poistamalla kentästä valmiiksi asetetut komponentit ja lisätään taivaan atmosfääri sekä suuntavalo kontrolloimaan taivasta ja korkeussumua. Kuvassa 19 on havaittavissa ilmakehän hajontaa. Hajonta on kuitenkin edelleen vähäistä ja mannekiini hukkuu taustan varjoihin.

Kuva 19. Dead hills Landscape 2.0 ”Sky Atmosphere” -komponentilla ja suuntavalolla.

Seuraavaksi lisättävä komponentti on eksponentiaalinen korkeussumu, jolla vahvistetaan ilmakehässä tapahtuvaa hajontaa. Kuvassa 20 on nähtävissä korkeussumun vaikutus sirontaan. On muistettava kytkeä päälle taivaan atmosfäärin vaikutus korkeussumuun projektinhallinnasta, mikäli eksponentiaalisen korkeussumun halutaan ottavan vaikutteita toisesta komponentista. Kuten aiemmin todettiin, eksponentiaalisen korkeussumun hajonnan vahvuus vaihtelee korkeuden mukaan. Komponentin asetuksissa voidaan vaikuttaa siihen, millä korkeudella sumu sijaitsee ja kuinka voimakasta häipyminen on. Koska sumu reagoi taivaan atmosfääriin, värien säätö tapahtuu sitä kautta. Lisäksi komponentista voidaan kytkeä päälle volumetrinen sumu ja sen asetukset. Kuvissa 20 ja 21 asetus ei ole vielä päällä. Kuvissa 20 ja 21 voidaan myös havainnoida taivaan atmosfäärin vaikutusta ympäristöön. Sirontatyyppi vaihtuu suuntavalon sijainnin mukaan. Komponentissa voidaan asettaa arvoja planeetan koolle ja atmosfäärin korkeudelle.

Kuva 20. Dead hills Landscape 2.0, eksponentiaalisen sumun kanssa.



Kuva 21. Dead hills Landscape 2.0, suuntavalon, eli ”auringon”, sijainnin muuttaminen vaikuttaa sironnan väriin ja käyttäytymiseen.

Kuvassa 22 volumetrinen sumu on kytketty päälle ja taivaalta tulevat valonsäteet saavat tilavuuden. Asetuksista voidaan säätää myöhemmin lisättävää paikallista sumua. Aikaisemmin mainitut komponentit ovat globaaleja efektejä, ja niiden käyttö on kevyempää kuin seuraavaksi esiteltävän volumetrisen sumun. Paikallinen volumetrinen sumu vaatii eksponentiaalisen sumun Volumetric Fog -asetuksen aktivoinnin.

Kuva 22. Dead hills Landscape 2.0, globaali volumetrinen sumu päälle kytkettynä. Taustalla näkyvissä vanha taivaskomponentti, joka korvataan volumetrisillä pilvillä kappaleessa 4.5.

Paikallisen sumun rakentaminen aloitetaan asettamalla halutulle sumualueelle moottorista löytyvä polygoniverkko nimeltä kuutio. Kuutio skaalataan täyttämään alue, jolle efekti vaikuttaa; näin ollen se toimii volumetrisen efektin rajoina (Kuva 23). On myös toinen tapa rajata volumetrisen efektin vaikutusalue: kenttään lisättävä partikkelijärjestelmä, joka luo partikkeleita. Partikkelit vokselisoidaan eli niistä luodaan 3D-esitykset. Yksi partikkeli toimii itsenäisesti sumun rajana.

Kun kuutio on halutulla paikalla, voidaan se asettaa käyttämään volumetrisen materiaalin instanssia. On myös mahdollista käyttää alkuperäistä materiaalia, mutta parametrien säätelyn helpottamiseksi kannattaa suosia instansseja.

Kuva 23. Dead hills Landscape 2.0, valkoinen kuutio paikoillaan.

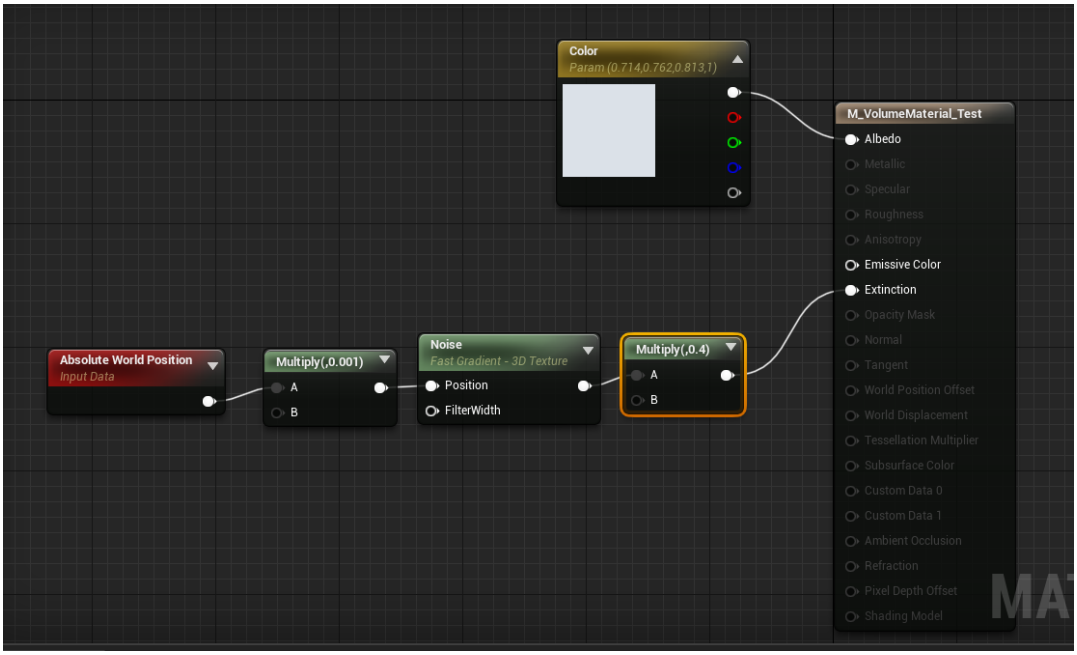
Volumetriset vokseliruudukot ovat Unreal Enginen perusasetuksilla hyvin matalaresoluutioisia, joten isolle alueelle sijoitettuna efektiin tulee rakeisuutta (kuva 24). Tämä johtuu siitä, että eksponentiaalisessa korkeussumussa näköetäisyys on määritelty hyvin kauas, jotta valonsäteet piirtyvät horisontissa. Tilavuutta on mahdollista tarkentaa, mutta se on hyvin raskasta grafiikkaprosessorille, eikä näin ollen mahdollista kaikissa tilanteissa. Tässä tapauksessa pienennetään sumun ja yleisten volumetristen efektien vaikutusaluetta, jotta sumusta saadaan korkeampiresoluutioinen. Toisin sanoen nostetaan resoluutiota näköetäisyyden kustannuksella [43]. Toinen vaihtoehto olisi käyttää kahta erilaista sumua: matalampaa resoluutiota ympäristöön ja korkeampaa lähietäisyydelle, ja vaihtaa näkyviin toinen riippuen pelaajan sijainnista [43].

Kuva 24. Dead hills Landscape 2.0, paikallisen sumun koon ja asetusten johdosta efektin resoluutio on hyvin matala.

Vaihtoehtoinen tapa tehdä pilvimäisyys on generoida se kohinanoodilla. Kohinanoodilla generointi toimii ympäristössä kevyemmin, koska materiaali sisältää vähemmän tarkkuutta (kuva 25). Koska aiempi tekstuuriin pohjautuva tekniikka ei sovi esimerkissä käytettyyn laajaan alueeseen, voidaan käyttää yksinkertaisempaa kohinanooditekniikkaa. Mikäli sumua käytettäisiin vain lähietäisyydellä, olisi volyymitekstuurin käyttö perusteltua. Volyymitekstuuria käytetään myöhemmin modulaarisen volumetrisen efektin teossa kappaleessa 5.

Kertomalla kohinanoodia eri arvoilla saadaan aikaan vaihteleva pilvimassa. Kohinan lisäksi materiaalissa voidaan edelleen vaihtaa esimerkiksi pilvien väriä. Tärkeimpänä noodina kohinan lisäksi on edelleen absoluuttinen maailman sijainti, sillä tämän noodin avulla katsotaan, missä kulmassa efekti on kameraan nähden (Kuva 26).

Kuva 25. Dead hills Landscape 2.0, kohinalla generoitu pilvimassa.



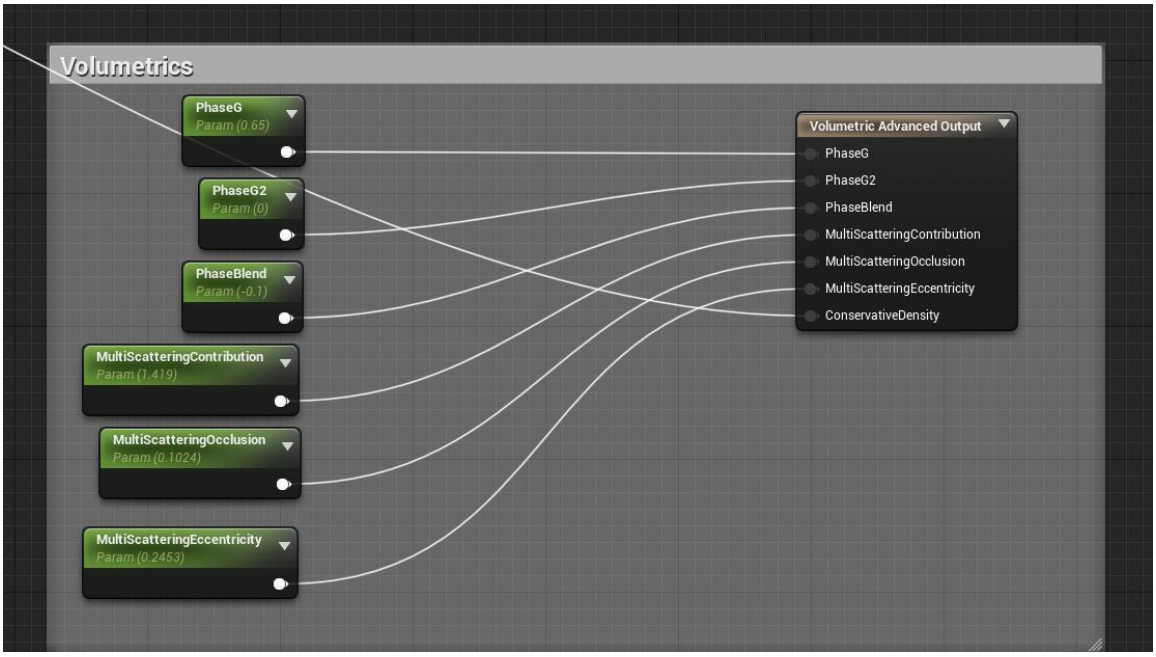
Kuva 26. Pelkällä kohinalla luotu pilvimateriaali.

Viimeiseksi säädetään hieman efektien arvoja, jotta saadaan kaikki sulautumaan yhteen. Taivaalla näkyvät valonsäteet jätetään pois, mutta pienentämällä efektin vaikutusaluetta voidaan käyttää korkeamman resoluution ruudukkoa, jolloin rakeisuus häviää. Kun saadaan säädettyä pilvimassat väreiltään ja valoarvoiltaan sopivaksi kentän valaistukseen, eivät ne näytä enää irrallisilta (kuva 27). Kuvasta voidaan myös nähdä valon vaikutus paikallisiin pilviin: ne tummenevat varjoalueelle siirtyessä. Yksinkertaisellakin materiaalilla voidaan saavuttaa toimivia ratkaisuja volumetristen efektien rakentamiseen.

Kuva 27. Valmiit volumetriset efektit.

4.5 Volumetriset pilvet

Lopuksi ympäristöön lisätään Volumetric Cloud -komponetti. Komponentti käyttää automaattisesti Unreal Enginen volumetristen pilvien materiaali-instanssia, mutta komponentille voi antaa myös kustomoidun pilvimateriaalin. Kuten muut volumetriset materiaalit, pilvien materiaalialue on volyymi ja sekoitustila additiivinen. Toimintaperiaate on sama kuin itseluodussa volumetrisessä materiaalissa, mutta pilvet käyttävät Unreal Enginen omaa Volumetric Advanced Output -noodia lisänä. Tämä noodi säätelee esimerkiksi pilvien läpinäkyvyyttä ja valon hajontaa (kuva 28). Komponentti käyttää valmiiksi pelimoottorin omaa pilvimateriaali-instanssia, mutta tämä esimerkissä on käytössä Michael Kinseyn Dinusty-tutoriaalin esimerkki [46].



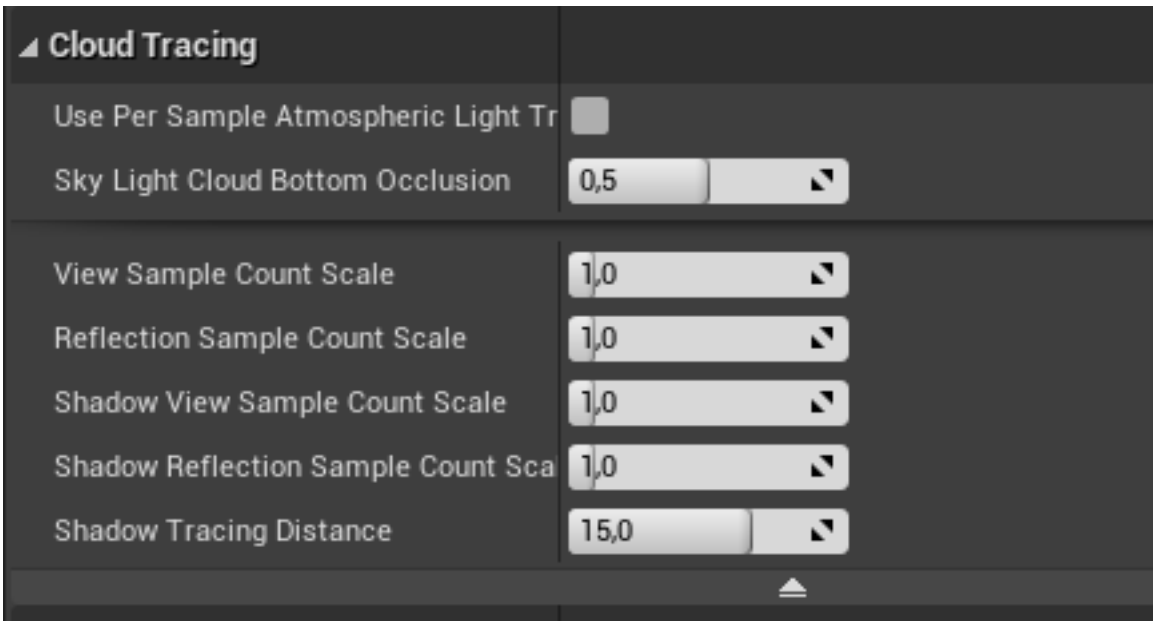
Kuva 28. Volumetristen pilvien ulostulonoodi.

Materiaali-instanssissa on säädöt pilvien koolle, tiiviydelle, kontrastille, nopeudelle ja peittävyydelle. Volumetrisen noodin asetuksilla voidaan vaikuttaa eri kanavien valon hajontaan ja vaiheiden sekoitukseen. Näiden muutamien asetusten avulla on mahdollista luoda eri ympäristöön sopivia pilvimassoja. Kuvissa 29 ja 30 on nähtävissä saman materiaali-instanssin eri asetusten vaikutukset, ja jo pienillä muutoksilla saavutetaan huomattavia eroja.

Kuva 29. Dinusty-tutoriaalin materiaali-instanssi.

Kuva 30. Aiemmassa kuvassa mainittu materiaali-instanssi eri parametreilla ja komponenttiasetuksilla.

Tiiviimmät ja peittävämmät pilvet tarvitsevat enemmän näytteitä kuin hajanaisemmat. Volumetristen pilvien komponentin asetuksista näytemäärää voidaan säätää suuremmaksi, jolloin pilvien rakeisuus katoaa. Kuvassa 31 on pilvien jäljitysasetukset oletusarvoissa. Kuvan esimerkkiä varten näytteiden skaalaa nostettiin kolmeen ja varjojen jäljitystä kahteenkymmeneen. Tällä oli jo huomattavia vaikutuksia kuvataajuuteen: oletusarvoilla kuvataajuus oli noin 120, ja kolminkertaisilla näytteillä kehysnopeus laski noin 80 ruutuun sekunnissa.



Kuva 31. Volumetristen pilvien komponentissa on useita laadun ja suorituskyvyn kannalta oleellisia asetuksia, joilla testaamalla löytää sopivat arvot.

5 Modulaarinen volumetrinen efekti

Tässä käytännön osiossa käydään läpi modulaarisen volumetrisen efektin rakentaminen Clever Simulation Entertaimentille. Koska volumetriset efektit voivat edelleen olla raskaita mobiilialustoilla, haluttiin tutkia sumuefektien implementointia kustannustehokkaasti. Monien eri käyttötarkoitusten vuoksi päädyttiin mahdollisimman modulaarisen materiaalin valmistamiseen, jolla on mahdollista mallintaa erilaisia efektejä useisiin projekteihin. Toisin kuin aiemman osion yksinkertaistettu paikallinen volumetrinen efekti seuraava materiaali rakentuu useasta eri osiosta ja käyttää myös volyymitekstuuria. Materiaalin teon aikana tutustutaan volyymikurvien ja varjojen käyttöön.

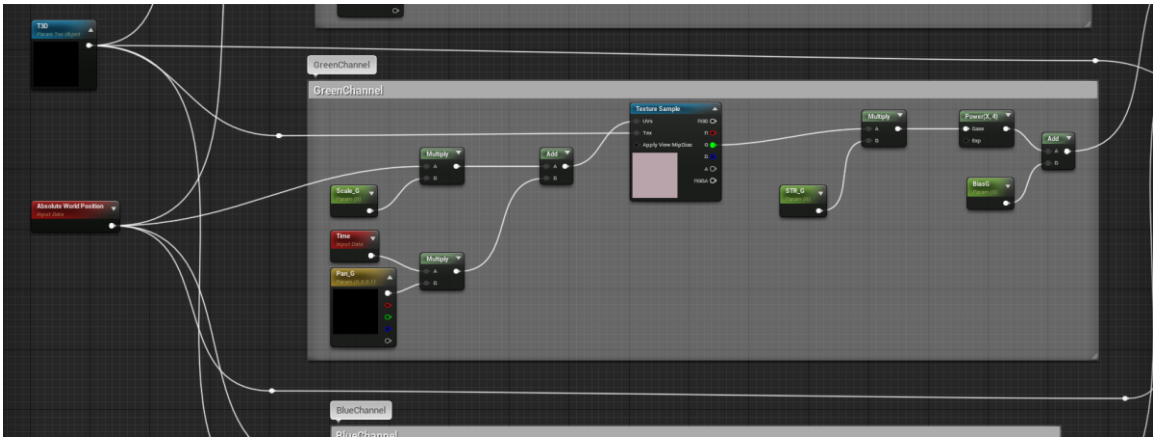
Projekteissa käytetään Unreal Enginen versio 4.26:tta, joka tukee uutta volumetristen efektien liitännäistä nimeltä Volumetrics, jota hyödynnettiin materiaalia luodessa. Liitännäinen aktivoidaan erikseen jokaisessa projektissa. Volumetrisen efektin materiaali käyttää volyymitekstuuria, volyymikurvia kartassa ja monikanavaisia kohinatekstuureja. Materiaalin luonti aloitettiin kartoittamalla erilaiset mahdolliset käyttötarkoitukset. Lähtökohtana oli, että efektiä voidaan hyödyntää usvana, lumi- ja hiekkamyrskynä, ja että se toimii erilaisilla alustoilla ilman mittavia optimointitarpeita.

Materiaalin pohja tehtiin testikentässä, jonka jälkeen sitä testattiin eri ympäristöissä ja korjattiin käyttäytymisen mukaan. Testatessa huomattiin muutamia teknisiä rajoitteita, jotka käydään myöhemmin läpi esimerkkien kanssa.

5.1 Materiaali ja materiaali-instanssi

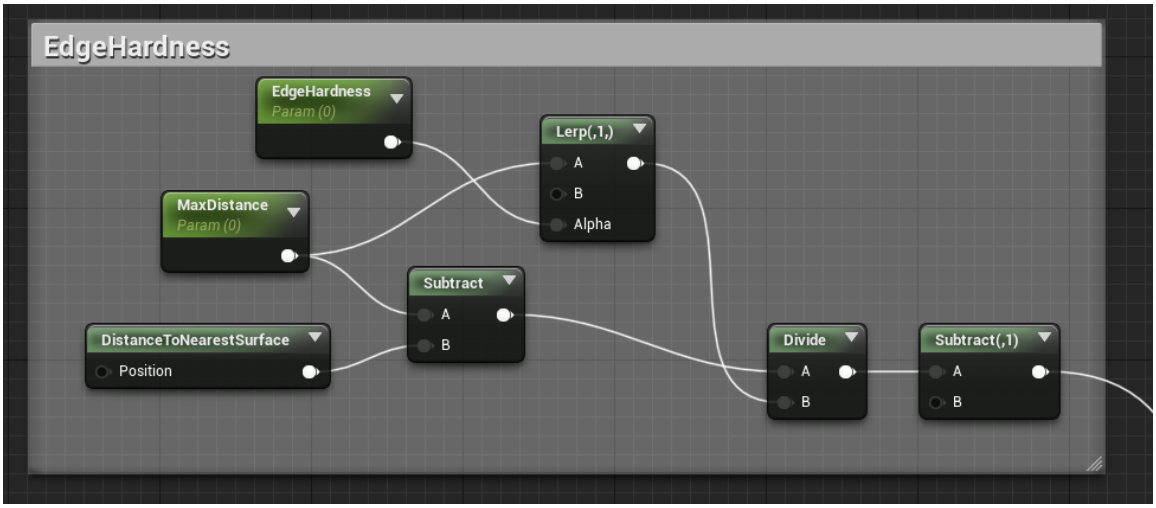
Materiaali käyttää Unreal Enginen Volumetrics-pluginin volyymitekstuuria, josta otettiin käyttöön kolme eri kanavaa: vihreä, punainen ja sininen (kuva 9). Kaikille kanaville luodaan materiaalissa yksilölliset asetukset niin vahvuuden, vääristymän ja nopeuden säätelyyn (kuva 32). Tämän jälkeen kanavat yhdistetään ja niille määritellään etäisyys maasta, reunan pehmeys ja väri. Kuvassa 32 tekstuuri ottaa absoluuttisen maailman sijainnin ja skaalautuu sen mukaan. Aika- ja kolmiulotteinen vektori -noodeilla määritellään efektin UV-kartan liikkuminen ja tekstuurin jälkeen säädetään vahvuutta ja vääristymää. Kaikkien kanavien tekstuurinäyte käyttää tekstuuriobjektia refe-

renssiä. Lopuksi efektille voidaan asettaa sekä sisäiset että ulkoiset heittovarjot. Materiaalialueena käytetään jälleen volyymiä ja lisäsekoitustilaa, jotta saadaan tarvittavat pääväri- ja katoamisulostulot käyttöön.



Kuva 32. Vihreän kanavan asetukset.

Tekstuurin UV-kartan asetusten jälkeen punainen, vihreä ja sininen kanava yhdistetään summaamalla ne toisiinsa. Summaamisen jälkeen lasketaan etäisyys lähimpään pintaan (kuva 33). Jotta efekti saadaan myötäilemään erilaisia pintoja, on projektista aktivoita etäisyyskenttien generointi päälle projektinhallinta-asetuksista. Etäisyyskenttien generointi tallentaa lähimpien pintojen etäisyyden volyymidataan [47]. Mikäli käytetään polygoniverkkoja, kuten kuutiota, efektin rajoina partikkelijärjestelmän sijaan, on syytä muistaa, että myös tilavuuden käyttämästä polygoniverkosta lasketaan etäisyyskenttiä, mikä aiheuttaa efektin vääränlaista käyttäytymistä (Kuva 34). Ongelma on ratkaistavissa polygoniverkon asetuksissa, jossa voidaan poistaa haluttujen objektien vaikutus etäisyyskenttien generointiin kohdasta Affect distance field lighting.

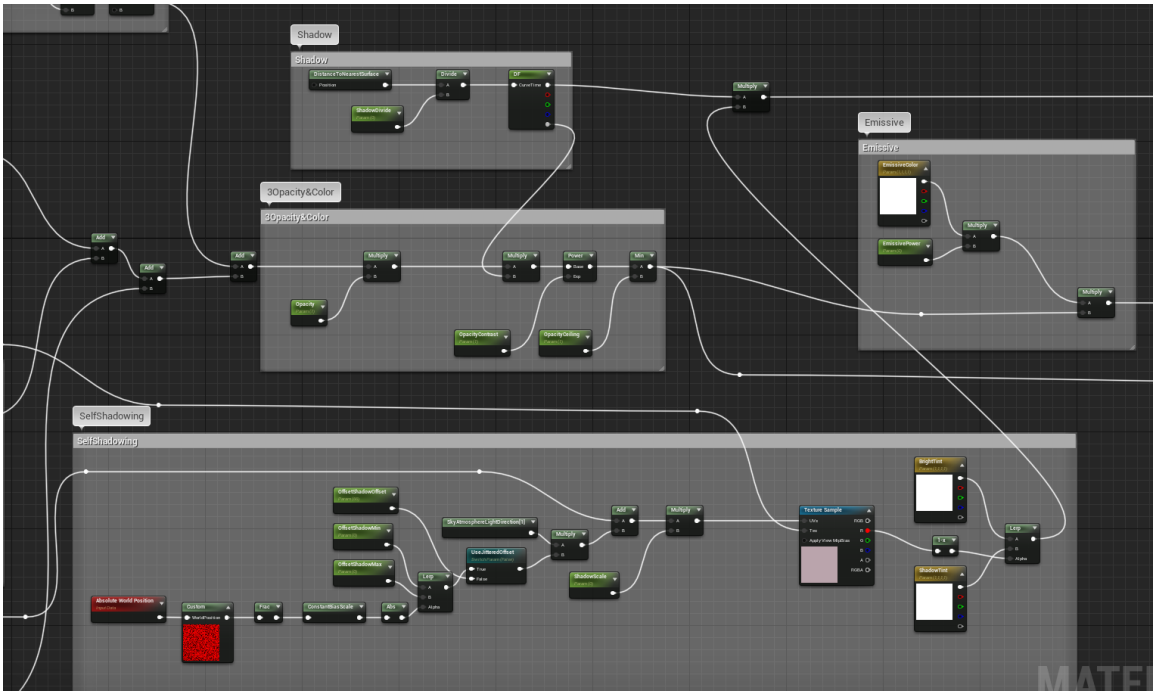


Kuva 33. Etäisyyden laskeminen pintoihin, etäisyyden määrittäminen ja reunan pehmennys.

Kuva 34. Mikäli etäisyyskenttien generointia ei ota pois päältä ei-toivotuilta polygoniverkoilta, voi efekti käyttäytyä väärin.

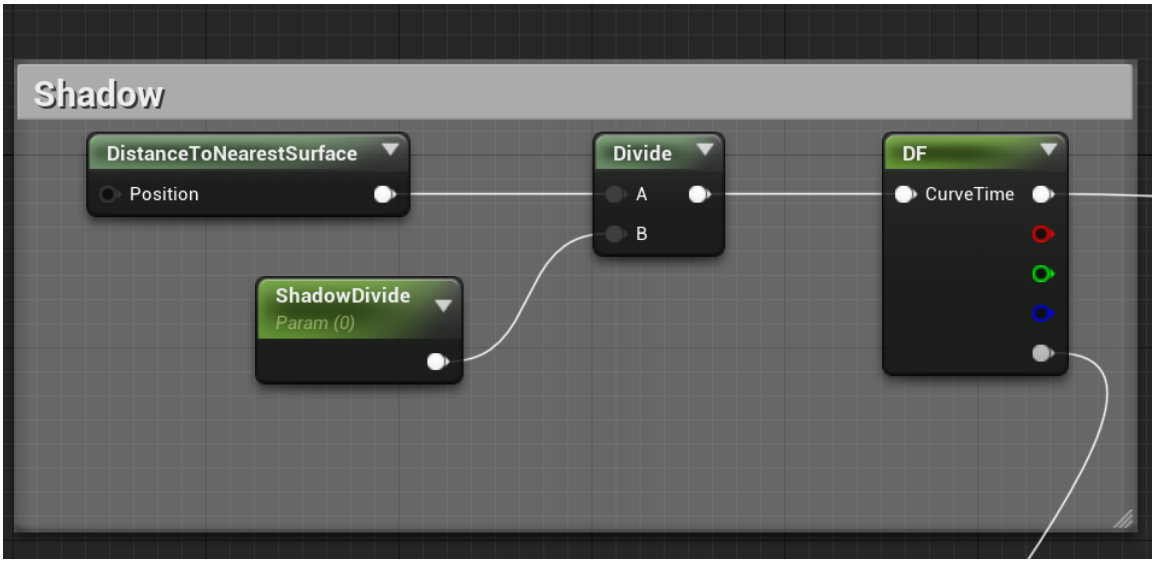
Viimeisenä efektiin lisätään läpinäkyvyyden, värin ja varjojen parametrit, jotka kaikki linkittyvät toisiinsa (Kuva 35). Järjestys on eri osien yhdistämisessä erityisen tärkeää, sillä kuten missä tahansa laskussa, kaavan järjestys vaikuttaa lopputulokseen. Ensin asetetaan läpinäkyvyyden säätö, väliin heittovarjon asetukset, jonka jälkeen läpinäkyvyyden kontrastin ja maksimiläpinäkyvyyden säädöt. Järjestys on tärkeä, koska heittovarjon toiminta perustuu tilavuuden läpinäkyvyyteen, kuten tullaan myöhemmin havaitsemaan. Lopuksi lisätään sisäiset varjostukset, jota kautta voidaan määritellä efektin tummien ja vaaleiden osien värit.

Varjoja on mahdollista implementoida monella eri tavalla, kuten säteenmarssituksella, laskemalla pintojen etäisyyksiä ja vaihtamalla värejä läpinäkyvyyden mukaan. Tämä efekti käyttää kahta tapaa, joilla huijataan sumulle varjoalueita, sillä niin sanotut oikeat varjot säteitä marssittamalla ovat liian raskaita käytettäväksi useissa tilanteissa, kuten aiemmin todettiin kappaleessa 3.3.

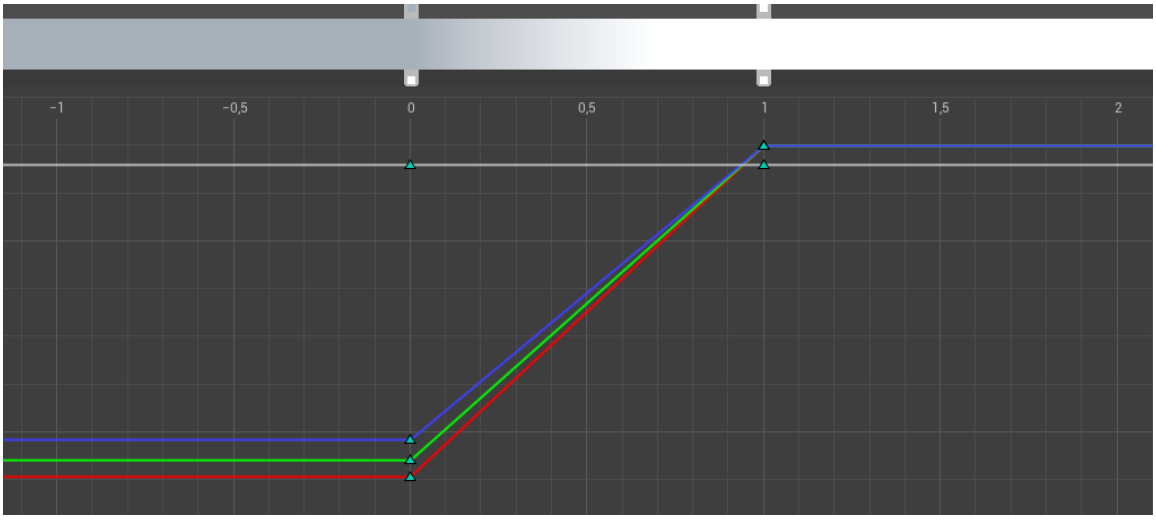


Kuva 35. Eri osien käyttöjärjestys efektin materiaaligraafissa on tärkeä.

Ensin implementoitavat heittovarjot käyttävät pintojen etäisyyden laskentaa, ja parametrilla määritellään, miten kauas varjot osuvat. Kurvilla asetetaan varjon läpinäkyvyys: määritellyllä etäisyydellä varjon läpinäkyvyys on yksi, ja sen ulkopuolella nolla. Kurvien käytöllä mahdollistetaan varjojen ja värien helppo muokattavuus (kuvat 36, 37), sillä komponenteilla voidaan tallentaa useita eri väridatoja yhteen kurvikarttaan ja kutsua niitä noodilla materiaalin sisällä, kuten kuvassa 36. Kuvassa 37 on nähtävissä kurvi, jonka väriarvot ovat alle nolla varjoissa ja nousevat yhteen valaistulla puolella.

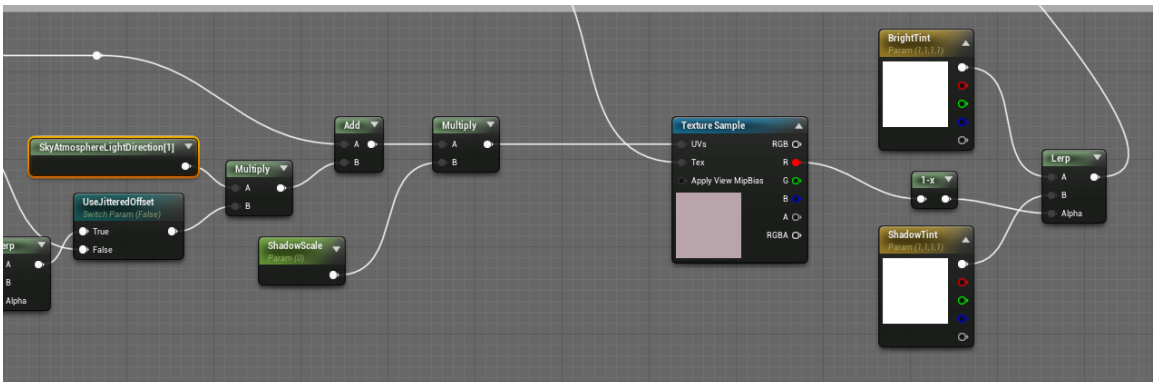


Kuva 36. Ulkoisen heittovarjon noodit. Ensin otetaan etäisyys lähimpään pintaan, jaetaan se halutulla luvulla ja määritetään lineaarisella kurvilla varjon väri.



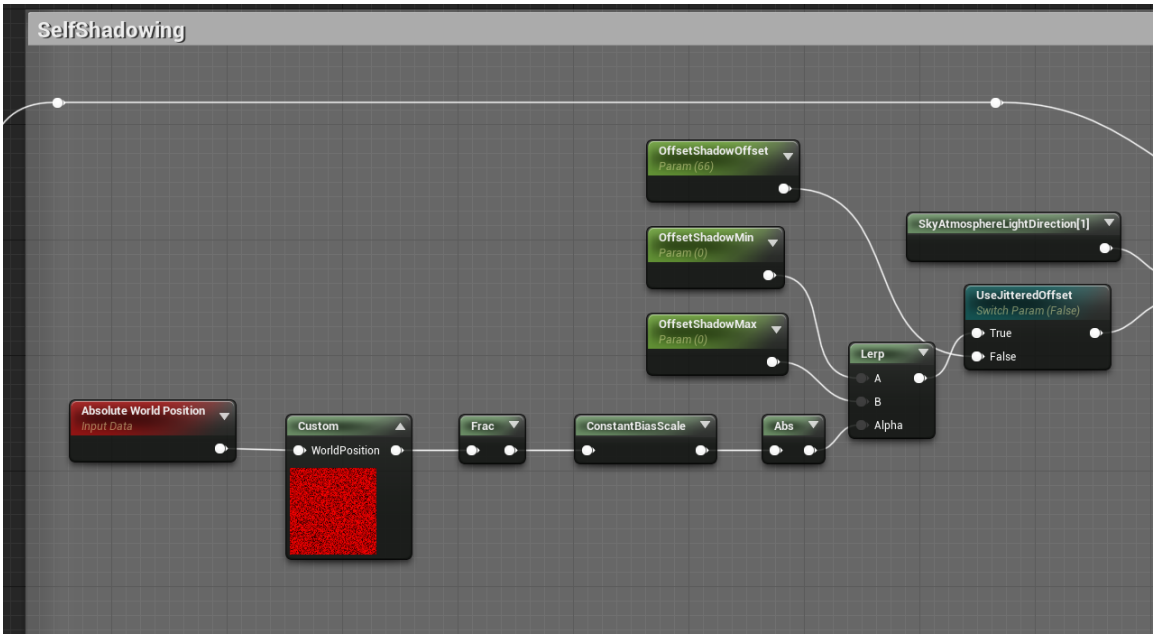
Kuva 37. Kurvikartan kurvi, jolla määritellään varjon väri.

Lopuksi materiaalille asetetaan värien säädöt, joilla myös luodaan efektille varjot. Materiaalissa otetaan huomioon suuntavalon vaikutus sumuun, jonka noodi on osa Unreal Enginen uutta taivaan atmosfääri -järjestelmää (kuva 38) [48]. Periaate on ottaa maailman sijainti A ja liikuttaa sitä suuntavaloa kohti sijaintiin B. Sen jälkeen verrataan sijainteja keskenään, ja mikäli B-sijainnin tiheys on suurempi kuin nolla, tummennamme sijaintia A sen mukaan [34].

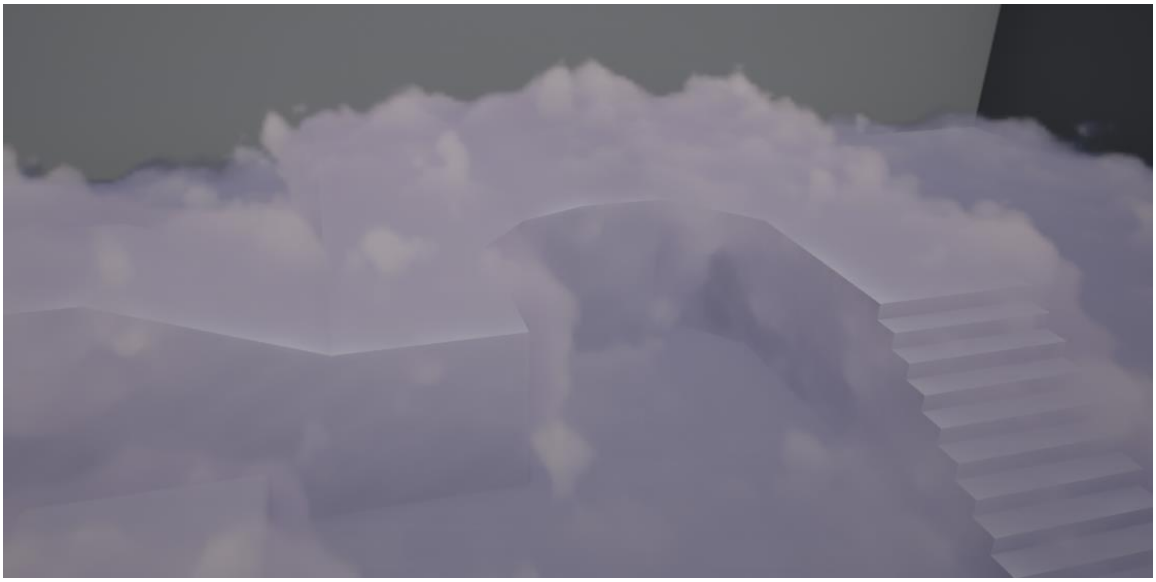


Kuva 38. Suuntavalon suunnan ja vaaleiden ja tummien osien värien määrittely.

Materiaalissa on myös käytössä Asher Zhun mukautettu uudelleenheijastaminen, jolla varjoja voidaan pehmentää entisestään (kuva 39). Perinteisen uudelleenheijastamisen lisäksi otetaan varjojen absoluuttinen maailman sijainti, jota siirrellään sattumanvaraisesti koodilla. Tämä värinä toimii pehmentävänä efektinä [34]. Mikäli varjoissa on havaittavissa artefakteja tai laskostumista, voi kokeilla parantaa laatua ottamalla tämän osion käyttöön. Mukautettu uudelleenheijastaminen ei kuitenkaan ole automaattisesti päällä, vaan sen pystyy tarvittaessa aktivoimaan materiaali-instanssin kautta. Kuvassa 40 varjoalueet violettina ja valokohdan valkoisella. Mitä syvemmälle valonsäteet kulkeutuvat tilavuudessa, sitä tummemman arvon ne saavat.

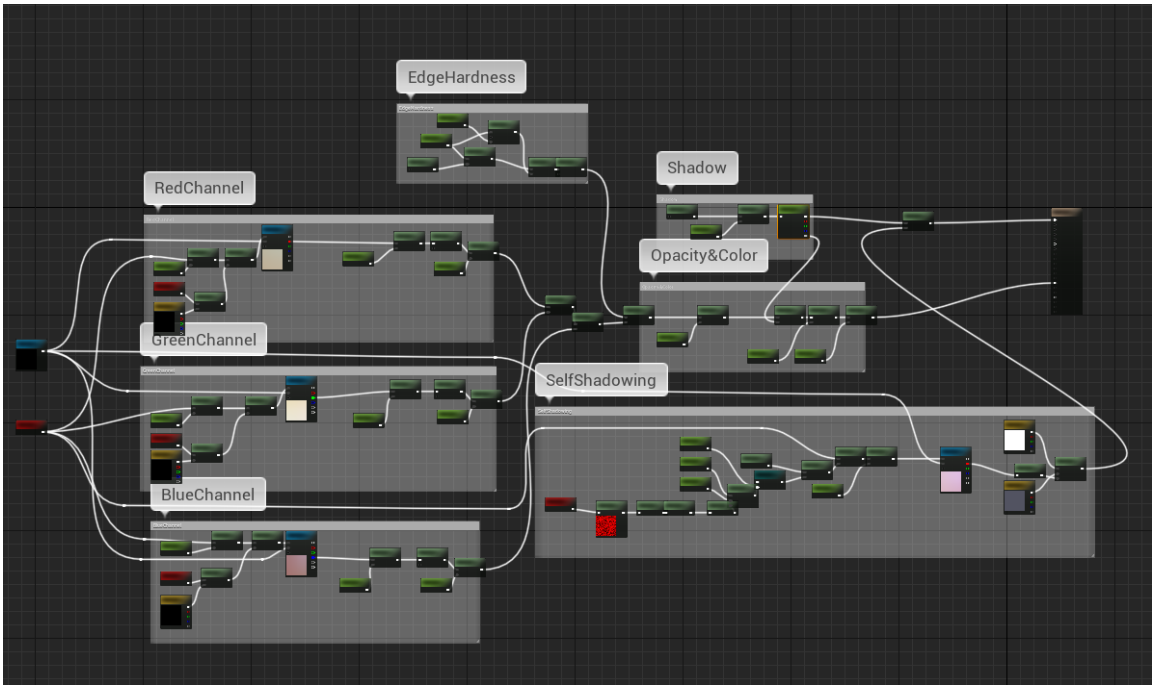


Kuva 39. Asher Zhun mukautettu uudelleenheijastaminen, joka pehmentää sumua entisestään [34].



Kuva 40. Värit luovat illuusion sumun varjostuksesta. Kuvassa myös nähtävissä sumun eri korkeuksien vaikutus sumun käyttäytymiseen.

Yhdistetyt osa-alueet kiinnitetään lopuksi ”Extiction”-, ”Emission”- ja ”BaseColor”-ulostuloihin. Valmis volumetrinen materiaali sisältää alle 200 ryhmiteltyjä piirtokutsuja, joka pitää materiaalin kevyenä jopa mobiilille [49]. Kommenttikentillä voidaan erotella materiaalin eri osia graafin selkeyttämiseksi, kuten kuvassa 41 on nähtävissä. Lisäksi eri osa-alueet on jaettu ryhmiin, jotta parametrit ovat järjestyksessä materiaali-instanssissa.

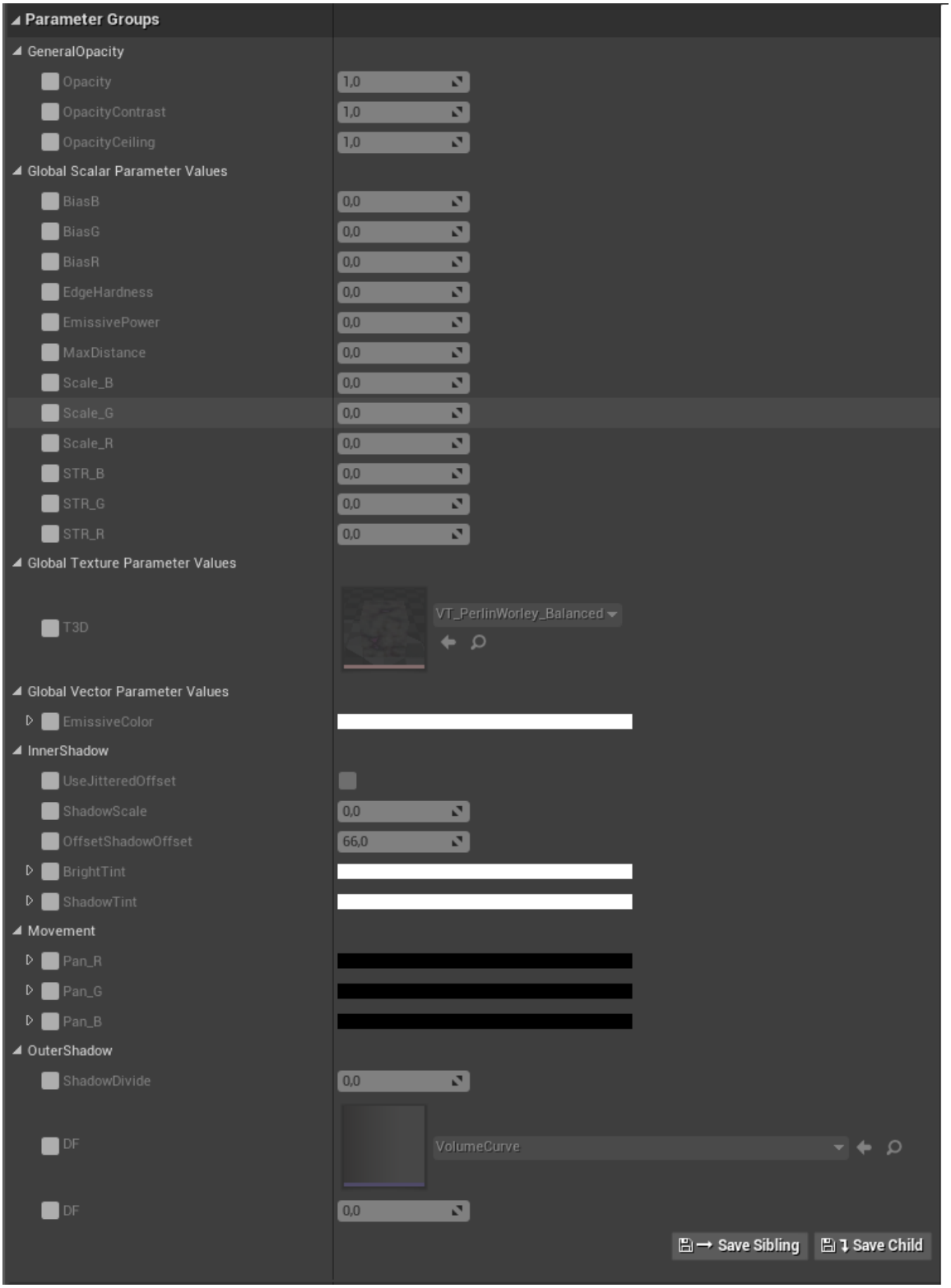


Kuva 41. Valmis volumetrinen materiaali. Kommenttikentillä selkeytetään materiaaligraafin luettavuutta.

Kuvassa 42 on nähtävissä, miten eri parametrit ryhmittyvät instanssissa. Ensimmäisenä tulevat läpinäkyvyyden asetukset, jossa voidaan säätää läpinäkyvyyttä ja sen kontrastia. Sen jälkeen voidaan määritellä jokaisen kanavan vääristymät, vahvuudet ja skaalat. Vääristymien ja skaalojen arvoja säätäessä voi huomata, miten molemmat parametrit käyttäytyvät samankaltaisesti, mutta niissä on huomattava ero: Vääristymä hävittää sumua reunoja myöten, näin pienentäen sumualueita, kun taas skaala vaikuttaa käytössä olevan volyymitekstuurin skaalaan. Pientä paikallista sumualuetta luodessa on skaalojen hyvä olla hyvin pieniä (0,001). Tämä johtuu siitä, että sumu käyttää absoluuttista maailman sijaintia ja ottaa skaalat ympäristön mukaan.

Tästä ryhmästä löytyvät myös etäisyyden parametrien säädöt, jotka ovat maksimietäisyys lähimmästä pinnasta sekä sumun reunojen pehmennys. Materiaalissa on mukana efektin emission värin ja vahvuuden säätömahdollisuus, jotka lisättiin myöhemmin. Emission tarvetta katsotaan myöhemmin LUPEOS-projektin esimerkissä.

Toisessa ryhmässä on mahdollista vaihtaa volyymitekstuuri, mikäli halutaan käyttää toista kohinatekstuuria. Kohinatekstuurilla voidaan muokata sumun muotoja. Sekä sisäisille- että ulkoisille varjoille löytyy niin skaalan kuin siirtymän säätömahdollisuudet. Efektin liikkeen nopeutta voidaan määritellä vektoriparametreillä. Ulkoisten varjojen värien ja läpinäkyvyyden säätö tapahtuu kurvilla. Materiaalia rakentaessa on tärkeää pitää graafi selkeänä ja nimetä parametrit tunnistettavasti, että instanssin käyttö on sujuvaa myös käyttäjillä.



Kuva 42. Materiaali-instanssin parametrit ryhmiteltynä.

5.2 Projekteihin implementointi

5.2.1 LUPEOS

Ensimmäinen tarve efektille oli LUPEOS-projektin talvikentässä, jossa sumulla haluttiin tuoda tunnelmaa puoliksi jäätyneelle joelle. Pakkasen aiheuttama sulan veden höyryäminen luotiin volumetrista efektiä käyttämällä.

Joki laski suoraan lievässä kulmassa, joten efektille tarvittiin sopivan muotoinen kuutio, joka myötäili rinteen laskua. Kun kohta oli mitattu, mallinnettiin Blender-mallinnusohjelmassa sopivan muotoinen kuutio ja asetettiin paikoilleen kenttään (Kuva 43). Kuution korkeudella ei ollut merkitystä, koska projektissa käytettiin etäisyyskenttien laskentaa ja pystyttiin määrittelemään sumun korkeuden lähimmästä pinnasta materiaali-instanssin kautta. Sumun tuli ottaa laskenta ainoastaan veden pinnasta, joten sillalta ja lumipenkereiltä poistettiin mahdollisuus vaikuttaa etäisyyskenttiin.

Kuva 43. Kuutio joelle aseteltuna. Rinteen laskukulma oli lievä, mutta vaati kustomoidun kuution.

Sumu laitettiin liikkumaan joen myötäisesti, hieman ylöspäin hajoten, ja asetusten säätämisen jälkeen havaittiin sumun olevan liian tummaa. Ongelma johtui suuntavalon puutteesta. Materi-

aali käyttää pääasiassa suuntavalon ja atmosfäärin hajontaa, joten niiden pienet arvot eivät riittäneet valaisemaan efektiä riittävästi, aiheuttaen lähes mustaa utua. Tästä johtuen materiaaliin lisättiin mahdollisuus emissioarvon säätöön, jolla pystytään valaisemaan efekti ilman erillistä valonlähdettä. Kuvassa 44 nähtävissä lopputulos.

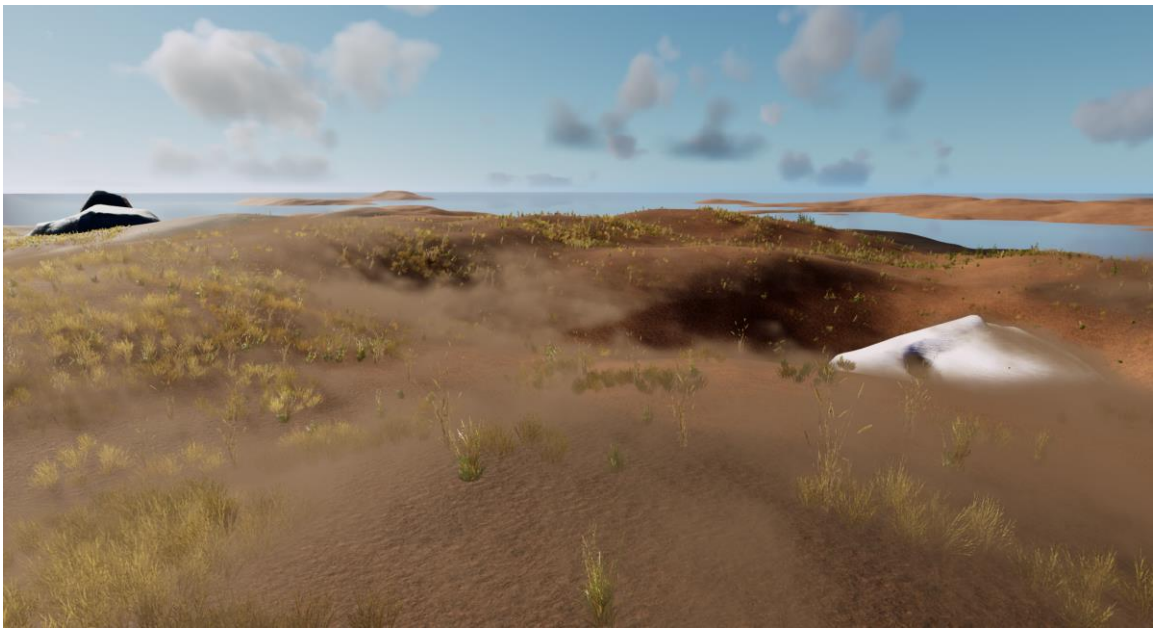
Kuva 44. Valmis volumetrinen sumu joen pinnalla. Utu liikkuu hitaasti ylöspäin, hälveten pienemmiksi osiksi.

5.2.2 Rokua GeoPark

Projekti sisälsi virtuaalisesti toteutettuja alueita Rokuan Geoparkista eri aikakausilta, joista lopuksi renderöitiin 360-videot virtuaalilaseille. Volumetristä efektiä käytettiin mallintamaan lumimyrskyä jääkaudella ja hiekan kulkeutumista luodolla. Koska lopullinen formaatti oli renderöity video, pystyttiin käyttämään efektejä korkeammalla resoluutioilla ja asettelemaan useampia efektejä limittäin. Limittäisyyttä ei yleensä suosita vokselipohjaisissa ratkaisuissa, koska vokselien ylipiirto moninkertaistaa materiaalilaskujen määrän ja efekti muuttuu nopeasti hyvin kalliiksi [50].

Ensimmäinen ongelma havaittiin volumetrisissä pilvissä virtuaalilasien testausvaiheen aikana. Kävi ilmi, että pilvet renderöityvät vain toiselle silmälle. Tämä johtunee siitä, ettei volumetrisille pilville ollut vielä mobiilitukea pelimoottorin versiossa 4.26. Ratkaisuna renderöitiin virtuaalista kameraa käyttämällä pilvistä staattiset tekstuurit, joita käytettiin taivaskomponentissa, sillä pilvien liike ei ollut tärkeä elementti lyhyessä simulaatiossa.

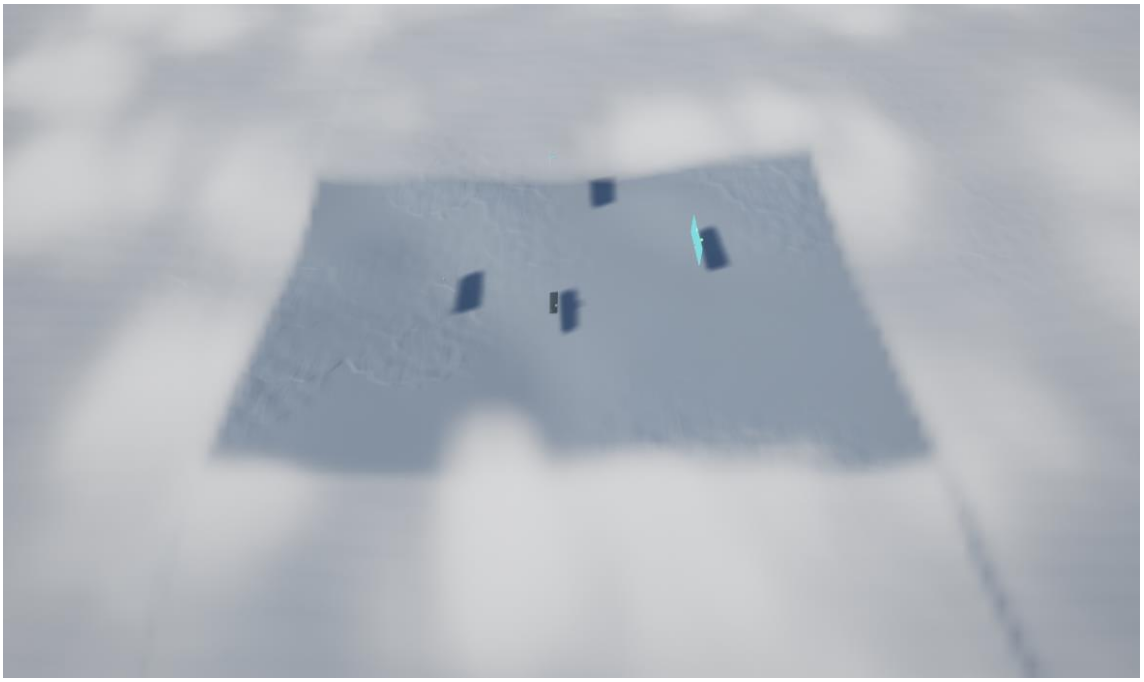
Toinen ongelma tuli ilmi 360-videoiden renderöinnin jälkeen. Mikäli pelaaja seisoi volumetrisen polygoniverkon sisällä, efektin laskostuminen on nähtävissä. Lisäksi heittovarjojen artefaktit olivat selkeästi erotettavissa laskostumisen johdosta. Ratkaisuna poistettiin paikallisen efektin heittovarjot ja siirrettiin efektiä reunustamaan pelaajalle määriteltyä aluetta, jonka ulkopuolelle pelaaja ei voinut siirtyä. Kuvassa 45 pelaaja vielä efektialueen sisällä, jolloin hiekan juoksutusefekti tulee hyvin lähelle pelaajaa.



Kuva 45. Hiekkaefekti luodolla. Panorointinopeus on määritelty suureksi, jolloin saadaan efekti hiekan kulkeutumisesta sivusuunnassa.

Todettiin, että tekstuurin viipalointi ei toimi 360-renderöinneissä, jos efekti limittyy kameran kanssa, koska kameran frustrumi eli piirtoalue on liian laaja ja yrittää piirtää efektin useasta suunnasta samanaikaisesti. Kun efektiä siirsi tarpeeksi kauas kamerasta, frustrumin kulma pieneni tarpeeksi ja frokselit suuntautuivat oikein. Näiden lisäksi hiekan juoksunopeutta säädettiin lopullisessa versiossa paljon hitaammalle, koska virtuaalilaseilla nopea liike aiheutti helposti pahoinvointia. Normaalissa pelitilassa laskostumista ei kuitenkaan ollut havaittavissa ja frokseleiden suuntaus toimii virtuaalilaseille oikein, jolloin vika oli ainoastaan 360-renderöinnissä.

Lumimyrskykentässä samaa efektiä käytettiin melkein samoilla asetuksilla. Ainoastaan efektin maksimietäisyys maasta säädettiin korkeammalle ja väri vaihdettiin valkoiseksi. Laskostuminen oli havaittavissa myös lumimyrskyissä, jolloin efektit siirrettiin taas reunustamaan pelialuetta (kuva 46).



Kuva 46. Lumimyrsky. Ylhäältä päin katsoessa efektin rajat näkyvissä pelaajalle määritellyn alueen reunoilla.

5.2.3 Deanuleagis Sámástit

Viimeisenä efekti käyttöönotettiin saamen kielen opetuspeliin, joka eroaa aiemmista projekteista sen tyylitellyllä grafiikalla. Paikallinen sumu otettiin käyttöön luomaan tunnelmaa ja taianomaisia efektejä suo- ja lähdekentissä. Erona aiempiin projekteihin efektin implementoivat muut projektin työntekijät. Kävi ilmi, että mikäli Unreal Enginen käytöstä on aiempaa kokemusta, pystyi efektiä säätämään sopivaksi testaamalla eri parametreja. Materiaalin implementointi onnistui helposti ja sillä oli mahdollista luoda efektejä erilaisiin ympäristöihin. Sumu ei vienyt liikaa millisekunteja eikä vaikuttanut suorituskykyyn liikaa. Näin ollen voidaan todeta, että materiaali toimi odotetulla tavalla ja käytön tulisi olla helppoa tulevaisuudessakin.

Projektissa hyödynnetään paljon Unreal Enginen tarjoamia volumetrisiä ratkaisuja, kuten pilviä ja uutta taivaan atmosfääriä. Koska taivaan atmosfääri hallitsee niin korkeussumuja kuin myös taivasta, tämän komponentin asetuksilla pystyttiin luomaan yhtenäinen kokonaisuus eri efekteistä. Suuntavalon sijainnilla on suuri merkitys sumujen käyttäytymiseen, koska sillä voidaan luoda sumuun valonsäteitä ja varjoja, kuten kuvassa 47 on nähtävissä.

Kuva 47. Efekti käytössä Staalo-hirviön kentässä muiden sumuefektien lisänä. Kuvassa nähtävissä eksponentiaalisen korkeussumun, volumetrisen sumun ja volumetristen pilvien toiminta yhdessä. Suuntavalo luo volumetriseen sumuun puiden varjoja.

6 Pohdinta

Säteenseurannan ja uuden tehokkaamman teknologian myötä volumetristen efektien tekniikat kehittyvät. Sumuja ei enää käytetä pelkästään staattisina objekteina, vaan myös interaktiivinen sumu ja vesisimulaatiot tulevat yleistymään. Epäsuorat tekniikat väistyvät yhä enemmän suorien tieltä, kun suorituskyky ei ole ongelma pisteitä laskiessa.

Uusien grafiikkaprosessoreiden myötä pelit ovat alkaneet implementoida yhä enemmän säteenseurannan tekniikoita. Esimerkiksi aiemmin mainittu tilavuuden polunseuranta tuo lisää mahdollisuuksia. Jo opinnäytetyön kirjoittamisen aikana tekniikat kehittyivät ja erilaiset tuet volumetrisille efekteille lisääntyivät. On mahdollista, että Unreal Engine 5:n uusien Lumen-valaistusjärjestelmän ja Nanite-renderöinnin myötä volumetristen efektien tekniikat tulevat muuttumaan ja ainakin varmasti kehittymään tulevaisuudessa.

Opinnäytetyötä kirjottaessa opin paljon erilaisista tekniikoista. Perehdyin niin säteenseurannan eri muotoihin kuin renderöintitapoihin ja vektorimatematiikkaan ensimmäistä kertaa. Clever Simulation Entertainmentille tehty materiaali antoi sopivasti haastetta hyödyntää opittuja tietoja ja taitoja. Sumujen rakentaminen ei loppujen lopuksi ole kovin monimutkaista, joten mahdollisimman helppokäyttöisen ja modulaarisen materiaalin luominen vaati tarkempaa perehtymistä erilaisiin tekotapoihin. Lisäksi materiaalia luodessa tuli ottaa huomioon instanssin käytettävyys sellaisen ihmisen näkökulmasta, joka ei aiheeseen ole perehtynyt.

Haastavin osuus tämän työn kirjoittamisessa oli lähteiden löytäminen, sillä moni volumetristen efektien tutkimuksista ovat vuosikymmenten takaa, ja niitä on mahdollista soveltaa monella eri tavalla. Lisäksi minulla ei ollut aiempaa kokemusta säteenseurannan tekniikoista tai valon käsittelystä digitaalisesti, joten opiskelin montaa eri aihealuetta työtä kirjoittaessani. Olen kuitenkin varma, että kaikesta oppimastani on varmasti hyötyä pelinkehitystyössä tulevaisuudessa.

7 Yhteenveto

Tämän opinnäytetyön tavoite oli kattava yleiskuva volumetrisistä efekteistä ja niiden käytöstä Unreal Enginessä. Teoriaosuudessa tutustuttiin volumetristen efektien historiaan elokuvateollisuudessa ja käyttötarkoituksiin lääketieteessä. Ennen volumetrisiä efektejä puhuttiin erilaisista sumuefekteistä. Ensimmäinen tietokonegrafiikan algoritmi, marssivat kuutiot, kehitettiin muodostamaan 3D-esitys magneetti- ja tietokonekerroskuvantamisesta saadusta datasta. Koska pelien volumetriset efektit ovat pohjimmiltaan valon hajontaa tilavuudessa, tutkittiin, kuinka valon aallonpituus suhteessa partikkelien kokoon aiheuttaa sen, että valo nähdään useina eri väreinä. Mitä lyhyempi aallonpituus, sitä sinisempänä valo näkyy.

Opinnäytetyössä tarkasteltiin erilaisia tekniikoita, niiden eroavaisuuksia käsitellä dataa ja esittää se eri muodoissa, kuten polygoniverkkoina tai 3D-tekstuureina. Tultiin tulokseen, että vaikka epäsuorat tekniikat ovat kevyitä, niiden kuvanlaatu ei ole yhtä tarkka kuin suorissa tekniikoissa, koska polygoniverkkojen luominen saattaa aiheuttaa artefakteja. Suorien tekniikoiden etuna taas on reaaliaikaisen renderöinnin mahdollisuus ja parempi laatu. Suorien tekniikoiden implementointi on haastavampaa, koska grafiikkasuorittimet eivät kykene käsittelemään 3D-datan laskentaa yhtä tehokkaasti kuin kolmioiden. Kuitenkin nykyään suorat tekniikat ovat yleisemmin käytössä, koska teknologian kehittymisen myötä pullonkaulat eivät ole enää niin suuri ongelma.

Käytännön osuudessa rakennettiin Unreal Enginen ympäristöön volumetriset efektit ja tutkittiin niiden käyttäytymistä. Unreal Engine sisältää pääasiassa kolme eri komponenttia volumetristen efektien hallintaan: atmosfäärinen sumu, taivaan atmosfääri ja eksponentiaalinen korkeussumu. Komponentit vaikuttavat eri tavoin kentän visuaaliseen ilmeeseen. Atmosfäärinen sumu ja taivaan atmosfääri mallintavat valon hajontaa ilmakehässä, ja eksponentiaalinen korkeussumu mallintaa sumun käyttäytymistä vaihtelevassa maastossa. Kaikki laskevat valon hajontaa hieman eri tarkoitukseen, jolloin on tärkeää tietää, mihin tarkoitukseen efektiä käytetään.

Eksponentiaalisen korkeussumun kautta päädyttiin lisäämään kenttään paikallinen volumetrinen sumu. Paikallinen sumu hyödyntää eksponentiaalisen korkeussumun asetuksia, mutta tarvitsee oman polygoniverkkonsa. Paikallinen sumu on materiaali, joka voidaan asettaa kenttään käyttämällä primitiivisiä muotoja tai partikkelijärjestelmää. Huomattiin, kuinka volumetrisen sumun näköetäisyys vaikuttaa efektin ulkonäköön, sillä Unreal Enginen vokselipohjaiset efektit ovat perus-

asetuksilla matalaresoluutioisia optimoinnin vuoksi. Efektiä voi tarkentaa joko vähentämällä näköetäisyyttä tai nostamalla resoluutiota. Tulee kuitenkin ottaa huomioon, että resoluution nostamisella on suuri vaikutus suorituskykyyn.

Käytännönosuuden lopuksi tehtiin valmis modulaarinen volumetrinen efekti Kajaanin Ammattikorkeakoulun yhteydessä toimivalle Clever Simulation Entertainmentille. Efektin materiaaliin lisättiin mahdollisuus säätää lukuisia arvoja, kuten läpinäkyvyyttä, skaalaa, vääristymiä, nopeutta, värejä ja varjoja. Efekti pystyy seuraamaan eri pintojen muotoja ja reagoimaan valonlähteeseen.

Efekti lisättiin kolmeen erilaiseen projektiin, joissa sitä korjattiin ja tarkasteltiin huomioon otettavia teknisiä rajoitteita. Efektiin lisättiin emissio, jotta se olisi näkyvämpi myös niukasti valaistuissa ympäristöissä. Volumetriset sumut toimivat myös virtuaaliympäristöissä, pois lukien uusi pilvikomponentti. Paikallisia sumuja luodessa tuli ottaa huomioon, ettei kamera limity tilavuuden kanssa, mikäli ympäristöstä oli tarkoitus renderöidä 360-videota, sillä tilavuuden frokselit eivät osaa silloin suuntautua oikein. Sumuefektiä käytetiin myös tyylitellyssä Saami-projektissa, jossa sillä luotiin tunnelmaa niin suolle kuin lähteelle.

Volumetriset efektit ovat nykyään helposti implementoitavissa, mutta mikäli ei ymmärrä efektien toiminnan periaatteita, on helppo kuormittaa peliä turhaan. Efektien optimointi on loppujen lopuksi vaivatonta, jos tietää mitä tekee. Efektit eivät välttämättä toimi kaikissa ympäristöissä, vaan efektin tyyppi ja tapa käsitellä dataa vaikuttavat lopputulokseen.

Jo tämän opinnäytetyön kirjoittamisen aikana tuet ja dokumentaatiot volumetrisille efekteille lisääntyivät ja tekniikat kehittyvät, ja monet työssä esitellyt tekniikat varmasti vanhenevat yllättävän nopeasti. Pystyäkseen oppimaan tulevaisuudessa uusia tekniikoita, on hyvä ymmärtää, mihin kaikki tekniikat loppujen lopuksi pohjautuvat: valon fysiikkaan ja sen ilmenemiseen erilaisissa tilanteissa, digitaaliseen muotoon käännettynä.

Lähteet

1 Gampa D., Kennedy D. 3d-dictionary [verkkolähde]. Luettu: 6.10.2020. Saatavissa: http://www.tweak3d.net/3ddictionary/

2 Green S. Nvidia Corporation: GDC 2005, NVIDIA Developer. San Francisco. 2005. Luettu: 11.10.2020 Saatavissa: http://http.download.nvidia.com/developer/presentations/2005/GDC/Sponsored_Day/GDC_2005_VolumeRenderingForGames.pdf

3 Ikits. M., Kniss J., Lefohn A. Hansen C. Volume Rendering Techniques. Teoksessa: GPU Gems [e-kirja]. Painos 5. Pearson: 2004. Luettu: 19.10.2020. Saatavissa: https://developer.nvidia.com/gpugems/gpugems/part-vi-beyond-triangles/chapter-39-volume-rendering-techniques

4 Wikipedia: Sähkömagneettinen säteily: Sähkömagneettisen säteilyn spektri. [Kuvalähde]. Viitattu 19.10.2020. Saatavissa: https://upload.wikimedia.org/wikipedia/commons/thumb/0/07/EM_spectrum_fi.svg/2560px-EM_spectrum_fi.svg.png

5 Aalto yliopisto. Sähkömagneettinen säteily: Luento 1 [verkkolähde]. Luettu: 19.10.2020. Saatavissa: https://foto.aalto.fi/opetus/350/k01/luento1/sms.html

6 Wallenius J. Turun Sanomat: Miksi taivas sininen niin… [verkkolähde]. 2010. Luettu: 19.10.2020 Saatavissa: https://www.ts.fi/teemat/145929/Miksi+taivas+on+sininen+niin

7 Kroll N. Premium Beat: Cinematography tip: Use fog to add depth to Your shot [verkkolähde]. 2015. Luettu: 19.10.2020 Saatavissa: https://www.premiumbeat.com/blog/cinematography-tip-use-fog-to-add-depth-to-your-shot/

8 Alcon Lightning: What is volumetric lighting design? [verkkolähde]. 2019. Luettu: 22.9.2020. Saatavissa: https://www.alconlighting.com/blog/lighting-design/volumetric-lighting-lighting-design/

9 Szczepanek P. Review of Real-time volume rendering techniques. Automatyki, Informatyki i Elektorniki, Krakow. 2007. Luettu: 11.10.2020. Saatavissa: http://teatime-coder.com/file_download/53/real-time+volume+rendering.pdf

10 Godzilla II: King of Monsters. 2019. [Elokuva]. Michael Dougherty. Ohj. Yhdysvallat. Legendary Pictures. Viitattu 14.10.2021.

11 Wendigo Carnage. 2014. [Elokuva]. James Cawley. Ohj. Viitattu 14.10.2021.

12 Alexandrov G. Creative Shrimp tutorials: 4 Reasons why all-devouring fog is actually an amazing thing [verkkolähde]. 2015. Luettu: 13.10.2020. Saatavissa: https://www.creativeshrimp.com/fog-tutorial-lighting-book-11.html

13 Lorensen W., Cline H. SIGGRAPH Computer Graphics. Marching Cubes: A high resolution 3d surface construction algorithm [artikkeli]. Berkeley, University of California. 1987;21(4): 163-169. Luettu: 13.10.2020. doi: 10.1145/37402.37422

14 OmniSci: Volume Rendering Definition. Luettu: 5.10.2020. Saatavissa: https://www.omnisci.com/technical-glossary/volume-rendering

15 Rockstar games: Red dead redemption 2. [Kuvalähde]. Viitattu 19.10.2020. Saatavissa: https://www.rockstargames.com/reddeadredemption2/screens

16 Meißner, M & Pfister, Hanspeter & Westermann, R & Wittenbrink, Craig. Volume visualization and volume rendering techniques. [verkkolähde]. 2000. Luettu: 4.11.2021. Saatavissa: https://www.researchgate.net/publication/228557366_Volume_visualization_and_volume_rendering_techniques

17 Computer Graphics Laboratory (CGL). Stuttgart Visualization Course: Volume visualization [luentomateriaali]. 2006. Luettu: 5.10.2020. Saatavissa: https://cgl.ethz.ch/teaching/former/scivis_07/Notes/stuff/StuttgartCourse/VIS-Modules-05-Volume_Visualization.pdf

18 Nikolov S. Coherent Labs. Overview of modern volume rendering techniques – Part 2 [verkkolähde]. 2013. Luettu: 14.9.2020. Saatavissa: https://coherent-labs.com/posts/overview-of-modern-volume-rendering-techniques-for-games-part-ii/

19 Crassin C., Neyret F., Lefebvre S., Eisemann E. Maverick: GigaVoxels: Ray-guided streaming for efficient and detailed voxel rendering. Teoksessa: In proceedings of the 2009 symposium on interactive 3d graphics and games (I3D) [e-kirja]. Association for

computing machinery, New York, 2009: 15-22. Luettu 14.9.2020. doi: 10.1145/1507149.1507152

20 Computer Graphics Laboratory (CGL). Stuttgart Visualization Course: Direct volume rendering [luentomateriaali]. 2006. Luettu: 12.10.2020. Saatavissa: https://cgl.ethz.ch/teaching/former/scivis_07/Notes/stuff/StuttgartCourse/VIS-Modules-06-Direct_Volume_Rendering.pdf

21 Lafortune E.P., Willems Y.D. Rendering Participating Media with Bidirectional Path Tracing. In: Pueyo X., Schröder P. (eds) Rendering Techniques '96. EGSR 1996. Eurographics. Springer, Vienna, 1996. doi: 10.1007/978-3-7091-7484-5_10

22 Hearn P. OnlineTechTips: What is Path Tracing and Ray Tracing? And Why do They Improve Graphics? 2019. Luettu: 15.11.2020. Saatavissa: https://www.online-tech-tips.com/computer-tips/what-is-path-tracing-and-ray-tracing-and-why-do-they-improve-graphics/

23 Brucks R. Shader Bits: Creating a volumetric ray marcher [verkkolähde]. 2016. Luettu: 6.10.2020. Saatavissa: https://shaderbits.com/blog/creating-volumetric-ray-marcher

24 Brucks R. Shader Bits: Ray marched heightmaps [verkkolähde]. 2016. Luettu: 6.10.2020. Saatavissa: https://shaderbits.com/blog/ray-marched-heightmaps

25 Ertl T., Udiger Westermann R. Solid texturing on a per-pixel basis [verkkolähde]. Teoksessa: Proceedings of Image and Multidimensional Digital Signal Processing (IMDSP). 1998, 98: 291-294. Luettu: 13.10.2020. Saatavissa: https://www.in.tum.de/fileadmin/w00bws/cg/Research/Publications/1998/imdsp98.pdf

26 Botha C., Preim B. Virtual Endoscopy [e-kirja]. Teoksessa: Visual computing for medicine. Painos 2. Morgan Kaufmann: 2014: 509-536. Luettu: 12.10.2020. doi: 10.1016/B978-0-12-415873-3.00013-4

27 Fisher M. Matt's Webcorner: Marching cubes. Standford, 2014. Luettu: 13.10.2020. Saatavissa: https://graphics.stanford.edu/~mdfisher/MarchingCubes.html

28 Lague S. Coding Adventure: Marching cubes [videolähde]. 2019. Katsottu: 13.10.2020. Saatavissa: https://www.youtube.com/watch?v=M3iI2l0ltbE

29 Marching cubes: The originally published 15 cube configurations. Viitattu 13.10.2020. Saatavissa. https://en.wikipedia.org/wiki/Marching_cubes#/media/File:MarchingCubes.svg

30 Strickland D. Tweak Town: Amazing Unreal Engine 4 volumetric lighting teases next-gen fog [verkkolähde]. 2020. Luettu: 20.10.2020. Saatavissa: https://www.tweaktown.com/news/71886/amazing-unreal-engine-4-volumetric-lighting-teases-next-gen-fog/index.html

31 Unreal Engine documentation: Volumetric fog [verkkolähde]. Luettu: 7.9.2020. Saatavissa: https://docs.unrealengine.com/en-US/Engine/Rendering/LightingAndShadows/VolumetricFog/index.html

32 Unreal Engine 4 documentation: Sky Atmosphere [verkkolähde]. Luettu: 24.2.2021. Saatavissa: https://docs.unrealengine.com/4.27/en-US/BuildingWorlds/FogEffects/SkyAtmosphere/

33 Smith R. Overdraw: Working with volume textures in Unreal Engine 4 [verkkolähde]. 2019. Luettu: 13.10.2020. Saatavissa: https://www.overdraw.xyz/blog/2019/11/16/working-with-volume-textures-in-unreal-engine-4

34 Zhu A. Create nice and feasible volumetric cloud in Unreal Engine 4. 2020. Luettu: 13.1.2021. Saatavissa: http://asher.gg/?p=2600

35 Unreal Engine 4 documentation: Volumetric clouds. Luettu: 14.1.2021. Saatavissa: https://docs.unrealengine.com/en-US/BuildingWorlds/LightingAndShadows/VolumetricClouds/index.html

36 De Jong S. Unreal Engine 4 GDC: Volumetric lighting in Unreal Engine 4 [videolähde]. 2018. Katsottu: 13.10.2020. Saatavissa: https://youtu.be/Xd7-rTzfmCo

37 Unreal Engine 4 documentation: Volumetric lightmaps [verkkolähde]. Luettu: 20.10.2020. Saatavissa: https://docs.unrealengine.com/en-US/Engine/Rendering/LightingAndShadows/VolumetricLightmaps/index.html

38 Tieteen termipankki, Lambert-Beerin laki [verkkolähde]. 2015. Luettu: 20.10.2020. Saatavissa: https://tieteentermipankki.fi/wiki/Fysiikka:Lambert-Beerin_laki

39 Clark J. The Beer-Lambert Law [verkkolähde]. Truro School, Cornwall, 2020 Luettu: 3.11.2020. Saatavissa: https://chem.libretexts.org/@go/page/3747

40 Drebin R., Carpenter L., Hanrahan P. SIGGRAPH Computer graphics: Volume rendering [artikkeli]. 1988;22(4): 65-74. Luettu: 20.10.2020. doi: 10.1145/378456.378484

41 Van Oosten J. 3dgep: Forward vs deferred vs forward+ rendering with DirectX 11 [verkkolähde]. 2015. Luettu: 20.10.2020. Saatavissa: https://www.3dgep.com/forward-plus/

42 Unreal Engine documentation: Rendering [verkkolähde]. Luettu: 28.9.2020. Saatavissa: https://docs.unrealengine.com/en-US/Engine/Rendering/Overview/index.html

43 Zhu A. Unreal Engine 4, Inside Unreal series: Expand your world with volumetric effects [videolähde]. 2020. Katsottu: 28.9.2020. Saatavissa: https://www.youtube.com/watch?v=R2RQm_Bu81I&t=1169s

44 Unreal Engine documentation: Volume textures [verkkolähde]. Luettu: 27.9.2020. Saatavissa: https://docs.unrealengine.com/en-US/Engine/Content/Types/Textures/VolumeTextures/index.html

45 Shadertoy: Tileable Perlin-Worley 3d. Luettu: 14.11.2020. Saatavissa: https://www.shadertoy.com/view/3dVXDc

46 Kinsey M. Dinusty: Volumetric Clouds in UE4 Tutorial Content [verkkolähde]. Luettu 20.9.2021. Saatavissa: https://www.artstation.com/marketplace/p/KzYA/dinusty-volumetric-clouds-in-ue4-tutorial-content

47 Unreal Engine 4 documentation: Distance fields. Luettu: 13.1.2021. Saatavissa: https://docs.unrealengine.com/en-US/BuildingWorlds/LightingAndShadows/MeshDistanceFields/index.html

48 Unreal Engine 4 documentation: Sky Atmosphere references [verkkolähde]. Luettu 15.9.2021. Saatavissa: https://docs.unrealengine.com/4.26/en-US/BuildingWorlds/FogEffects/SkyAtmosphere/Reference/

49 Unreal Art Optimization: Unreal's Rendering Passes [verkkolähde]. Luettu 16.9.2021. Saatavissa: https://unrealartoptimization.github.io/book/profiling/passes/

50 Brucks R. UE4 Volumetric Fog Techniques [verkkolähde]. Luettu 21.9.2021. Saatavissa: https://shaderbits.com/blog/ue4-volumetric-fog-techniques