KARELIA-AMMATTIKORKEAKOULU
Tietojenkäsittelyn koulutus

Raimo Rautiainen

Sääolosuhteisiin mukautuvan dynaamisen materiaalin luonti Unreal Engine 4:llä

Opinnäytetyö
Joulukuu 2021

Karelia AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Joulukuu 2021
Tietojenkäsittelyn koulutus

Tikkarinne 9
80200 JOENSUU
+358 13 260 600 (vaihde)

Tekijä(t)
Raimo Rautiainen

Nimeke
Sääolosuhteisiin mukautuvan dynaamisen materiaalin luonti Unreal Engine 4:llä

Toimeksiantaja
-

Tiivistelmä

Opinnäytetyössä tutkittiin pelimaailmojen elävöittämistä etsimällä dynaamisia tapoja mukauttaa peliobjektien ulkonäköä. Tavoitteena oli löytää vaihtoehtoja peliobjektien käyttämien materiaalien pelinaikaiseen muokkaukseen siten, että pelimaailmoista pystyttäisiin luomaan uudenoloisia.

Pelimaailman elävöittämiseksi suunniteltiin ja toteutettiin sääolosuhteisiin mukautuvia materiaaleja. Työn toteutuksessa käytettiin Unreal Engine 4 -pelimoottoria, joka tarjosi useita mahdollisia tapoja pelinaikaisen dynaamisuuden ratkaisemiseen. Tarvittavien sääolosuhde-efektien määrä ja monimutkaisuus arvioitiin kehityksen alussa. Lopuksi toteutetut sääolosuhde-efektit muunnettiin funktioiksi ja kytkettiin niitä kontrolloivaan ohjaajaluokkaan.

Opinnäytetyön aikana onnistuttiin toteuttamaan toimiva sääolosuhteita simuloiva materiaali. Toteutuksen aikana onnistuttiin myös perehtymään eri tapoihin luoda mukautuvuutta materiaaleihin. Toteutettu järjestelmä uudisti pelimaailmaa ja saavutti sille asetetut tavoitteet. Opinnäytetyön lopuksi todettiin, että dynaamiset materiaalit voivat antaa pelimaailmoille jatkuvasti muuttuvan tunteen ja lisäävät peliobjektien uudelleenkäyttömahdollisuuksia.

Kieli
suomi

Sivuja 31
Liitteet 2
Liitesivumäärä 7

Asiasanat
pelikehitys, Unreal Engine 4, materiaalit, dynaamiset materiaali-instanssit

THESIS
December 2021
Degree Programme in Business Information Technology

Tikkarinne 9
80200 JOENSUU
FINLAND
+ 358 13 260 600 (switchboard)

Author (s)
Raimo Rautiainen

Title
The Creation of Dynamic Weather Materials with Unreal Engine 4

Commissioned by
-

Abstract

The goal of the thesis project was to research ways to make game worlds feel renewed with the help of weather effects. The project focuses on looking into the options of creating dynamic changes in the appearances of game objects.

The renewability of game worlds was carried out by designing and developing dynamic weather materials. The development was done with Unreal Engine 4 game engine which offered multiple ways to make changes to materials during gameplay. During development, the number of required weather effects was estimated along with their required complexity. Later on, these effects were turned into functions and connected into a controller system.

The project resulted in creating a dynamic weather material. The possible options for creating dynamic changes to materials were also researched thoroughly. In conclusion, dynamically changing materials can give the game world an ever-changing feeling and increase the reusability of game assets.

Language
Finnish

Pages 31
Appendices 2
Pages of Appendices 7

Keywords
game development, Unreal Engine 4, materials, dynamic material instances

Sisältö

Liitteet

Erikoissanasto

Node	Kaaviokäyttöliittymässä käytettävä koodi-ilmaisu.
Tekstuurikartta	Tekstuurikoordinaatteihin liitettävä kuva, joka liitetään peliobjekteihin.
Render Target	Peliajan kuluessa muokattavissa oleva tekstuurikartta.
Blueprint	Kaaviokäyttöliittymäpohjainen ohjelmointikieli Unreal Engine 4 -pelimoottorissa.
Base Color	Materiaalin värin asettava arvo tai tekstuurikartta.
Normal	Materiaalin pinnanmuodonvaihteluja jäljittelevä tekstuurikartta.
Roughness	Materiaalin kiiltävyyden asettava arvo tai tekstuurikartta.
World Position Offset	Peliobjektin sijainnin ja koon muokkauksen mahdollistava materiaalinominaisuus.
WorldAlignedTexture	Tekstuurikoordinaattien kartoittamiseen käytetty materiaalifunktio.
Lerp	Alpha-arvoon perustava kahden arvon väliltä valitseva node.
Alphamaski, maski	Läpinäkyvyyttä kontrolloiva tekstuurikartta.
Alpha	Läpinäkyvyyttä kontrolloiva yksittäinen arvo.
Tekstuurikoordinaatit	Kaksikanavainen koordinaatisto, jonka avulla tekstuurikartat liitetään peliobjekteihin.
Panner	Tekstuurikoordinaatteja animoiva node.
TransformVector	Suhteuttaa sille syötetyn vektoriarvon eri koordinaatistoon.
Sine	Sinifunktio
Floor	Node, joka pyöristää arvon alaspäin seuraavaan täyslukuun.
Noise-funktio	Satunnaisen noise-tekstuurikartan luova funktio.
CollectionParameter	Materiaaliparametrikokoelmaa viittaava node.

1 Johdanto

Opinnäytetyönaiheena on dynaamisten materiaalien luonti Unreal Engine 4 -pelimoottorilla. Aiheen mukana sukelletaan mahdollisiin tapoihin elävöittää pelimaailmoja, tavoitellen käytännöllistä ja kontrolloitavissa olevaa lähestymistapaa sääolosuhteiden esittämiseen. Työn päätavoitteena on toteuttaa järjestelmä, jonka avulla voidaan luoda pelimaailmoihin mukautuvuutta, hyödyntämällä pelimoottorin tarjoamaa viitekehystä dynaamisten materiaalien luontiin. Päätavoitteen ohella työssä sivuutetaan pelimaailmojen interaktiivisuuden lisäämistä, antamalla pelaajille enemmän vuorovaikutusmahdollisuuksia dynaamisten materiaalien avulla.

Järjestelmän luonnin yhteydessä päästään oppimaan materiaaliefektien toteutuksesta ja niissä käytettävistä tekniikoista. Tavoitteena on hakea optimaalisia ratkaisuja erilaisten materiaaliefektien luontiin ja käyttää niitä hyväksi sääolosuhteita simuloivissa efekteissä. Tämän lisäksi kullekin efektille tavoitellaan parametrisoitua ratkaisua, jota voidaan kontrolloida käyttötapauksen perusteella.

Omat tavoitteeni opinnäytetyötä kohtaan sijoittuvat Unreal Engine 4:n tarjoamiin toiminallisuuksiin syventymiseen ja uuden oppimiseen. Tavoitteenani on oppia siitä, miten materiaalit toimivat ja miten niitä voidaan muokata peliajan kuluessa. Haluan myös ymmärtää materiaalien luontia paremmin, jotta voin paikantaa ja oikaista olettamukseni sitä kohtaan. Aihe vaatii perehtymistä materiaalien luontiin, usean materiaaliefektin yhdistämiseen ja materiaalien mukautuvuuden toteuttamiseen, antaen hyvän mahdollisuuden sille asetettujen tavoitteiden saavuttamiselle.

2 Tietoperusta

2.1 Dynaamisten materiaalien hyödyntäminen peleissä

Dynaamiset materiaalit antavat mahdollisuuden luoda pelinaikaisia visuaalisia muutoksia pelimaailman objekteihin. Dynaamisilla materiaaleilla toteutettavat muutokset voivat vaihdella yksinkertaisesta peliobjektin värin vaihdosta, peliobjektin tuhoamiseen tai sääolosuhteiden esitykseen. Useat pelimoottorit antavat pelinaikaisiin muutoksiin mahdollisuuden, kuten myös Unreal Engine 4, jolla opinnäytetyön kehitystyö toteutettiin.

Murphy (2018) esittelee dynaamisten muutosten tekoa blueprinttien ja materiaalien avulla YouTubeen julkaistulla videolla. Esityksessä hän käyttää Unreal Enginen landscape-järjestelmää ja piirtää sen pinnalle laavaa muistuttavan efektin. Muutokset tehtiin Render Target -tekstuurikartalla, jota hän kuvaa tekstuurikartaksi, jota voidaan muuttaa pelinaikana. Esitelty Render Target -järjestelmä käyttää erillistä sivellin materiaalia muutosten piirtämiseen. Järjestelmän blueprint-luokka hoitaa dynaamisien materiaali-instanssien luonnin ja parametrien asetuksen. Muutokset vastaanottava materiaali käyttää Render Target -tekstuurikarttaa kahden eri materiaalitason väliseen kommunikaatioon. (Murphy 2018)

Unreal Engine 4:n dokumentaatiossa on esimerkki projekti, jossa esitetään peliobjektin pinnan korottamista Render Target -järjestelmän avulla. Render Target -tekstuurikartan päivittämistä hallinnoitiin blueprint-luokalla. Blueprint-luokan määrittäminen koostui kolmesta osasta. Render Target -tekstuurikartan luonnista ja kahden dynaamisen materiaali-instanssin asetuksesta. Ensimmäinen dynaaminen materiaali-instanssi kuului materiaalille, joka aikoo käyttää Render Target -tekstuurikarttaa. Toinen materiaali-instanssi oli sivellinmateriaalille, jonka avulla piirretään luotuun Render Target -tekstuurikarttaan. Sivellinmateriaalissa luotiin pyöreä gradienttiefekti, jossa voidaan muuttaa siveltimen sijaintia, kokoa ja voimakkuutta. Hallinoivaan blueprint-luokkaan tallennettiin muuttujaarvot, joilla kutakin siveltimen ominaisuutta kontrolloitiin. Render Target -

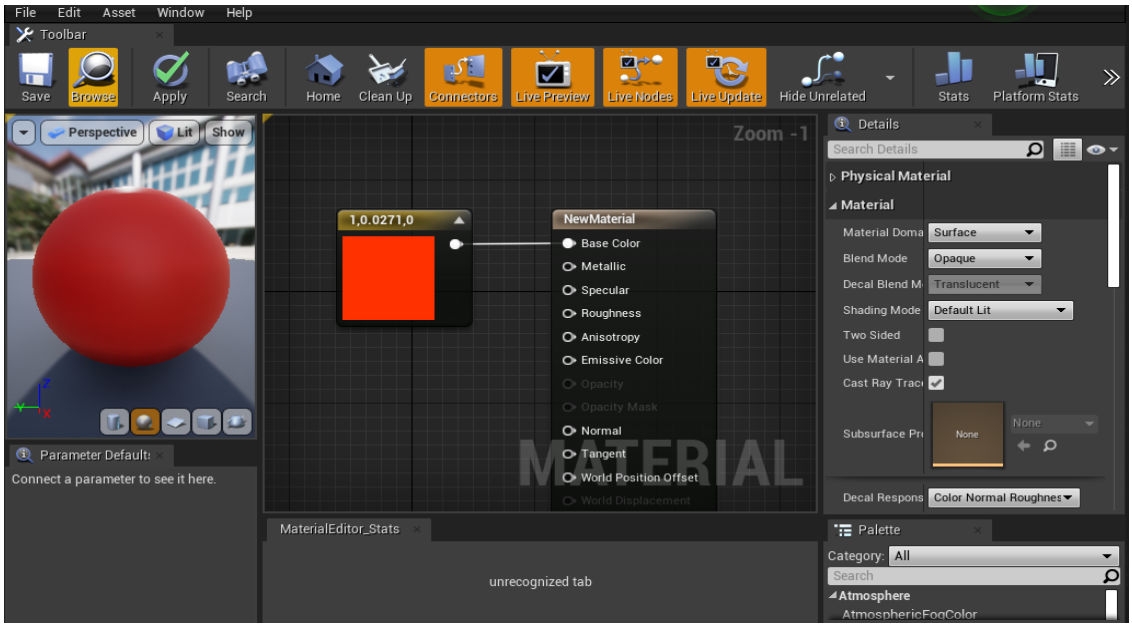
tekstuurikarttaan piirtäminen tapahtui funktiolla, jolla etsittiin materiaalista sijainti, johon Render Target -tekstuurikarttaan haluttiin piirtää. Lopputuloksena oli Render Target -järjestelmä, jolla pystyttiin nostamaan peliobjektin pinnankorkeutta piirtämällä muutoksia materiaalin omaavaan Render Target -tekstuurikarttaan. (Epic Games, Inc. 2021a.)

2.2 Unreal Engine 4 ja materiaalit

Unreal Engine 4 on Epic Gamesin kehittämä pelimoottori. Pelimoottori antaa mahdollisuuden luoda reaaliaikaisia elämyksiä interaktiivisissa 3D-maailmoissa. Yleisimmin sitä käytetään pelikehityksessä, mutta Epic Games on laajentanut mahdollisten käyttötapausten listaa muun muassa elokuvatuotantoon, arkkitehtuurivisuaalisointiin ja moniin muihin 3D-mahdollisuuksista hyötyviin käyttötarkoituksiin. Opinnäytetyön aikana keskitytään Unreal Engine 4:n tarjoamiin työkaluihin luoda ja muokata 3D-maailmoja, joiden pääasiallinen käyttötarkoitus on esiintyä pelimaailmoina. (Epic Games, Inc. 2021b.)

Unreal Engine 4:n tarjoamista työkaluista opinnäytetyölle merkityksellisimmät ovat materiaalit ja materiaalieditori. Materiaaleja voidaan ajatella peliobjekteihin liitettävinä maalipintoina, jotka määrittävät peliobjektien ulkonäön. Esimerkkejä materiaalien ominaisuuksista ovat pinnanväri, -kiiltävyys tai -läpinäkyvyyden määrittäminen. (Epic Games, Inc. 2021c.)

Materiaalieditori puolestaan on node-pohjainen kaaviokäyttöliittymä materiaalien luontiin (kuva 1). Materiaalieditorin avulla materiaaleja voidaan luoda muistilappujen tapaisten node-ilmaisujen avulla. Materiaalieditori antaa mahdollisuuden kaavion kommentointiin ja kaavioliitosten uudelleen reititykseen, jotta tehdyt kaaviot pysyisivät luettavina ja ymmärrettävinä. Materiaalieditorissa on myös muutosten nopea esikatselumahdollisuus ja käännösvirheiden automaattinen havaitsemistoiminto. (Epic Games, Inc. 2021d.)



Kuva 1. Materiaalieditorin käyttöliittymä (Kuva: Raimo Rautiainen).

2.3 Materiaalien ominaisuudet

Node-ilmaisukaavion lisäksi materiaalieditorissa on ominaisuuspaneeli, josta voidaan asettaa materiaalille erilaisia käyttäytymismääritteitä. Määritteiden avulla voidaan esimerkiksi kertoa renderöijälle materiaalin käyttötarkoitus, eli Material Domain. Eri käyttötarkoitusmääritteet antavat renderöijälle tiedon tarvittavasta kääntöohjeistuksen määrästä. Muita käyttäytymismääreittä on huomattavasti, mutta niistä opinnäytetyöhön liittyviä ovat Blend Mode ja Shading Model. (Epic Games, Inc. 2021e.)

Blend Mode on käyttäytymismäärite, joka määrittää miten materiaali piirretään sen takana olevan ympäristön päälle. Opaque Blend Mode määrittää materiaalin pinnan kiinteäksi. Valon ei tarvitse läpäistä kiinteiden pintojen läpi, mikä tarkoittaa, että Opaque-materiaalit piirretään aina sellaiseen muun ympäristön päälle. Seuraava opinnäytetyölle merkittävä määrite on Translucent Blend Mode, jota käytetään läpinäkyvyyden tuottamiseen. Translucent-materiaalit ottavat vastaan läpinäkyvyysarvon tai tekstuurikartan, jonka avulla ne määrittävät läpinäkyvyytensä. Läpinäkyvyysarvo luetaan siten, että tekstuurikartan valkeat osat renderöidään kiinteinä materiaaleina, mustat täysin läpinäkyvinä ja harmaat sävyt ovat niiden väliltä. (Epic Games, Inc. 2021f.)

Seuraavana käyttäytymismääritteenä on Shading Mode, joka asettaa sen, miten materiaali heijastaa siihen osuvaa valoa. Teknisemmin sanottuna, se kertoo renderöijälle, miten materiaalille syötettyjen ohjeiden on tarkoitus luoda sen lopullinen ulkonäkö. Opinnäytetyölle merkityksellisimmät Shading Mode -määritteet ovat Unlit ja Default Lit. Unlit-materiaalit eivät heijasta niihin osuvaa valoa ollenkaan ja ainoastaan säteilevät niille asetettua emissive-parametrin väriä. Unlit on täydellinen vaihtoehto esimerkiksi erikoisefektien tuottamiseen. Seuraava määrite, eli Default Lit, on nimensä mukaan Shading Moden oletusarvo. Default Lit -määritettä käytetään suurimmalle osalle materiaaleista. Default Lit -materiaalit tukevat suoraa ja epäsuoraa valaistusta ja käyttävät Specular-parametri arvoa heijastukseen. (Epic Games, Inc. 2021g.)

2.4 Materiaaliparametrit ja -instanssit

Materiaaliparametrit ovat materiaalien sisällä olevia muuttuja-arvoja, joiden avulla voidaan muokata materiaalien ulkonäköä. Parametrien arvon muokkaus ei edellytä materiaaliluokan uudelleen kääntöä, jolloin muutosten teko on helposti kontrolloitavissa. (Epic Games, Inc. 2021h.)

Materiaali-instanssit puolestaan ovat hierarkiajärjestelmä materiaalien avaruudessa. Ne hyödyntävät materiaaliparametreja ja antavat ohjelmoijalle mahdollisuuden tehdä muokkauksia materiaaleihin nopeasti ja helposti. Instanssoidut materiaalit jakavat ominaisuutensa niiden päämateriaalien kanssa ja sen vuoksi mikä tahansa parametri, joka on päämateriaalissa, voidaan muuntaa myös sen instanssoidussa versiossa. Tällä tavoin voidaan luoda samankaltaisia materiaaleja, joilla on eroja värien tai muiden efektien voimakkuuksien välillä. (Epic Games, Inc. 2021i.)

Parametrit, joita voidaan käyttää voivat vaihdella yksittäisistä skalaariarvoista, neljän pisteen vektoriarvoihin. Tämän lisäksi parametrit voivat olla tekstuurikartta-tyyppisiä, tai jopa boolean arvoja, joita voidaan käyttää portteina sulkemaan ohjelmoitua logiikkaa, jota ei tarvita tietyn instanssin yhteydessä.

Materiaaliparametrikokoelma on assetti, joka tallentaa skalaari- ja vektoriarvoja, joita useat eri materiaalit pystyvät hyödyntämään samaan aikaan. Tämä antaa mahdollisuuden tehdä ns. globaaleja muutoksia useaan materiaaliin samanaikaisesti. (Epic Games, Inc. 2021j.)

2.5 Materiaalifunktiot ja -tasot

Materiaalifunktiot ovat pätkiä materiaalikaaviota, jotka voidaan tallentaa ja käyttää uudestaan eri materiaaleissa. Niiden on tarkoitus tehostaa materiaalien luontia antamalle ohjelmoijalle valikoiman yleisesti käytetyistä node-verkostoista. Tämä antaa myös mahdollisuuden piilottaa tilaa vieviä ja monimutkaisia node-verkostoja yksinkertaisen funktion sisälle. Materiaalifunktioita voi muokata samalla tavalla kuin materiaaleja, mutta niille voi asettaa syöttö- ja tulostusparametrejä, joiden avulla funktion sisällä olevaa materiaaliverkoston virtausta voidaan kontrolloida. (Epic Games, Inc. 2021k.)

Materiaalitasot, ovat materiaalifunktioita, jotka hyödyntävät materiaali attribuuttien tallentamista materiaalifunktion sisälle. Materiaalitasojen avulla voidaan luoda kokonaisia materiaaleja ja tuoda ne mukaan toisiin kaavioihin yhden materiaalifunktio-noden avulla. Materiaalitasot käyttävät niiden luontiin tarkoitettua erillistä materiaalifunktio-assettia. Sen sisällä funktio ottaa vastaan samat syötettävät tiedot kuin normaalit materiaalikaaviot ja sullovat sen datan yhteen tulostusparametriin. (Epic Games, Inc. 2021l.)

3 Menetelmälliset valinnat

3.1 Kehitystyökalun valinta ja perustelut

Kehitystyökaluksi valittiin Unreal Engine 4 -pelimoottorin versio 4.26. Valinta perustui pelimoottorin tarjoamiin menetelmiin materiaalien ja niiden pelinaikaisten

muutosten luontiin. Unreal Engine 4:n tarjoamat menetelmät koettiin riittäviksi tavoitteiden saavuttamiseksi.

Tarjotuista menetelmistä keskeisimmät olivat materiaalieditori ja materiaali-instanssit, mitkä mahdollistivat materiaalien tuotannon ja parametrisoidun testauksen. Näiden lisäksi kehitykseen käytettiin materiaalifunktiota, virtaviivaistamaan parametrisointia ja mahdollistamaan materiaaliefektien lisäyksen toisiin materiaaleihin.

Efektien parametrisoinnissa käytettiin apuna materiaaliparametrikokoelmaa, johon voi tallentaa globaalisti jaettavia muuttujia. Kokoelma mahdollisti sen, että siihen tallennettuihin muuttujiin pääsee käsiksi kaikista materiaaleista, joihin efektit on tarkoitus lisätä. Kokoelmaan pääsee käsiksi myös koodin avulla, mikä mahdollisti parametrien ohjelmoidun mukautuvuuden toteuttamisen.

3.2 Menettelytapa materiaaliefektien luontiin

Kehitystyö keskittyi materiaalieditorin puolelle. Aluksi efektit luotiin erillisinä materiaaleina ja niille annettiin niitä kontrolloivat parametrit. Tämän jälkeen materiaalien ulkonäkö ja parametrisointi testattiin materiaali-instanssien avulla. Testausvaiheessa materiaalit asetettiin erilaisiin 3D-objekteihin mahdollisten yhteensopivuusongelmien löytämiseksi.

Materiaalin valmistuttua, se muunnettiin materiaalifunktion muotoon ja sen parametrit asetettiin funktioon syötettäviksi muuttuja-arvoiksi. Tämän jälkeen funktion tarvitsemat parametrit lisättiin materiaaliparametrikokoelmaan, josta niiden muokkautuvuus kiinnitettiin niitä kontrolloivaan blueprint-luokkaan. Lopuksi valmis materiaalifunktio liitettiin sitä käyttävään valmiiseen materiaaliin ja sen päällystettävät ominaisuudet syötettiin funktion läpi.

3.3 Vaihtoehtoiset työkalut

Useat pelimoottorit antavat mahdollisuuden luoda dynaamisia muutoksia materiaaleihin. Esimerkiksi Unity3d-pelimoottori tarjoaa kattavan materiaali- ja varjostinkehitykseen tarkoitetun viitekehyksen. Sen avulla voidaan luoda mukautettuja varjostimia ja materiaaleja, joita voidaan myös muokata pelinajan kuluessa. (Unity Technologies 2021a.)

Unity3d-pelimoottori tarjoaa monia funktioita, joilla päästään käsiksi materiaalien ominaisuuksiin koodin kautta. Pelinaikana mahdollisia muutoksia ovat muun muassa värin vaihtaminen, liukulukuarvojen muokkaaminen ja tekstuurikarttojen asettaminen. (Unity Technologies 2021b.)

4 Järjestelmän toteuttaminen

4.1 Materiaalifunktioille asetetut vaatimukset

Järjestelmän toteuttaminen alkoi vaatimusten asettamisella. Vaatimuksia käytettiin suuntaa antavasti ja niillä tavoiteltiin kehitystyön ohjausta siten, että toteutetut materiaalifunktiot olisivat mahdollisimman yhdenoloisia. Tässä vaiheessa palattiin opinnäytetyölle asetettuihin tavoitteisiin ja vaatimukset koottiin niitä tukeviksi. Tavoitteet, joihin keskityttiin, olivat järjestelmän käytännöllisyys, efektien kontrolloitavissa olevuus, pelimaailman mukauttaminen ja interaktiivisuuden lisäys.

Ensimmäinen vaatimus oli järjestelmän käytännöllisyyden varmistaminen. Tätä vaatimusta käytettiin kehotteena pitämään valmiiden materiaalifunktion vastaanottamat parametrit mahdollisimman yksinkertaisina. Tämä tarkoittaa sitä, että parametrejä tulisi olla niin vähän kuin mahdollista ja ne tulisi olla nimetty niiden muokkaamaa olosuhdetta kuvaavasti.

Seuraava vaatimus oli järjestelmän kontrollointimahdollisuuden varmistaminen. Vaatimus kehotti materiaalifunktioiden luontia siten, että niissä on parametrejä, joita säätämällä materiaalifunktion luoma efekti voidaan muokata eri peliobjekteihin sopivaksi.

Kolmannessa vaatimuksessa painotettiin pelimaailman mukautuvuuden varmistamista. Tämä vaatimus ohjasi materiaalifunktioiden luontia siten, että ne voidaan liittää valmiisiin pelimaailmassa esiintyviin materiaaleihin.

Viimeinen vaatimus oli pelimaailman interaktiivisuuden lisäys. Vaatimuksella ohjattiin efektien suunnittelua siten, että niillä olisi interaktiivisia ominaisuuksia. Interaktiivisiksi ominaisuuksiksi lasketaan ominaisuudet, joiden avulla pelaaja itse pystyisi luomaan muutoksia efekteihin.

4.2 Lumifunktion toteuttaminen

Ensimmäinen tuotettavista sääefekteistä oli lumi (kuva 2). Lumen materiaaliefektin suunnittelu alkoi sen jakamisesta sitä eniten ilmentäviin ominaisuuksiin. Ominaisuudet olivat lumen ilmestyminen, -kertyminen, -painautuminen ja -sulaminen. Nämä neljä ominaisuutta toteutettiin erilaisten materiaalifunktioiden avulla ja yhdistettiin keskenään lumiefektin omaan materiaalifunktioon. Funktiossa neljä ominaisuutta määrittelee sen, miten lumen materiaalitaso näytetään muiden materiaalien päällä.

Kuva 2. Lumesta tuotettu materiaaliefekti (Kuva: Raimo Rautiainen).

Ensimmäisenä luotiin lumen materiaalitaso. Materiaalitaso tehtiin Material Layer -assetin avulla, jossa sille valitut tekstuurikartat, eli Base Color ja Normal, syötettiin WorldAlignedTexture-noden läpi. Sen avulla lumen tekstuurikartat ovat yhtenäisiä riippumatta siitä mihin peliobjektiin materiaalitaso on liitetty. Tämän jälkeen tekstuurikarttoihin lisättiin niiden voimakkuuksia muuntelevat funktiot ja niihin liitettiin myös käyttökohtaisesti kontrolloitavissa olevat parametrit. Materiaalitasokaavio on esitetty liitteessä 1.

Lumen ilmestyminen ja sulaminen toteutettiin tekstuurikarttamaskin ja läpinäkyvyysarvon avulla. Maskin läpinäkyvyyttä muunneltiin lumen läpinäkyvyysparametrin avulla siten, että ne molemmat syötettiin mukautettuun Lerp-funktioon. Funktio ottaa vastaan läpinäkyvyysarvon ja tekstuurikartan, ja palauttaa niistä jommankumman riippuen siitä, miten lähellä läpinäkyvyysarvo on arvoa 0.5. Tämän funktion avulla voidaan luoda siirtymä 0 arvon, tekstuurikartan ja arvon 1 välillä. Lopputuloksena on efekti, joka muistuttaa lumen ilmestymistä ja sulamista. Ilmestymisen materiaaliefekti ja siinä käytetty mukautettu Lerp-funktio on esitetty liitteessä 1.

Seuraavana haasteena oli lumen kertymisen toteuttaminen. Kertyminen haluttiin toteuttaa niin, että lumi näyttäisi ainoastoaan kertyvän peliobjektien päälle. Tätä varten tarvittiin alphamaski, joka on tietoinen pelimaailman koordinaatistosta. Alphamaski antaisi materiaaliefektin ilmestyä pinnoille, jotka osoittavat pelimaailman koordinaatistossa ylöspäin. Maskin luonti alkoi käyttämällä apuna TransformVector-nodea, joka antaa mahdollisuuden muuntaa pinnanosoitussuuntatiedon suhteellistettuna pelimaailman koordinaatistoon. Kun TransformVector-nodesta suodattaa sinisen värikanavan, jäljelle jää maski, jossa on vain ylöspäin osoittavat pinnansunnat. Tätä tekniikkaa käytettiin vektorin (0.0, 0.0, 1.0) ja peliobjektin oman Normal-tekstuurikartan kanssa. Yhdistämällä nämä kaksi maskia, saatiin tulokseksi lumen kertymisen sijoittava alphamaski, joka ottaa huomioon pinnansuunnan ja peliobjektin pinnanyksityiskohdat. Liitteessä 1 on esitetty pinnansuuntaa hyväksikäyttävän alphamaskin toteutus.

Lumen painautumisen toteuttamiseksi materiaalifunktioon lisättiin Render Target -tekstuurikartta (liite 1). Tekstuurikartta tallennettiin funktion sisään erillisenä parametrina, jotta siihen piirtävä blueprint-luokka pystyy löytämään sen ja päivittämään sitä. Blueprint-luokka ohjelmoitiin siten, että sille voi antaa referenssin peliobjektiin, jolle se luo dynaamisen materiaali-instanssin ja korvaa materiaalin käyttämän Render Target -tekstuurikartan blueprint-luokan luomalla vastikkeella. Referenssi vastikkeeseen on tallennettu luokan sisällä olevaan taulukkoon, josta sitä muokataan silloin kun järjestelmä on vuorovaikutuksessa Render Target -tekstuurikartan omaavan peliobjektin kanssa.

Render Target-tekstuurikarttaan piirtämistä varten luontiin yksinkertainen sivellinmateriaali. Sivellinmateriaalia käytettiin Scene Component -luokassa, joka liitettiin pelaajahahmon molempiin jalkoihin. Scene Component -luokka tarkisti, onko pelaajan alla Render Target -järjestelmän peliobjektitaulukkoon lisättyä peliobjektia. Tarkastukseen käytettiin erillistä collision-kanavaa ja sitä tarkastavaa SphereTraceByChannel-funktiota.

Painautumisen toteuttamisen seuraava osa on nostaa peliobjektin pinnan korkeutta. Tämä toteutettiin kertomalla VertexNormalWS-noden antama tieto peliobjektin verteksien osoitussuunnasta lumen korkeutta kuvaavalla muuttuja arvolla. Kertolaskun tulos suodatettiin aiemmin luotujen lumen ilmestymis- ja kertymismaskien avulla, jotta peliobjektin pinta liikkuisi vain ylöspäin. Painautumisen toteutuksen viimeisessä vaiheessa Render Target -tekstuurikartta liitettiin suodattavan ilmestymis- ja kertymismaskin kertoimiksi. Tällöin tekstuurikarttaan piirretyt muutokset, jotka kuvastavat nolla arvoja, madaltavat maskin kokonaismääräistä vaikutusta pinnankorkeuden nostamiseen. Tuloksena on lumen korkeuden vaihtelu ja mahdollisuus piirtää siihen uponneita polkuja. Pinnankorkeudenvaihtelun toteutus on esitelty liitteessä 1.

Lumen materiaalifunktion tuotannon viimeinen vaihe oli lumen materiaalitason piirtäminen funktioon syötettyjen pohjamateriaalin ominaisuuksien päälle. Tarvittavat ominaisuudet olivat Base Color, Roughness, Normal ja World Position Offset. Ne liitettiin lumen materiaalitason vastaaviin ominaisuuksiin Lerp-noden

avulla, jonka alphaksi syötettiin jokaisen aiemmin luodun alphamaskin yhdistelmä (liite 1).

4.3 Sadefunktion toteuttaminen

Seuraava tuotettavista sääefekteistä oli sade (kuva 3), jonka materiaaliefekti jaettiin kahteen osaan. Ensimmäinen tuotettavista efekteistä oli lammikoiden syntyminen pohjamateriaalin päälle. Lammikot haluttiin toteuttaa siten, että niiden suuruus kasvaa sateen edetessä. Sen lisäksi haluttiin, että lammikot vaikuttavat niiden ympärillä olevaan mastoon nostamalla sen kosteusarvoa. Toinen tuotettavista efekteistä oli sadepisaroiden osuma lammikoiden pintaan. Osumaefektin toteutuksessa koettiin tärkeäksi osumien koon ja nopeuden vaihtelu, mikä mahdollistaisi sateen voimakkuuden esittämisen efektissä.

Kuva 3. Esitys toteutetusta sadefunktiosta (Kuva: Raimo Rautiainen).

Lammikot toteutettiin tekstuurikartan avulla. Tekstuurikarttaa käytettiin alpha maskina, jonka voimakkuuksia muunneltiin funktioon syötettävän sateen voimakkuus parametrin avulla. Alphamaskiin lisättiin kosteuden vuotamisarvo, joka vähensi maskin kontrastia, mutta antoi mahdollisuuden toteuttaa kosteiden leviämisen lammikoiden ympärille. Tämän jälkeen lammikoihin luotiin pinnanosoitussuuntaa hyväksikäyttävä maski, aivan kuin lumiefektissä, TransformVector-noden avulla. Viimeiseksi lammikoihin lisättiin vedenpinnan toteuttamiseen tarkoitettu Normal-tekstuurikartta, jonka tekstuurikoordinaatit syötettiin panner-nodeen niiden animointia varten. Lammikko alphamaskin toteutuksen vaiheet näkyvät liitteessä 2.

Seuraavaksi toteutettiin sadepisaroiden osumat. Osumien toteutus jakautui osumaefektin laskemiseen tekstuurikoordinaattien avulla, osumien animointiin ja niiden lisäykseen lammikoiden Normal-tekstuurikarttaan. Osumaefektin laskeminen alkoi keskittämällä tekstuurikoordinaatit, jonka jälkeen laskettiin koordinaattien ja pisteen (0.0, 0.0) välinen etäisyys. Laskun tuloksena on keskipisteestä ulospäin voimistuva gradienttiefekti. Osumaefektissä tarvitaan vastakkaista efektiä, joka saadaan vähentämällä gradienttiefekti arvosta yksi. Tämän jälkeen gradienttiefektin voimakkuutta muokataan vähennys- ja jakolaskuilla, jotka muuttavat efektin ympyrältä näyttäväksi. Lopuksi varmistetaan, että efektin arvot ovat nollan ja yhden välillä, minkä jälkeen lopputulos syötetään Sine-noden läpi. Sine-node antaa tulokseksi rengasmaisen efektin, jonka paksuutta voidaan kontrolloida aiemmalla jakolaskulla, ja kokoa voidaan kontrolloida lisäämällä mikä tahansa arvo aiemmin laskettuun koordinaattien ja pisteen (0.0, 0.0) väliseen etäisyyteen. Osumaefektin vaiheet esitetty liitteessä 2.

Seuraavaksi toteutettiin osumien animointi. Animoidessa tekstuurikoordinaateista tarvittiin niiden alaspäin pyöristetty arvo, joka saatiin Floor-noden avulla. Pyöristetyt koordinaatit syötettiin Noise-funktioon, jolloin takaisin saatiin satunnainen arvo jokaiselle koordinaatille. Tämän jälkeen varmistettiin, että koordinaattiarvo on nollan ja yhden välillä ottamalla satunnaisesta koordinaattiarvoista niiden murto-osan. Minkä jälkeen satunnaiset arvot animoitiin aika-funktion avulla. Jokaiseen koordinaattiin lisättiin ajasta otettu murto-osa ja niitä verrattiin sen vastakkaiseen laskutoimitukseen. Lopputulokseksi saatiin tekstuurikoordinaattien mukainen arvoruudukko, jonka jokaisen ruudun arvo liikkuu nollan ja yhden välillä omaan satunnaiseen aikaansa. Animoitu ruudukko liitettiin kontrolloimaan osumaefektin kokoa etäisyyslaskennan yhteydessä ja häivyttämään osumaefektin sen koosta riippuen. Tarkempi esitys animaation toteuttamasta node-verkostosta näkyy liitteessä 2.

Lopulta osumaefekti liitettiin lammikon Normaali-tekstuurikarttaan (liite 2). Luotua osumaefektiä käytettiin alphamaskin tavoin siihen käytetyn tekstuurikoordinaatiston suodattamiseen. Sen jälkeen siihen liitettiin yksi vektorikomponentti lisää, jolloin se muutti tekstuurikoordinaatiston kolmen pisteen vektoriksi. Kol-

men pisteen vektorina, se voitiin liittää lammikon tekstuurikarttaan. Tuloksena oli lammikko, jonka pinnalla on sadepisaroiden osumia muistuttava efekti.

Sateen materiaalifunktion viimeinen vaihe oli luotujen lammikoiden piirtäminen funktioon syötettyjen pohjamateriaalin ominaisuuksien päälle. Materiaalifunktio otti vastaan Base Color, Roughness ja Normal ominaisuudet, ja liitti omat ominaisuutensa niiden päälle ensimmäisessä vaiheessa luodun lammikon alphamaskin avulla. Lammikon ja pohjamateriaalin ominaisuuksien liittäminen on esitetty liitteessä 2.

4.4 Parametrisoitu muuttuminen

Materiaalifunktiot vaativat niille syötettäviä arvoja, jotta ne voivat muuttua peliajan kuluessa. Arvojen syöttäminen toteutettiin materiaaliparametrikokoelman avulla, johon tallennettiin jokainen materiaalifunktioiden vaatima arvo. Kokoelmasta löytyvät arvot saatiin materiaaleihin CollectionParameter-noden avulla, josta valittiin toteutettuja materiaalifunktioita varten luotu kokoelma ja sen jälkeen toivottu parametri.

Materiaaliparametrikokoelmaan tallennettujen arvojen muuttuminen käsiteltiin blueprint-luokan avulla. Luokkaan tallennettiin materiaalifunktioiden vaatimat arvot niitä varten luodun struct-rakenteen muodossa. Tallennettuja arvoja säilytettiin taulukossa, josta ne luettiin vuorollaan. Vuorojen aikana tämänhetkisiä arvoja verrattiin taulukkoon tallennettuihin arvoihin ja ne päivitettiin tarvittaessa.

5 Pohdinta

5.1 Saavutetut tavoitteet

Opinnäytetyön aikana lähdettiin tutustumaan ja tutkimaan erilaisia tapoja elävöittää pelimaailmoja. Päätavoitteena oli toteuttaa järjestelmä, joka luo mukau-

tuvuutta pelimaailmaan dynaamisilla materiaaleilla. Opinnäytetyössä tavoiteltiin myös käytännöllistä ja kontrolloitavissa olevaa tapaa esittää sääolosuhteita. Näiden tavoitteiden lisäksi tahdottiin kokeilla pelimaailmojen interaktiivisuuden lisäämistä dynaamisten materiaalien avulla.

Opinnäytetyössä toteutettu järjestelmä onnistui elävöittämään pelimaailmaa materiaalifunktioiden avulla. Toteutetut materiaalifunktiot loivat kontrolloitavissa olevia sääolosuhteita, jotka mukautuivat niille syötettyjen parametrien mukaisesti. Sääolosuhteet ja niiden välinen vaihtelu koettiin uskottavaksi, koska materiaaliefektit pystyivät piirtämään itsensä toistensa päälle virtaviivaisesti. Käytettävyyttä kohtaan asetetut tavoitteet koettiin myös saavutetuiksi, koska materiaalifunktiot olivat yksinkertaista lisätä valmiisiin materiaaleihin ja niiden kontrollointi onnistui pienellä määrällä parametrejä. Materiaaliefektien kontrolloitavuus saavutettiin onnistuneesti materiaalien ulkopuolella olevasta blueprint-luokasta. Blueprint-luokka oli yhdistyneenä materiaalin käyttämiin parametreihin ja materiaaliparametrikokoelmaan.

Materiaalifunktioiden toteutuksessa kokeiltiin monia eri tekniikoita. Kummassakin materiaalifunktiossa haettiin optimaalisia ratkaisuja, aloittamalla yksinkertaisesti ja lisäämällä halutut toiminallisuudet minimaalisessa muodossa. Myöhemmin niihin asetettiin mukautuvuutta ja monimutkaisuutta. Kehityksen loppuvaiheessa, niistä yritettiin karsia kaikki ylimääräiset ilmaisut.

Onnistuin saavuttamaan myös itselleni asettamani tavoitteet. Opinnäytetyön aikana pystyin syventymään Unreal Engine 4:n tarjoamiin toiminnallisuuksiin ja opin paljon uutta materiaalien luonnista ja niiden pelinaikaisesta muokkauksesta. Näiden lisäksi, ymmärrykseni materiaalien luontia kohti kasvoi huomattavasti ja onnistuin paikantamaan vääriä olettamuksia, joita tunnistin omaavani opinnäytetyötä tehdessä.

5.2 Jatkokehitysmahdollisuudet

Opinnäytetyössä saavutettiin paljon, mutta järjestelmän kehitystyötä voisi viedä pidemmällekin. Järjestelmään voisi esimerkiksi lisätä muita sääolosuhteita. Sääolosuhteet voitaisiin suunnitella niin, että ne ovat kontrolloitavissa yhteisillä parametreillä, kuten esimerkiksi kosteusarvolla ja lämpötilalla. Tämä loisi pohjustuksen kattavan ja realistisen vuodenaika järjestelmän tuottamiseen.

Järjestelmässä käytetyn ohjaajaluokan voisi toteuttaa niin, että se laskee jokaisella kierroksella uudet satunnaiset parametrit. Satunnaisuuden avulla jokaiselle kierrokselle voitaisiin luoda uudenoloisuutta ja kierrokset pysyisivät arvaamattomina.

Tämänhetkistä Render Target -järjestelmää pystyisi myös jatkokehittämään. Järjestelmän tarkoituksena oli mahdollistaa vain valittujen objektien tekstuurikarttojen päivittäminen. Vaihtoehtoisesti peliobjektien sisällyttämisprosessia voitaisiin virtaviivaistaa kehittämällä yhtenäisen tavan, jolla kaikki materiaalifunktion omaavat peliobjektit saavat päivityksiä automaattisesti.

Materiaalifunktioiden tuottamia efektejä voitaisiin viilata realistisimmiksi. Lammikoiden tuottamiseen voitaisiin ottaa mukaan ympäristöstä löytyvät pinnankorkeudenvaihtelut ja käyttää niitä lammikkojen sijoittamisen apuna. Lumiefektiin voitaisiin lisätä lumikerroksen omia pinnanmuodonvaihteluja ja muokata lumen kertymisen ominaisuuksia. Esimerkiksi tuulensuunnalla ja -voimakkuudella voitaisiin mahdollistaa lumen kertyminen tuulensuunnan vastaisiin pystysuoriin pintoihin.

5.3 Loppuhavainnot

Opinnäytetyötä tehdessä havaittiin, että Unreal Engine 4 -pelimoottorin tarjoamilla toiminnallisuuksilla pystytään luomaan käytännöllistä pelinaikaista mukautuvuutta pelimaailmaan. Materiaalit ja niiden ominaisuudet antoivat paljon pelivaraa erilaisten dynaamisten efektien kehittämiseen. Materiaalifunktiot ja -tasot olivat käytännöllisiä ja virtaviivaistivat kehitystyötä. Render Target -tekstuurikarttojen käyttö materiaaleissa antoi lisämahdollisuuksia mukautuvuu-

den toteutukseen. Dynaamisesti mukautuvien materiaalien kehittäminen toi pöydälle uusia haasteita ja pakotti katsomaan materiaalien luontia erilaisista näkökulmista. Tämän takia kehitystyökalun tarjoamat menetelmät virtaviivaistettuun kehitystyöhön ja nopeaan testaukseen, koettiin tarpeellisiksi.

Opinnäytetyössä saavutettuihin tuloksiin päästiin käyttäen samankaltaisia Render Target -tekniikoita, kuin mitä Murphy (2018) käytti esittelemässään dynaamisessa landscape-materiaalissaan. Opinnäytetyössä toteutettu järjestelmä käytti Render Target -tekstuurikarttaa ja muita alphamaskeja eri materiaalitasojen ja efektien väliseen kommunikointiin. Opinnäytetyössä haettu tulokulma kumminkin painotti monien materiaaliefektien käytännöllistä ja kontrolloitavissa olevaa esittämistapaa.

6 Lähteet

Epic Games Inc. 2021a. Creating a Height Field Painter with Blueprints and Render Targets. Epic Games Inc. https://docs.unrealengine.com/4.26/en-US/RenderingAndGraphics/RenderTargets/BlueprintRenderTargets/HowTo/HeightFieldPainter/. 7.12.2021.

Epic Games Inc. 2021b. The world’s most open and advanced real-time 3D creation tool. Epic Games Inc. https://www.unrealengine.com/en-US/. 7.12.2021.

Epic Games Inc. 2021c. Materials. Epic Games Inc. https://docs.unrealengine.com/4.26/en-US/RenderingAndGraphics/Materials/. 7.12.2021.

Epic Games Inc. 2021d. Material Editor Reference. Epic Games Inc. https://docs.unrealengine.com/4.26/en-US/Source/RenderingAndGraphics/Materials/Editor/. 7.12.2021.

Epic Games Inc. 2021e. Material Properties. Epic Games Inc. https://docs.unrealengine.com/4.26/en-US/RenderingAndGraphics/Materials/MaterialProperties/. 7.12.2021.

Epic Games Inc. 2021f. Material Blend Models. Epic Games Inc. https://docs.unrealengine.com/4.26/en-US/RenderingAndGraphics/Materials/MaterialProperties/BlendModes/. 7.12.2021.

Epic Games Inc. 2021g. Shading Models. Epic Games Inc. https://docs.unrealengine.com/4.26/en-US/RenderingAndGraphics/Materials/MaterialProperties/LightingModels/. 7.12.2021.

Epic Games Inc. 2021h. Making Parameters. Epic Games Inc. https://docs.unrealengine.com/en-US/RenderingAndGraphics/Materials/HowTo/Making_Parameters/index.html. 27.4.2021.

Epic Games Inc. 2021i. Material Instances. Epic Games Inc. https://docs.unrealengine.com/en-US/RenderingAndGraphics/Materials/MaterialInstances/index.html. 27.4.2021.

Epic Games Inc. 2021j. Material Parameter Collections. Epic Games Inc. https://docs.unrealengine.com/en-US/RenderingAndGraphics/Materials/ParameterCollections/index.html. 17.5.2021.

Epic Games Inc. 2021k. Material Functions Overview. Epic Games Inc. https://docs.unrealengine.com/4.26/en-US/RenderingAndGraphics/Materials/Functions/Overview/. 7.12.2021.

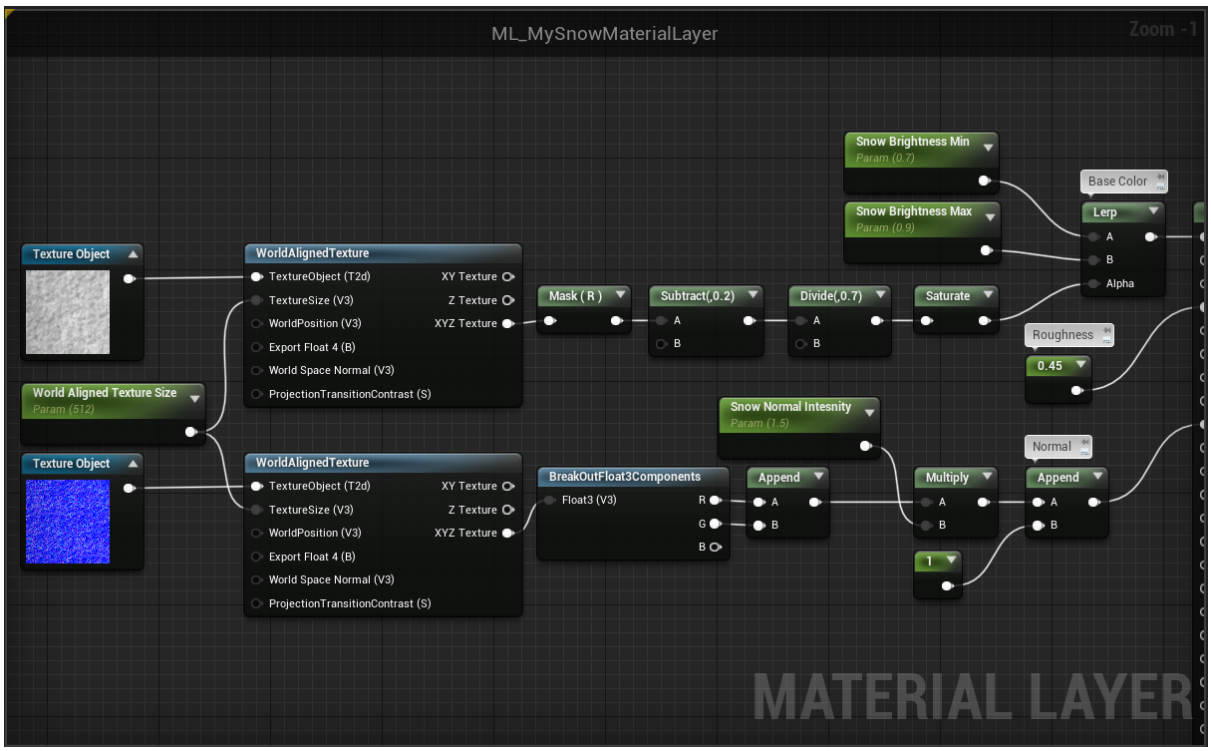
Epic Games Inc. 2021l. Material Layers. Epic Games Inc. https://docs.unrealengine.com/4.26/en-US/RenderingAndGraphics/Materials/MaterialLayers/.

Murphy. C. 2018. Building High-End Gameplay Effects with Blueprint. https://www.youtube.com/watch?v=67z5u8ZcEcw. 10.12.2021.

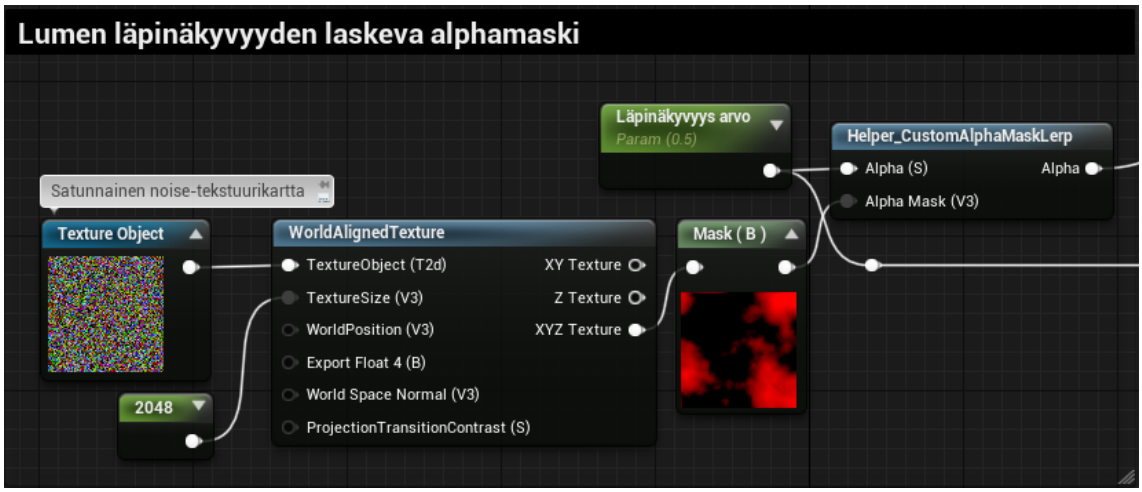
Unity Technologies. 2021a. Material. https://docs.unity3d.com/ScriptReference/Material.html. 10.12.2021.

Unity Technologies. 2021b. Accessing and Modifying Material parameters via script. https://docs.unity3d.com/Manual/MaterialsAccessingViaScript.html. 10.12.2021.

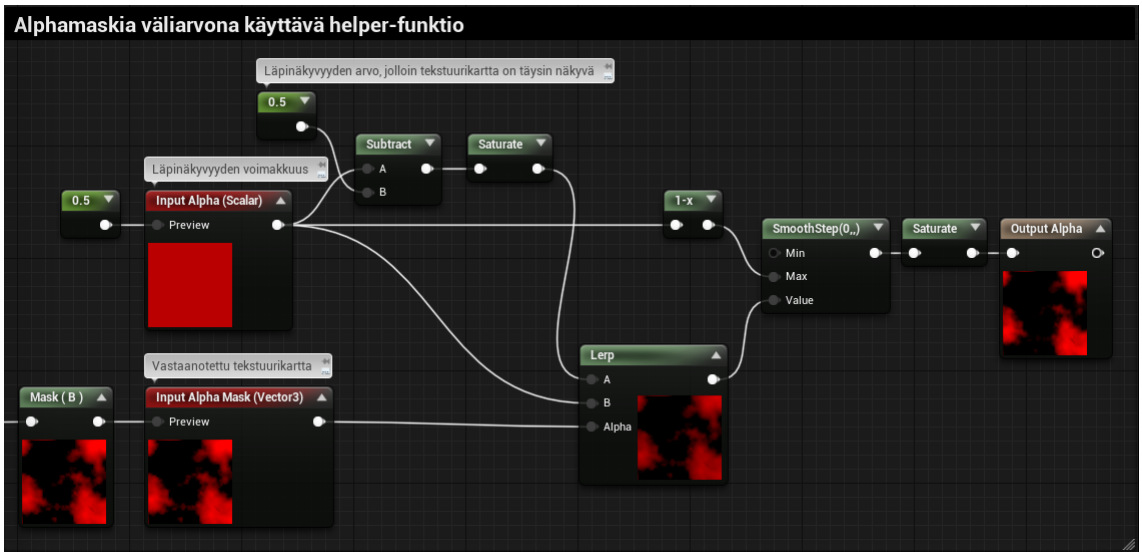
Liite 1 Lumen materiaaliefektin tuottaminen



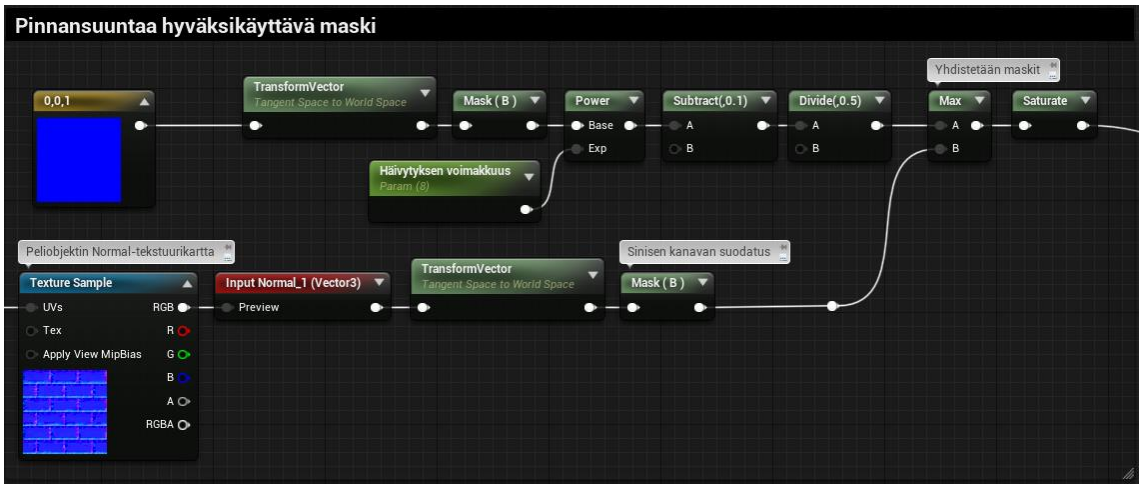
Kuva 4. Lumen tuottava materiaalitasokaavio, jossa lumen tekstuuriobjektit syötetään WorldAlignedTexture-nodeen ja molemmille luodaan säätelyarvot (Kuva: Raimo Rautiainen).



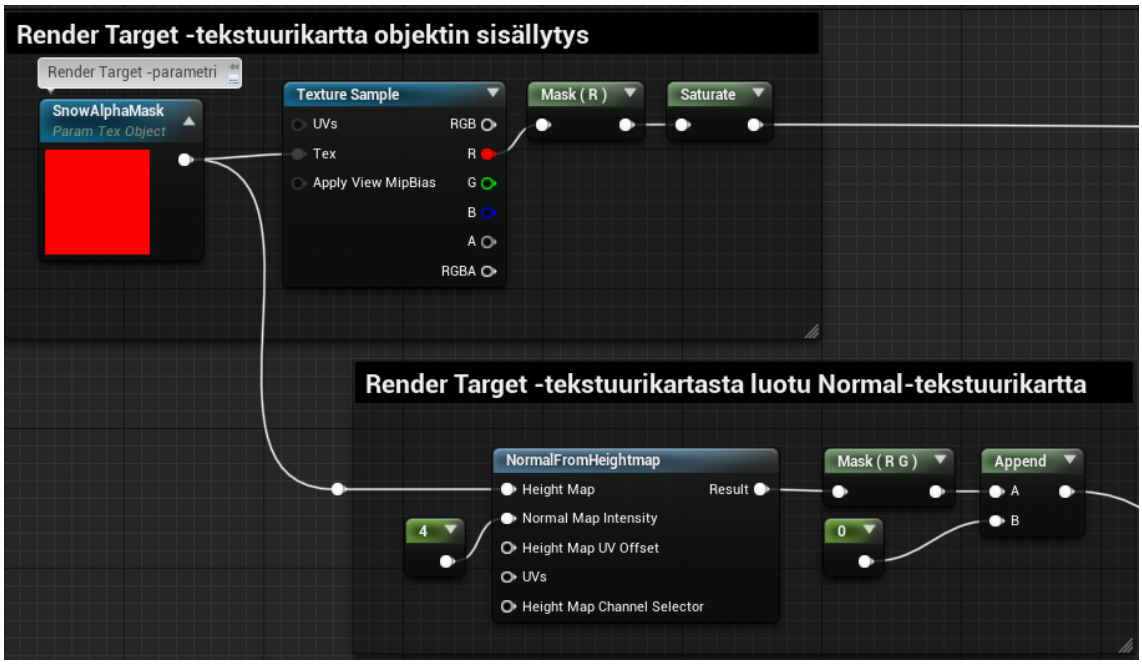
Kuva 5. Lumen läpinäkyvyysarvon laskeva alphamaski. Satunnaisen noise-funktion tuottama tekstuurikarttaobjekti syötetään WorldAlignedTexture-node läpi, jotta käytetty alphamaski toistuu yhtenäisesti kaikkien sitä käyttävien peliobjektien päällä (Kuva: Raimo Rautiainen).



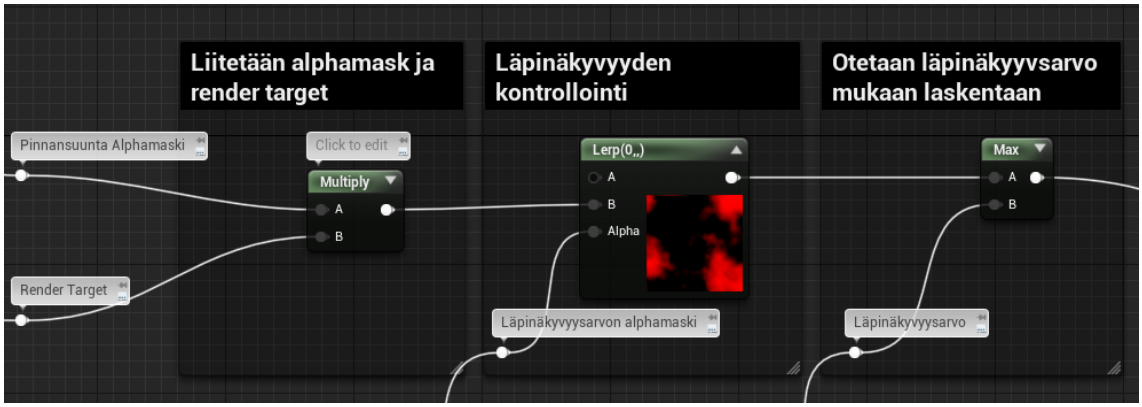
Kuva 6. Läpinäkyvyysarvon vaihtelun auttamiseen suunniteltu apufunktio, joka ottaa vastaan läpinäkyvyysarvon ja tekstuurikartan, ja palauttaa niistä (Kuva: Raimo Rautiainen).



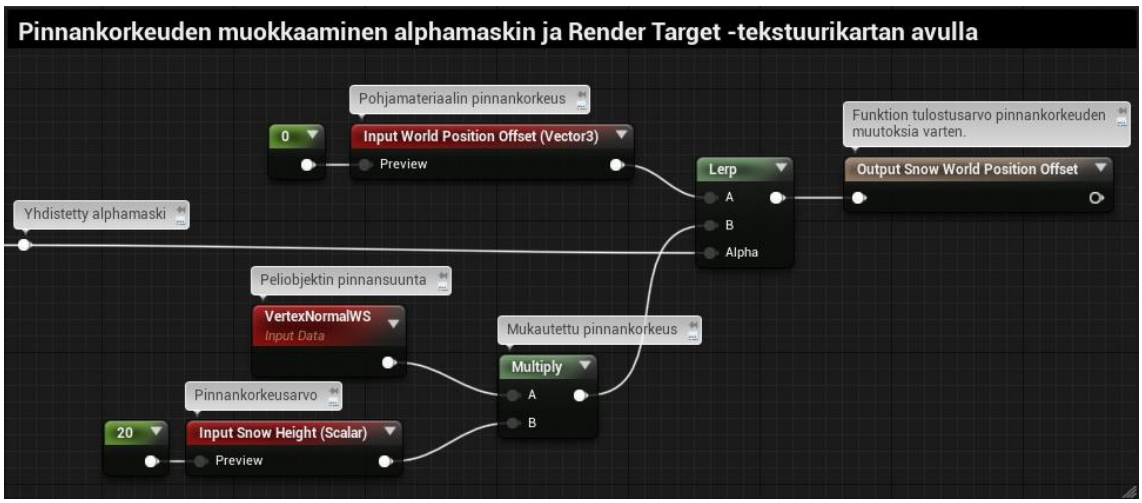
Kuva 7. Pinnansuuntaa ja peliobjektin Normal-tekstuurikarttaa käyttävä alpha-maskin luonti (Kuva: Raimo Rautiainen).



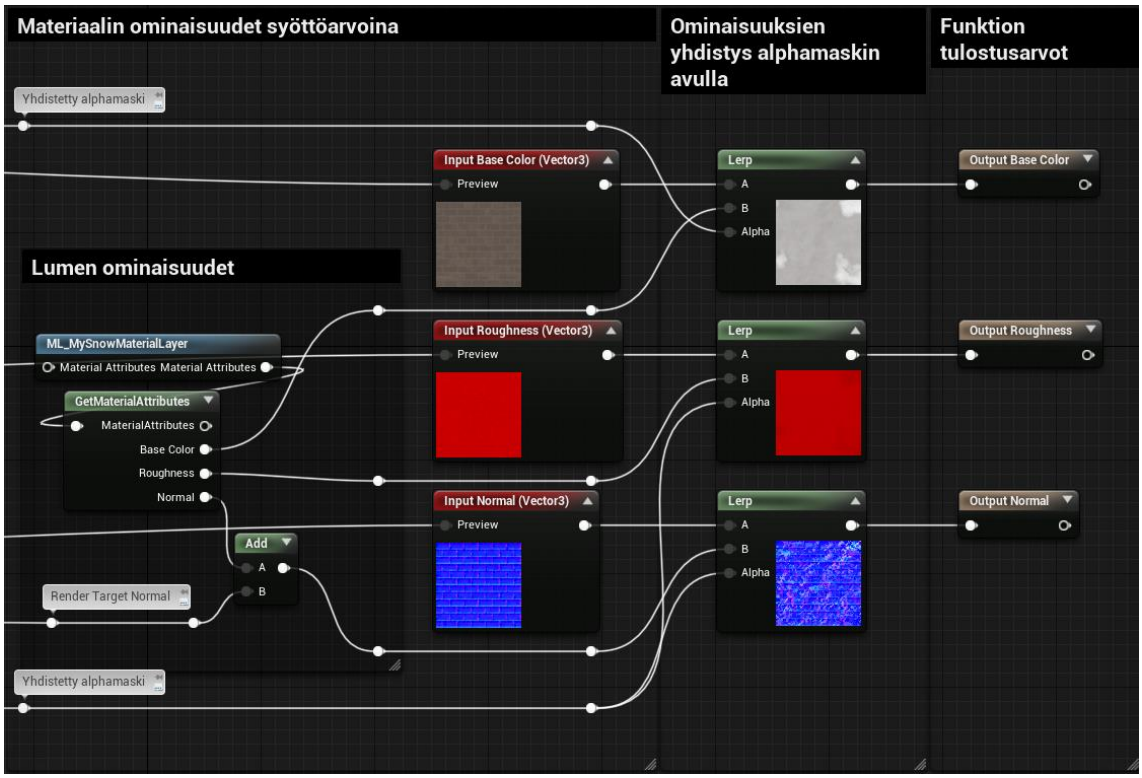
Kuva 8. Render Target -tekstuurikartan lisäys ja Normal-tekstuurikartan luonti sen perusteella (Kuva: Raimo Rautiainen).



Kuva 9. Alphamaskien, Render Target -tekstuurikartan ja läpinäkyvyysarvon yhdistäminen (Kuva: Raimo Rautiainen).

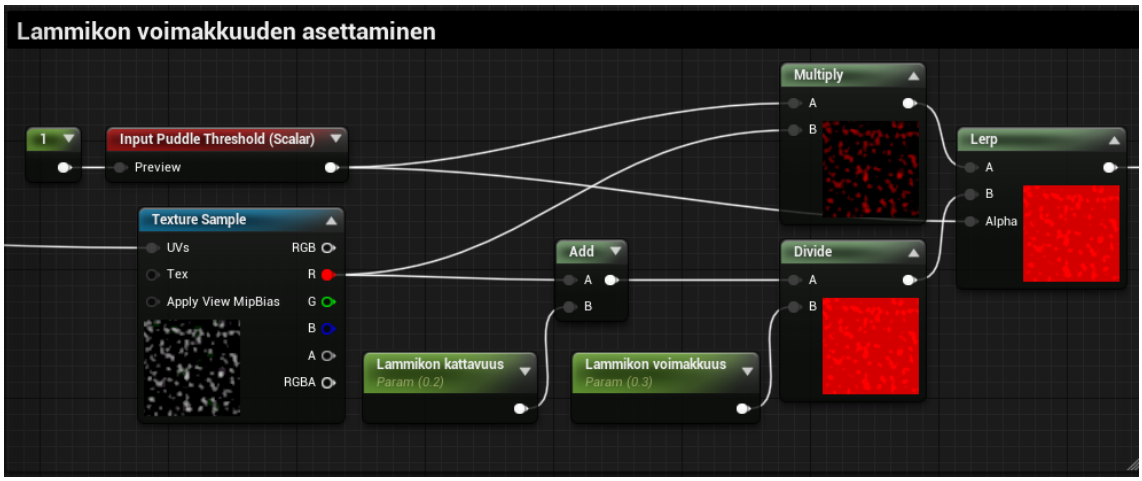


Kuva 10. Pinnankorkeuden muutos (Kuva: Raimo Rautiainen).

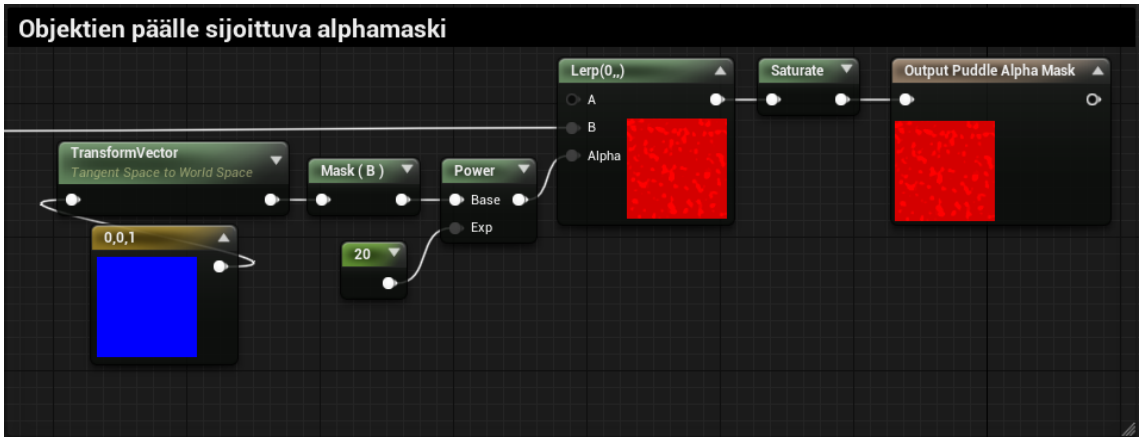


Kuva 11. Materiaalitason ja pohjamateriaalin yhdistäminen Lerp-nodella ja yhdistetyllä alphamaskilla (Kuva: Raimo Rautiainen).

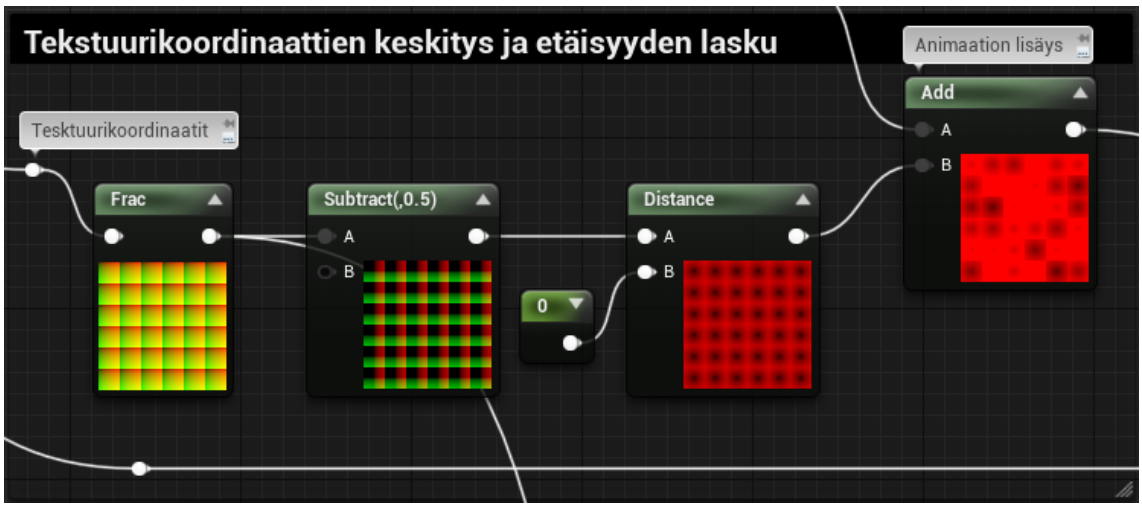
Liite 2 Sateen materiaaliefektin tuottaminen



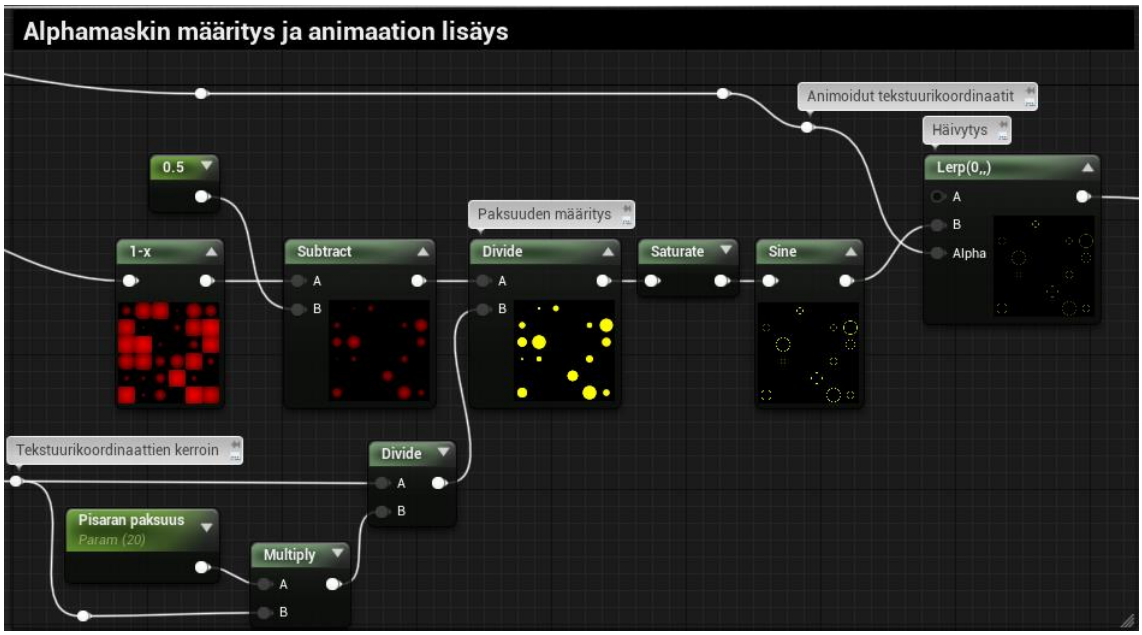
Kuva 12. Lammikon tekstuurikartan voimakkuuksien määritys (Kuva: Raimo Rautiainen).



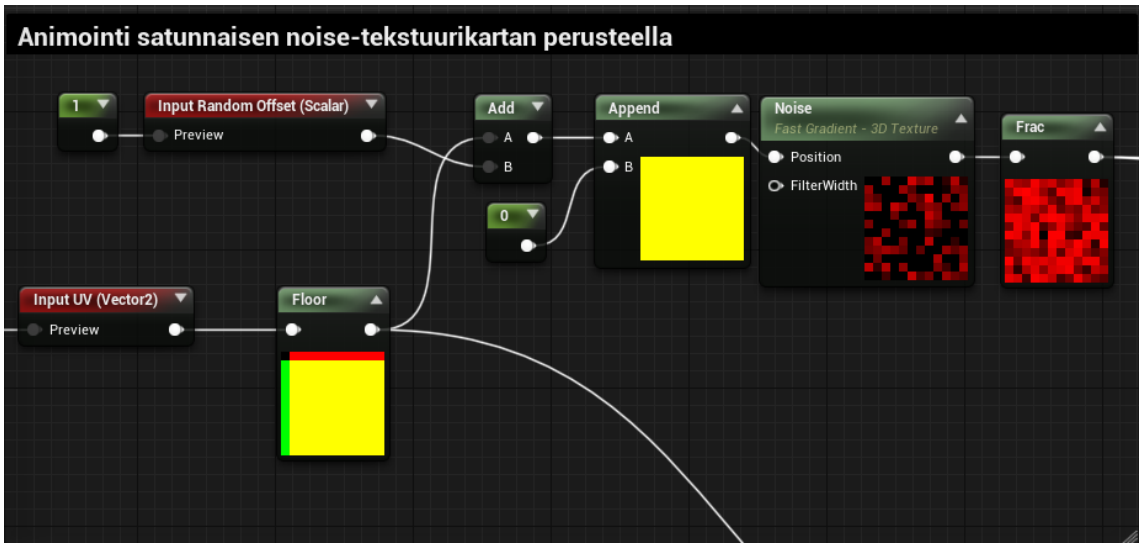
Kuva 13. Lammikon ilmestymisen rajoitus peliobjektien pinnalle (Kuva: Raimo Rautiainen).



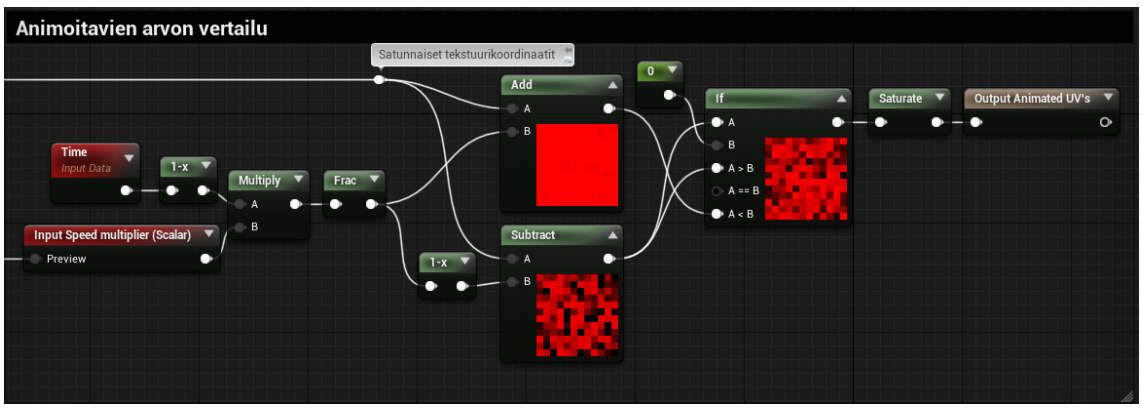
Kuva 14. Osumaefektin tekstuurikoordinaattien keskitys ja etäisyyslaskenta (Kuva: Raimo Rautiainen).



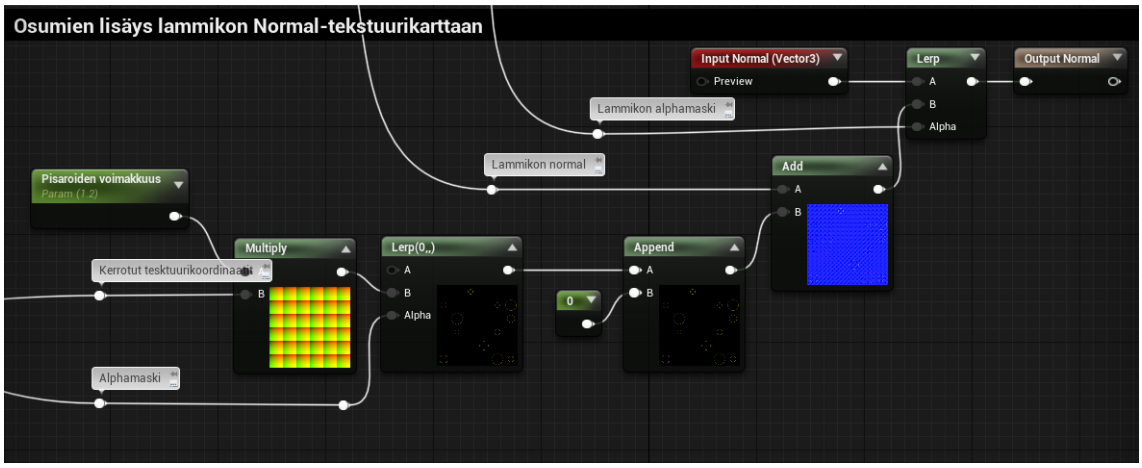
Kuva 15. Osumien koon ja paksuuden määritys (Kuva: Raimo Rautiainen).



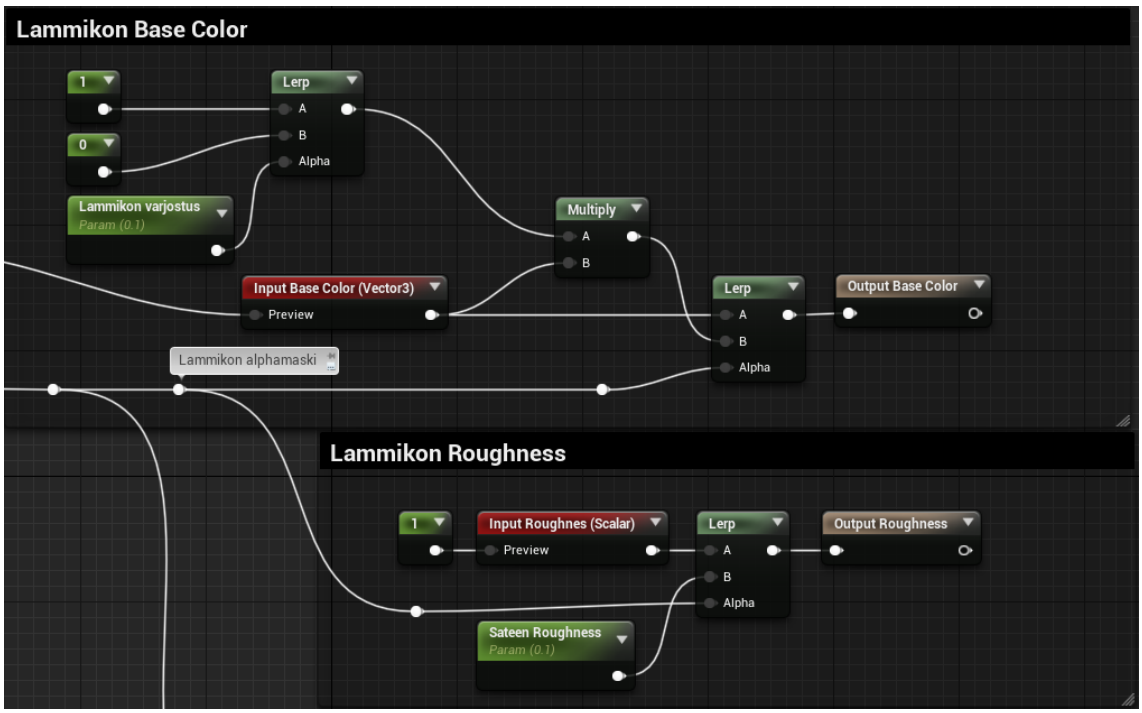
Kuva 16. Satunnaisen noise-funktion käyttö animoinnissa (Kuva: Raimo Rautiainen).



Kuva 17. Satunnaisella noise-funktiolla luotujen tekstuurikoordinaattiarvojen välinen vertailu (Kuva: Raimo Rautiainen).



Kuva 18. Osumien voimakkuuden asetus ja lisäys lammikon Normal-tekstuurikarttaan (Kuva: Raimo Rautiainen).



Kuva 19. Lammikon Base Color ja Roughness -arvojen määritys (Kuva: Raimo Rautiainen).