Essi Kykkänen

VISUAALISTEN EFEKTIEN KEHITTÄMINEN PELIHAHMOLLE

Opinnäytetyö

Tekniikan ammattikorkeakoulututkinto

Peliohjelmoinnin koulutus

2024

Kaakkois-Suomen ammattikorkeakoulu

Kaakkois-Suomen ammattikorkeakoulu

Tutkintonimike	Insinööri (AMK)
Tekijä/Tekijät	Essi Kykkänen
Työn nimi	Visuaalisten efektien kehittäminen pelihahmolle
Toimeksiantaja	Normogames Oy
Vuosi	2024
Sivut	68 sivua
Työn ohjaaja(t)	Marko Oras

TIIVISTELMÄ

Tämä opinnäytetyö käsittelee visuaalisia efektejä ja niiden kehittämistä aina suunnittelusta toteutukseen. Kehitysympäristönä toimi Unreal Engine -pelimoottori, jonka sisäistä Niagara VFX -työkalua hyödynnettiin efektien toteuttamiseen. Opinnäytetyön päätavoitteena oli suunnittelun ja toteutuksen avulla selvittää, kuinka visuaalisia efektejä voidaan kehittää pelihahmolle. Tarkemmin työn aikana kehitetyt visuaaliset efektit olivat partikkeliefektejä. Päätavoitteen lisäksi oli tarkoitus selvittää myös keinoja efektien optimoimiseksi.

Työ toteutettiin kehittämistutkimuksena. Tutkimuksen aikana havainnoitiin työn kannalta samantyyppisiä olemassa olevia efektejä, joiden perusteella selvitettiin niille ominaisia piirteitä. Havaintoja hyödynnettiin suunnitteluvaiheessa, joka toimi pohjana efektien toteutukselle. Suunnittelussa huomioitiin sekä visuaalisen että teknisen puolen suunnittelu.

Niagara-työkalun ja kehitysympäristön ominaisuuksia hyödyntäen työn tuloksena valmistui kolme partikkeliefektiä, jotka toimivat yhdessä hahmon animaatioiden kanssa. Niagara-työkalua käsitellään työn teoreettisessa osiossa ja sen ominaisuuksiin syvennytään enemmän efektien toteutusvaiheessa. Toteutuksen kuvaus pohjautuu kehittämistyön aikana tehtyihin kenttämuistiinpanoihin.

Koska kaikki suunnitellut efektit saatiin toteutettua, Niagara-työkalu siis osoittautui soveltuvaksi efektien kehittämiseen. Efekteillä on usein erilaisia toiminnallisuuksia, jotka on otettava huomioon pelihahmon osalta. Suunnitteluvaiheessa efektikokonaisuuksia jaettiin pienempiin osiin. Osittelu selkeytti alustavien ratkaisujen tekemistä teknisen suunnittelun aikana, joka puolestaan helpotti efektien toteuttamista. Optimoinnin keinojen soveltaminen jäi hieman yksipuoliseksi, mutta sen jatkaminen on eräs jatkokehityksen mahdollisuus. Optimoinnin osalta saatiin kuitenkin kerättyä hyödyllistä teoriaa efektien tehokkuuteen vaikuttavista tekijöistä.

Asiasanat: visuaaliset efektit, partikkeliefektit, optimointi, efektikehitys

South-Eastern Finland University of Applied Sciences

Degree title	Bachelor of Engineering
Author (authors)	Essi Kykkänen
Thesis title	Visual effect development for a video game character
Commissioned by	Normogames Oy
Time	2024
Pages	68 pages
Supervisor	Marko Oras

ABSTRACT

The main objective of the thesis was to research how visual effects can be developed specifically for a video game character. Additionally, another aim was to discover methods for optimizing the visual effects. Unreal Engine was chosen as the development environment for this work, utilizing its built-in tool called Niagara VFX System for the visual effect creation.

The effects were designed both visually and technically during the design phase. Visual designs were represented in the form of drawn concept material, which illustrated different stages of the effects and the separation of their components. During the technical design phase, various implementation techniques were considered and eventually, the effects were implemented in Unreal Engine based on the designs.

As a result of the research, three different visual effects were designed and implemented. The implementations followed the designs closely. Therefore, the Niagara VFX System works well as a VFX development tool. Attaching the effects to a game character was eventually achieved simply. However, effects often have individual requirements when attaching them to a character. For that reason, searching for unique solutions is usually inevitable. The thesis doesn't work directly as a guide, but it provides useful insights into what the VFX development process should include.

Keywords: visual effects, particle effects, optimization, effect development

SISÄLLYS

1 JOHDANTO

Visuaaliset efektit ovat merkittävä osa videopelejä nykypäivänä. Ne tuovat peleihin lisää eloa, jännitystä sekä immersiivisyyttä eli virtuaalitodellisuuteen ”uppoutumista”. Erilaisia videopeleissä nähtäviä efektejä ovat esimerkiksi räjähdykset, savu, tuli, vesi ja partikkelit. Visuaaliset efektit eivät ole ainoastaan koristeita pelin visuaalisen ilmeen kannalta, vaan ne myös välittävät pelaajalle tärkeää informaatiota. (The Ultimate Guide to VFX… s.a.) Opinnäytetyössä keskitytään visuaalisiin efekteihin pelihahmojen kannalta, ja efektejä hyödyntämällä voidaan lisätä muun muassa hahmojen syvällisyyttä ja kiinnostavuutta.

Tässä opinnäytetyössä käsitellään visuaalisia efektejä ja tutkitaan, kuinka niitä voidaan kehittää pelihahmoille. Tarkoituksena on suunnitella pelihahmolle erityyppisiä efektejä hyödyntäen tutkimusta varten kerättyä teoreettista pohjaa ja suunnitelman perusteella toteuttaa efektit Unreal Engine -pelimoottorissa.

Aihe työlle valikoitui tekijän oman kiinnostuksen perusteella. Efektien kehittämisessä yhdistyy sekä visuaalinen että tekninen osaaminen. Kiinnostus visuaalisia efektejä kohtaan kasvoi erityisesti opintojen loppupuolella ja työn avulla niiden kehittämistä pääsee tutkimaan perusteellisesti. Opinnäytetyön jälkeen olisi mielenkiintoista jatkaa efektien parissa myös myöhemmin työuralla.

Opinnäytetyön aiheesta kiinnostuneena toimeksiantajaksi ryhtyi Normogames Oy, joka on vuonna 2022 perustettu kotkalainen peliyritys. Yritys kehittää tietokoneella pelattavia videopelejä, joten opinnäytetyössä keskitytään efektien kehitykseen tietokonepelien näkökulmasta. Toimeksiantajayritys pystyy myöhemmin hyödyntämään tutkimusta omien tarpeidensa mukaan, mutta tutkimuksessa kehitetyt efektit eivät tule konkreettiseen käyttöön toimeksiantajan projekteihin.

2 TUTKIMUSASETELMA

Työn sisältö koostuu muutaman erilaisen visuaalisen efektin suunnittelusta ja toteutuksesta Unreal Engine -pelimoottorissa. Unreal Engine valittiin kehitysympäristöksi toimeksiantajan toiveesta. Työn aikana kehitettävien efektien määrä on rajattu kolmeen.

2.1 Työn tavoitteet

Opinnäytetyön yleisenä päämääränä on selventää visuaalisten efektien kehityksen vaiheita ja menetelmiä sekä työn tekijälle että toimeksiantajayritykselle. Tämä päätavoite jakautuu kahteen osaan: efektien suunnitteluun ja niiden toteuttamiseen. Työn aikana visuaaliset efektit suunnitellaan teoreettisen tutkimuksen avulla konseptiksi ja lopulta toteutetaan tehtyjen suunnitelmien pohjalta. Suunnitteluvaiheessa pyritään hyödyntämään jo tunnettuja suunnittelun periaatteita, mutta myös tutkimaan, mitä seikkoja niiden lisäksi olisi hyvä sisällyttää suunnitteluprosessiin.

Suunnittelu kattaa sekä visuaalisen puolen että teknisen suunnittelun. Teknisyydellä tarkoitetaan efektin teknisen toteutuksen suunnittelemista tutkimalla pelimoottorin työkaluja. Näin saadaan alustavasti kartoitettua keinoja, millä efektiä voitaisiin lähteä toteuttamaan Unreal Engine -pelimoottorissa. Saman efektin aikaansaamiseksi voi olla useampiakin eri toteutustekniikoita, jolloin teknisen suunnittelun avulla saadaan tehtyä ratkaisuja, jotka sopivat juuri tietyntyyppisen efektin toteutukselle.

Opinnäytetyön aikana on tarkoitus kehittää kolme erilaista visuaalista efektiä pelihahmolle. Pelihahmo voi olla lähes minkä muotoinen objekti tahansa, mutta tässä työssä pelihahmon määritelmää tarkennetaan kuitenkin koskemaan humanoideja eli rakenteeltaan karkeasti ihmisen mallisia 3D-pelihahmoja. Tutkimusta on silti mahdollista hyödyntää eri muotoisten hahmojen kohdalla, sillä toteutustavat eivät riipu hahmon muodosta.

Tässä opinnäytetyössä kehitettävät efektit liittyvät myös tarkennetusti pelihahmon erityisiin kykyihin. Peleissä hahmojen kykyjä ovat tyypillisesti erilaiset taisteluliikkeet ja -voimat. Pelihahmojen kykyjä kuvastetaan paljon animaati-

oilla, mutta visuaaliset efektit voivat tuoda esimerkiksi taisteluliikkeiden liikeratoja ja voimakkuutta selkeämmin esille sekä luoda syvyyttä ja tunnelmaa pelattavuuteen. Efektien avulla voidaan antaa ilme myös mielikuvituksellisille elementeille. Esimerkiksi taianomaiset voimat olisivat kovin mitäänsanomattomia pelkillä animaatioilla kuvastettuna, mutta efektien avulla maagisuus saadaan heräämään henkiin.

Työn aikana kehitettävät efektit liittyvät siis hahmon kykyihin. Valitut kyvyt, joiden perusteella efektit toteutetaan, ovat maahan kohdistuva isku, parannusloitsu ja maaginen AOE-hyökkäys. Maahan kohdistuva isku aiheuttaa reaktion maa-alueeseen, parannusloitsu pyörii hahmon ympärillä palauttaen elinvoimaa, ja maagisen AOE-hyökkäyksen on tarkoitus kuvastaa pelihahmon taianomaista voimaa, jolla on laaja vaikutusalue. Kyseiset kyvyt on valittu niillä perustein, että ne ovat yleisiä hyökkäys- ja voimatyyppejä videopeleissä, ja niille kehitettävät efektit vaativat erilaisia ratkaisuja.

Työssä ei ole tarkoituksena toteuttaa 3D-hahmon mallinnusta ja animointia itse. Niitä varten hyödynnetään Mixamo-palvelua, josta hahmo ja sille tarvittavat animaatiot saadaan ladattua valmiina. Yksinkertaisemmat 3D-mallit, joita hyödynnetään varsinaisesti efektien toteutuksessa, tuotetaan itse.

2.2 Tutkimusongelma ja -kysymykset

Työn päätavoitteena on selvittää, kuinka pelihahmoille voidaan kehittää efektejä, mutta lisäksi myös tutkia millä tavoin efektejä voidaan optimoida. Näin ollen opinnäytetyön tutkimusongelmaksi voidaan määritellä visuaalisten efektien kehittäminen pelihahmolle Unreal Enginessä.

Opinnäytetyön aikana pyritään vastaamaan neljään tutkimuskysymykseen. Tutkimusongelmasta johdetut tutkimuskysymykset ovat seuraavat:

1. Mitä efektin suunnitteluprosessissa tulisi huomioida?

Kuten muutkin asiat peleissä, myös efektit tulee suunnitella ennen niiden toteuttamista. On tiettyjä periaatteita, joita suunnittelussa tulisi seurata, mutta periaatteiden ympäriltä voi löytyä myös muita huomionarvoisia seikkoja.

2. Miten kehitysympäristön työkalulla voidaan toteuttaa yksinkertaisia pelihahmon efektejä?

Työkalu sisältää monia ominaisuuksia erilaisten efektien toteuttamiseksi. Kehityksen aikana työkalusta voi nousta esiin yleisiä ominaisuuksia, jotka toistuvat efektien toteutuksessa.

3. Kuinka efektit saadaan integroitua pelihahmon animaatioihin?

Visuaalisten efektien täytyy näkyä oikein pelihahmon animaation eri vaiheissa. Efektin vääränlainen ajoitus, sijainti tai rotaatio animaatioon nähden ei ole eduksi immersiivisyydelle, jota efekteillä tahdotaan juuri lisätä.

4. Millä tavoin visuaalisia efektejä voidaan optimoida?

Ilman optimointia efektit voivat kuormittaa pelin suorituskykyä. Pelien pelattavuus ei kuitenkaan saisi kärsiä efektien takia, minkä vuoksi optimointi on tärkeä osa-alue efektien kehittämisessä.

Kehitysprosessin aikana pyritään selvittämään vastauksia ja ratkaisuja yllä esitettyihin tutkimuskysymyksiin, jotka lopulta ratkaisevat tutkimusongelman. Työn konkreettisena tuotoksena syntyy kolme erityyppistä optimoitua pelihahmon efektiä.

2.3 Tutkimusote

Opinnäytetyö toteutetaan kehittämistutkimuksena. Kehittämistutkimuksessa pyritään muutokseen; kehittämisellä halutaan parantaa taustalla olevia, ei-sosiaalisia ilmiöitä, kuten tuotteita, palveluita, prosesseja tai asiantiloja (Kananen 2012, 13, 41). Muutokseen pyrkivien tutkimusten yläkäsitteenä voidaan pitää interventiotutkimusta, jonka eräs tutkimusmuoto on kehittämistutkimus (Jönssön & Luukka 2005, Kanasen 2017, 10 mukaan). Tässä opinnäytetyössä taustalla oleva prosessi on visuaalisten efektien kehittäminen. Kehittämisen avulla pyritään kasvattamaan pääsääntöisesti toimeksiantajayrityksen tietoisuutta efektien kehitysprosessista ja mahdollisesti tuomaan se myös osaksi yrityksen toimintaa.

Kehittämistutkimus ja kehittämistyö ovat lähellä toisiaan. Ollakseen tutkimuksellista kehittämistyön on perustuttava tieteellisyyteen. Tämä tapahtuu kehittä-

mistyön dokumentoinnilla ja tieteellisten menetelmien käyttämisellä. Jotta tutkimus olisi lisäksi tieteellistä, on sen perustuttava uutuuteen, sillä uuden tiedon tuottaminen on yksi tieteen kriteereistä. (Kananen 2012, 20–21.) Toimeksiantajayrityksellä ei ole ollut aiempia menetelmiä visuaalisten efektien toteuttamiseksi, joten tutkimuksen pohjalta syntyvä toimintamalli on yrityksen kannalta uutuudenarvoinen. Lisäksi toimeksiantajayritys on työskennellyt aiemmin pääosin Unity-pelimoottorin parissa, jolloin tutkimuksen työstäminen juuri Unreal Enginellä tuo uutta näkökulmaa myös kehitysympäristön kannalta.

Yritys pystyy myöhemmin hyödyntämään tutkimusta omissa projekteissaan, mutta se ei tule välttämättä viralliseksi osaksi yrityksen toimintaa, vaan jää enemmänkin toimintamalliehdotukseksi. Täten tutkimus pyrkii siis muutokseen. Opinnäytetyö perustuu laajasti aiemmin tutkittuun tietoon ja teoriaan, sillä visuaaliset efektit itsessään eivät ole uusi asia videopelien saralla. Interventionistisille tutkimuksille onkin tyypillistä, että ne perustuvat aikaisempaan teoriaan ja tutkimuksiin (Kananen 2017, 11). Näiden seikkojen perusteella tutkimusta voidaan siis pitää kehittämistutkimuksena.

2.4 Aineistonkeruu

Ensisijainen eli primäärinen tutkimusaineisto kerätään tutkimusongelmaa varten (Kananen 2012, 44). Tämän opinnäytetyön kohdalla primääriaineiston kerääminen tapahtuu tutkimuksen aikana strukturoimattomalla havainnoinnilla sekä kenttämuistiinpanoilla. Strukturoimaton havainnointi on tyypillinen menetelmä, kun tavoitteena on hankkia mahdollisimman paljon ennakkotietoa tietystä asiasta. Kun tiedetään mitä ilmiössä oletetusti tapahtuu, voidaan sen teoriaa käyttää apuna strukturoimattomassa havainnoinnissa. (Anttila 1998.) Havainnointi tapahtuu kuvien ja videoiden avulla. Tällainen havainnointi sopii työhön hyvin efektien suunnitteluvaiheen osalta. Sen avulla voidaan tutkia olemassa olevien ilmiöiden avulla hahmojen ja efektien välisiä yhteyksiä sekä erilaisille efekteille ominaisia piirteitä. Menetelmä sopii lisäksi tekniseen suunnitteluun, kun halutaan selvittää pelimoottorin työkalun tarjoamia ominaisuuksia ennen varsinaista teknistä toteutusta.

Sekundääristä aineistoa puolestaan ovat kaikki valmiit dokumentit. Tällaisen aineiston avulla saadaan koostettua työlle teoriapohjaa. Sekundäärisen aineiston kerääminen tapahtuu aiheeseen liittyvien verkkoartikkelien, kuvien ja muiden tallenteiden hankkimisella. (Kananen 2012, 44.)

3 TEOREETTINEN VIITEKEHYS

Teoreettisen viitekehyksen avulla luodaan yleiskuva työn aiheen ympärille. Työ käsittelee pelihahmojen visuaalisia efektejä, joten aiheen ympäriltä löytyy käsiteltävää yleisesti efekteistä, niiden merkityksestä sekä toteutustavoista.

3.1 Visuaalisten efektien merkitys videopeleissä

Alun perin filmiteollisuudesta lähtöisin olevat visuaaliset efektit ovat nykypäivänä yleinen ja merkittävä osa myös videopelejä. Visuaalisilla efekteillä tarkoitetaan tietokoneella generoituja tehosteita, jotka tekevät pelattavuudesta jännittävämpää ja lisäävät pelimaailmaan eloisuutta (The Ultimate Guide to VFX… s.a.). Termistä *visuaalinen efekti* käytetään tavanomaisesti myös lyhennettä VFX. Videopelien visuaaliset efektit kuitenkin eroavat osittain filmiteollisuuden efekteistä. Puhuttaessa visuaalisista efekteistä videopelien yhteydessä voidaan tarkoittaa esimerkiksi partikkeliefektejä, erilaisia simulaatioita sekä videokuvan tehokeinoja. Elokuvateollisuudessa visuaaliset efektit ovat tietokoneella erikseen luotuja tai manipuloituja kuvia, joita voidaan yhdistää näyteltyyn videokuvaan (Marshall s.a.).

Videopelien efektit käyttävät hyväkseen reaaliaikaista renderöintiä eli kuvan piirtämistä näytölle, kun taas filmien efektit ovat esirenderöityjä. Tämä tarkoittaa sitä, että elokuvan efekti on suunniteltu, luotu ja renderöity katsottavaksi ainoastaan tietystä kuvakulmasta. Videopeleissä pelaajalla on usein kuitenkin mahdollisuus tarkastella efektiä useammastakin eri kuvakulmasta reaaliaikaisesti. Suunniteltaessa visuaalisia efektejä videopeleihin on siis huomioitava koko peliympäristö vain yhden kuvakulman sijaan. Toisin kuin elokuvissa, videopelien efektejä kehitettäessä on lisäksi pidettävä mielessä pelin resurssit ja laitteiston rajoitukset, sillä suorituskyvyn ja pelattavuuden on säilyttävä hyvänä useidenkin efektien simuloinnista huolimatta. (The Difference Between Game… s.a.)

Nykypäivänä videopelien yhteydessä puhutaan paljon immersiivisyydestä. Sillä tarkoitetaan pelin kyvykkyyttä saada pelaaja irtautumaan hetkeksi todellisesta elämästä ja uppoutumaan pelitodellisuuteen (Wirtz s.a.). Pelin immersiivisyyteen vaikuttaa moni tekijä, joista yksi on juuri visuaaliset efektit. Niiden avulla voidaan luoda peliin uskottavuutta ja realistisempaa vaikutusta.

3.2 Visuaalisten efektien tyypit

Visuaaliset efektit toimivat yläkäsitteenä monille erityyppisille efekteille. Efektien tyyppejä voidaan kuitenkin jaotella tiettyihin pääryhmiin: pelattavuuden, ympäristön ja käyttöliittymän efekteihin sekä elokuvallisiin efekteihin. Pelattavuuden efekteihin lukeutuvat muun muassa räjähdykset ja partikkelit. Tällaiset efektit ovat usein dynaamisia ja interaktiivisia, kuten hahmon tekemät iskut. Ympäristön efektejä ovat esimerkiksi sumu, sade ja lumi, jotka sijoittuvat globaalisti pelimaailmaan tarkoituksenaan lisätä tunnelmaa. Efektien hyödyntäminen puolestaan käyttöliittymässä voi näkyä esimerkiksi painikkeiden animaatioissa ja siirtymissä. Peleissä voidaan myös nähdä elokuvamaisia animoituja välikohtauksia, jotka tukevat tarinankerrontaa, mutta eivät sisällä pelattavuutta. Tällaisiin välikohtauksiin voidaan lisätä elokuvallisia efektejä. (Mad VFX 2023.) Muita peleissä tavattuja visuaalisia efektejä ovat muun muassa nesteiden tai kankaiden käyttäytymisen simulointi, valaistuksen efektit sekä erilaiset liikkeen tehokeinot ja etäisyyksien tarkennukset (Veselinovikj 2024).

Erilaisilla peligenreillä voi olla omia vaatimuksia pelin visuaalisten efektien tyyleihin. Esimerkiksi toimintapeleille tyypillisiä efektejä ovat intensiiviset räjähdykset ja aseiden tulitus, jotka aiheuttavat pelaajassa jännitystä ja adrenaliinin tunnetta. Roolipeleille luonteenomaisia efektejä ovat puolestaan erilaiset valoefektit, kuten loitsut ja taiat sekä ympäristön efektit. Pulmapeleissä efektien tarkoitus on parantaa pelikokemusta hienovaraisesti esimerkiksi yksinkertaisemmilla partikkeliefekteillä. (The Ultimate Guide to VFX… s.a.)

Tässä opinnäytetyössä visuaalisten efektien keskiössä on pelihahmo, joka on myös avainasemassa pelattavuuden kannalta. Näin ollen työn aikana kehitettävät efektit ovat tyypiltään siis pelattavuuden efektejä, mikä tarkoittaa, että työn osalta keskitytään pääsääntöisesti partikkeliefekteihin. Lisäksi valitut

efektit sisältävät monia maagisia elementtejä, joten ne sopisivat parhaiten fantasia- ja roolipelien kategoriaan.

3.3 Visuaalisten efektien tyylilajit

Peligenre, grafiikkataiteen tyyli sekä pelin yleinen sävy vaikuttavat efektien tyylilajiin, sillä pelin kaikkien tyylien on oltava johdonmukaiset ja yhteensopivat. Erilaisia efektien tyylilajeja ovat muun muassa realismi, osittainen realismi sekä tyylitellyt efektit (Mad VFX 2022).

Realistiset efektit pyrkivät olemaan mahdollisimman todenmukaisia ja jäljittelemään todellisia ilmiöitä. Realististen efektien toteuttamisessa käytetään usein fyysisiä simulaatioita oikeanlaisten muotojen, uskottavuuden, liikkeen sekä yksityiskohtaisuuden saavuttamiseksi. (Mad VFX 2022.) Realististen efektien kehittämiseksi on tutkittava oikeita ilmiöitä todellisen yksityiskohtaisuuden aikaansaamiseksi. Realistisia efektejä on nähtävissä esimerkiksi Red Dead Redemption 2 -pelissä (kuva 1).

Kuva 1. Red Dead Redemption 2 -pelin realistinen efektityyli (Steam 2019)

Myös osittain realistiset efektit pyrkivät kuvaamaan ilmiöitä todenmukaisesti, mutta lisäksi ne pitävät sisällään olemassa olemattomia ilmiöitä, kuten taikuuden. Osittain realististen efektien avulla voidaan luoda vääristynyttä todellisuuden tuntua ja niillä painotetaan enemmän tietynlaisen tunnelman luomista. (Mad VFX 2022.) Esimerkiksi God of War -pelisarjan peleissä esiintyvät efektit voidaan luokitella osittain realistisiin efekteihin (kuva 2).

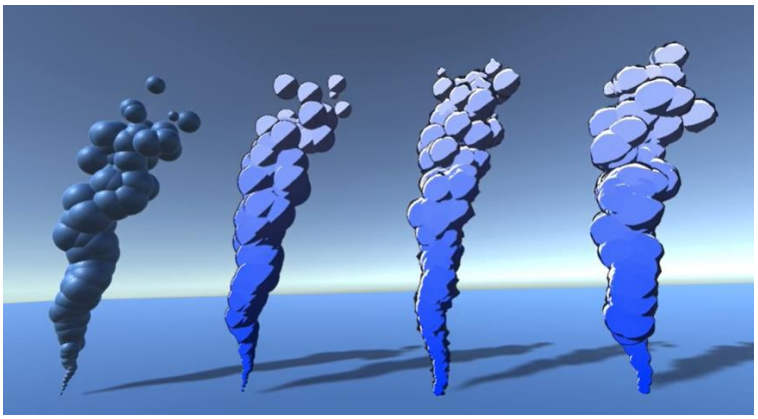
Kuva 2. Osittain realistinen efektityyli God of War Ragnarök -pelissä (Steam 2024)

Tyylitellyt efektit ovat hyvin vahvasti sidonnaisia pelin muuhun taiteelliseen tyyliin, ja efektien tyylin on sovittava pelin yleiseen tyylilajiin. Tyyliteltyihin efekteihin lukeutuu monia erityylisiä efektejä. Tällaisia efektejä ovat muun muassa pikselimäiset efektit, piirretyt 2D-animaatioefektit tai Cel-varjostetut efektit (Mad VFX 2022). Esimerkiksi pikselitaide on 2D-pelien grafiikkatyyli, jossa kuvat rakentuvat pienistä neliömäisistä pikseleistä. Tällainen grafiikkatyyli on edustettuna muun muassa pelissä nimeltä Enter the Gungeon, jonka efektit ovat myös tyylitellysti pikselimäisiä (kuva 3).

Kuva 3. Enter the Gungeon -pelin pikselimäinen efektityyli (Steam 2016)

Cel-varjostuksella tarkoitetaan tekniikkaa, jolla yritetään saada 3D-mallit näyttämään 2D-piirroksilta. Tekniikan avulla pyritään usein saamaan aikaan piirrosmainen efekti, joka kuitenkin käyttäytyy kolmiulotteisesti. (Mad VFX 2022.)

Cel-varjostuksella 3D-malliin voidaan lisätä valaistusta ja varjostuksia syvyyden korostamiseksi. Tekniikan avulla voidaan näin toteuttaa esimerkiksi sarjakuvamaisempia piirrosefektejä 3D-malleja hyödyntäen (kuva 4). Cel-varjostuksen tekeminen voi myös olla yksinkertaisempaa verrattuna käsintehdyn sarjakuvamaisen 2D-animaatioefektin tekemiseen.



Kuva 4. Cel-varjostettu efekti (Reddit 2019)

Tämän opinnäytetyön aikana kehitettävissä efekteissä tavoitellaan tyylittelyn sekä osittaisen realismin yhdistävää tyylilajia. Osittainen realismi tulee esille efektien vaiheiden ajoituksessa ja käyttäytymisessä, kun taas tyylittely näkyy efektien muodoissa ja komponenteissa.

3.4 Visuaalisten efektien perusteet

Visuaalisten efektien toteuttamiseen ei ole vain yhtä oikeaa toimintatapaa. On kuitenkin olemassa tiettyjä raameja, joita suunnittelussa ja toteutuksessa tulee noudattaa. Tässä luvussa keskitytään efektien suunnittelun teoriaan sekä lisäksi tarkastellaan Unreal Engine -pelimoottorin efektityökalun rakennetta.

3.4.1 Suunnittelu

Muodot, värit ja ajoitus ovat keskeisimpiä suunnitteluperiaatteita visuaalisten efektien kohdalla ja niiden huolittelulla on suuri merkitys efektin ymmärrettävyyden kannalta. Efektin rakenteen suunnittelulla voidaan painottaa tiettyjä elementtejä ja tasapainottaa siten efektiä. Myös efektin voimakkuuden tasolla on merkitystä sen näyttävyyden kannalta, sillä kaikki efektit eivät kuvasta sa-

manlaista voiman suuruutta. Pelejä varten viimeistellyt efektit lisäksi usein reagoivat pelimaailman kanssa eri tavoin. Tämän työn aikana efektien suunnittelussa ei kuitenkaan huomioida niiden vuorovaikutusta peliympäristön kanssa.

Tässä luvussa on tutkittu League of Legends -pelin kehittäjien kokoamaa ohjeistusta visuaalisten efektien kehitykseen. Ohjeistus on suunnattu efektien kehittämiseen juuri League of Legends -pelille, mutta ohjeita on mahdollista silti soveltaa yleisesti efektien suunnitteluun.

Efektin rakenne

Efekti voidaan jakaa primäärisiin ja sekundäärisiin elementteihin. Efektin primäärinen elementti palvelee hahmon liikkeen päätavoitetta ja kommunikoi eniten ja selkeinten pelattavuuden kanssa. Primäärisillä elementeillä on hyvä olla selkeät ääriviivat, vahvat muodot ja kontrasti, korkea läpinäkymättömyys sekä intensiivinen liike. Sekundääriset elementit puolestaan tukevat primäärisiä elementtejä ja niille on ominaisempaa sumeammat ääriviivat, yksinkertaiset muodot, läpinäkyvyys ja hienovaraisempi liike. (Riot Games 2017.) Jos primääriset ja sekundääriset elementit eivät ole tasapainossa, efekti voi ottaa liikaa visuaalista valtaa pelissä. Riippuu toki paljon pelin teemasta ja visuaalisesta ilmeestä, kuinka näyttäviä efektejä siihen voidaan sovittaa. Yleisenä käytäntönä efektien tulisi kuitenkin olla tasapainossa pelin muiden visuaalisuuksien kanssa, ja tätä voidaan säädellä juuri primääristen ja sekundääristen elementtien keinoin.

Pelihahmon hyökkäyksellä on aina tietyn kokoinen vaikutusalue, mikä aiheuttaa esimerkiksi vahinkoa toisille hahmoille. Tässä nousee esiin efektin kommunikoinnin tärkeys pelattavuuden kannalta, sillä efektin on täsmällisesti otettava tämä vaikutusalue haltuun. Mikäli efekti on esimerkiksi huomattavasti pienempi kuin hyökkäyksen varsinainen vaikutusalue, se antaa virheellistä tietoa pelaajalle aiheuttaen mahdollisesti tälle tahatonta vahinkoa (Riot Games

2017). Pelaajan on mahdoton tietää hyökkäyksen näkymättömiä rajoja, minkä vuoksi efektin on osoitettava rajat selkeästi.

Efektin voimakkuuden taso

Hahmoilla on usein eritasoisia kykyjä. Pelin alussa hahmolla on yleensä joitakin perustason kykyjä ja pelin edetessä se voi saada itselleen voimakkaampia kykyjä. Efektin visuaalisen tyylin on vastattava sen merkityksen ja voimakkuuden tasoa (Riot Games 2017). Peruskyvyt ovat yleensä heikompia, jolloin niiden visuaalinen ilme on myös pidettävä yksinkertaisena. Hahmon voimakkaampien kykyjen on puolestaan oltava paljon näyttävämpiä, jotta voiman suuruus tulee esille paremmin. Tällöin pelaaja myös ymmärtää sen olevan vahvempi kyky. Efektien visuaalisuuksien välillä on siis oltava selkeä ero perustuen niiden voimakkuuden tasoon.

Muodot

Efekti voi koostua lähes minkälaisista muodoista tahansa, mutta muotojen käyttöä on yhtenäistettävä merkitsemään tiettyjä asioita pelin eri puolilla. Johdonmukaisuus efekteissä auttaa ymmärtämään niiden tarkoitusta paremmin. Esimerkiksi terävät muodot yhdistetään usein vaarallisiin ja vahingoittaviin asioihin, kun taas pyöreät muodot sopivat suojaavaan efektiin paremmin. (Riot Games 2018.) Kuvassa 5 on havainnollistettu muotojen ryhmittelyä eri merkityksin.

Kuva 5. Muotojen esimerkkiryhmittely (Riot Games 2018)

Vihreässä laatikossa on terveyteen yhdistettäviä symboleja, joista plusmerkki on yksi yleisin peleissä käytetty pelihahmon terveydentilan symboli. Keltaiseen laatikkoon on sisällytetty vahingoittamiseen liitettyjä piikikkäitä symboleja, ja

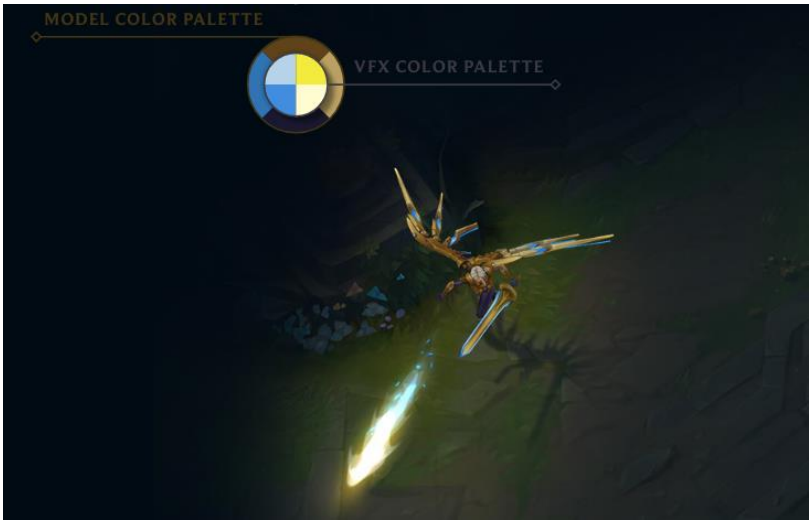
vaaleanpunaisessa laatikossa on suojaukseen yhdistettyjä turvallisia pyöreitä muotoja. Johdonmukaisten muotojen avulla voidaan erottaa samankaltaisen värimaailman omaavia efektejä toisistaan, jos efektejä on pelissä useita samanaikaisesti näkyvillä.

Värit

Erityisesti värien kohdalla nousee esiin efektin kommunikointi ja informaation välittäminen pelaajalle (Riot Games 2018). Eri värit voivat aiheuttaa ihmisissä tiettyjä yleisiä psykologisia reaktioita ja mielleyhtymiä. Esimerkiksi punainen väri yhdistetään usein vaaraan, kun taas sininen on rauhoittavampi väri. (Archetti 2024.) Värit siis helpottavat pelaajaa ymmärtämään efektin merkitystä, vaikka sitä ei suoranaisesti tietäisi. Kuvassa 6 on eritelty eri väreihin liitettyjä tyypillisiä tunnereaktioita.

Kuva 6. Yleisiä väreihin liitettyjä tunnetiloja (mukaillen Archetti 2024)

Lisäksi efektin värien yhtenäistäminen pelihahmon omiin perusväreihin (kuva 7) helpottaa efektien tunnistamista varsinkin sellaisissa peleissä, joissa pelihahmoja ja efektejä on lukuisia erilaisia samanaikaisesti näkyvissä. Tällöin pelin aikana on selkeämpi hahmottaa mille hahmolle mikäkin efekti kuuluu, ja mistä hyökkäys tulee. (Riot Games 2018.)



Kuva 7. Hahmon ja efektin värien suhde League of Legends -pelissä (Riot Games 2017)

Efektin värimaailman tulisi ideaalisesti sisältää verrattain yhteneviä värejä. Värien vastakkainasettelu tuo toki efektiin lisää mielenkiintoa, mutta aiheuttaa myös kilpailua värien merkitysten välillä hankaloittaen efektin primäärisen ja sekundäärisen elementin erottamista. Sekundäärisen elementin värin on oltava himmeämpi ja läpinäkyvämpi, kun taas primäärisen elementin on otettava osansa voimakkaammin haltuun. (Riot Games 2017.)

Ajoitus

Visuaalisen efektin komponenttien ajoitus on tärkeä periaate realistisemman vaikutelman aikaansaamiseksi. Ajoituksessa on tärkeää välttää lineaarisuutta. Esimerkiksi räjähdyksen kesto on harvoin suoraviivaista, vaan liike on aluksi nopeaa ja hidastuu loppua kohti. (Riot Games 2018.) Efekteillä tulisikin täten olla valmistautumis- ja hälvenemisvaihe huippukohdan lisäksi. Tapa, jolla efekti muuttuu sen elinajan aikana, tarjoaa olennaista tietoa sen toiminnallisuudesta. (Riot Games 2017.) Hälvenemisvaihe kertoo pelaajalle, että esimerkiksi vahinkoa tekevä hyökkäys on hiipumassa pois. Mikäli efekti alkaa ja loppuu äkillisesti, pelaaja ei ehdi valmistautua siihen, ja visuaalisesti se saa efektin myös näyttämään kankealta.

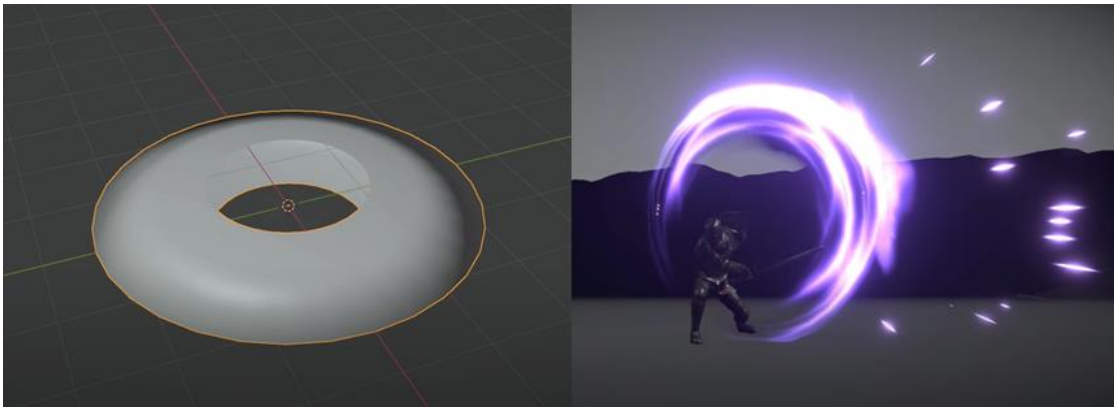
3.4.2 Toteutustekniikat

Eräs yleisimmin käytetty tekniikka visuaalisten efektien luomiseksi on partikkelisysteemit. Partikkelisysteemit ovat pohjimmiltaan useiden yksittäisten partik-

kelien kokoelmia. Lopullisen efektin aikaansaamiseksi jokainen partikkeli luodaan, animoidaan ja renderöidään erikseen. Partikkelisysteemien avulla voidaan luoda moniakin erilaisia efektejä, kuten tuli-, savu-, pöly- sekä räjähdysefektejä. (Madeinshoreditch 2023.)

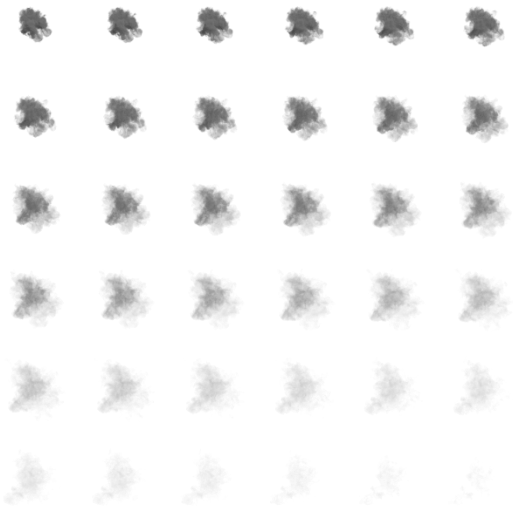
Visuaalisten efektien toteuttamiseksi tarvitaan usein myös 3D-malleja, 2D-tekstuureja sekä shadereita. Shaderit ovat pieniä ohjelmia, joilla voidaan saada aikaan ainutlaatuisia tyylejä ja tehosteita, joita ei muilla tekniikoilla pystytä saavuttamaan. Yleisesti ottaen shadereita käytetään muun muassa erityisten tekstuurien ja materiaalien luomiseen videopeleissä sekä valon heijastusten ja taittumisen simulointiin. (Madeinshoreditch 2023.)

3D-malleja voidaan hyödyntää efekteissä esimerkiksi ammuttavina projektiileina tai antamaan efektille tietynlainen muoto. Esimerkiksi kuvassa 8 on ensin mallinnettu kehämäinen 3D-malli, jota hyödyntämällä efekti saa muotonsa ja liikkumapintansa kuvastamaan pelihahmon miekaniskua.



Kuva 8. Kehämäinen 3D-malli (vas.) ja valmis efekti mallia hyödyntäen (oik.) (mukaillen Aguiar 2022)

Animoiduilla 2D-tekstuureilla pyritään monesti saamaan efektissä aikaan liikettä. Esimerkiksi animoitu 2D-savutekstuuri sisältää monta kuvaa savusta sen eri vaiheissa (kuva 9). Savuefekti voidaan sitten toteuttaa partikkelisysteeminä 2D-kuvien sarjalla, jotka toistetaan peräkkäin.

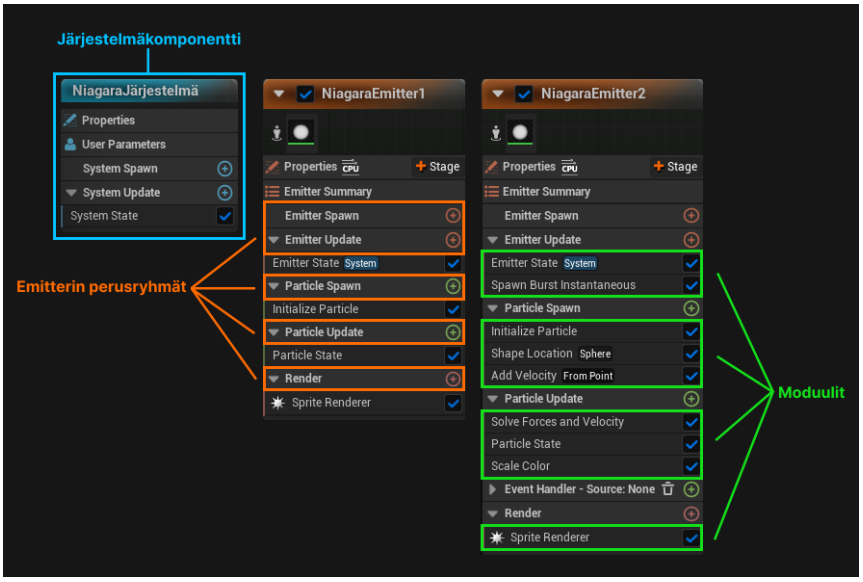


Kuva 9. Animoitu savutekstuuri (Aguiar 2023)

Animoitu tekstuuri luo enemmän realistisuutta efektiin. Liikettä voi kuitenkin luoda yksinkertaistakin kuvatekstuuria hyödyntäen partikkelien arvoja satunnaistamalla. Työn aikana ei pyritä jäljittelemään ilmiöitä realistisesti animoiduilla tekstuurisarjoilla, vaan aikomuksena on hyödyntää yksinkertaisten partikkelien satunnaistamista realistisemman vaikutelman luomiseksi.

3.5 Niagara-järjestelmä

Tässä opinnäytetyössä visuaalisten efektien kehitysympäristönä toimii Unreal Engine 5 -pelimoottori, tarkemmin versio 5.4.2. Pelimoottorin ensisijainen työkalu visuaalisten efektien kehittämiseksi on Niagara VFX -järjestelmä. Järjestelmä sisältää 4 ydinkomponenttia: järjestelmäkomponentin, emitterit, moduulit ja parametrit (Niagara Overview s.a.). Pelimoottori sisältää myös vanhemman Cascade-työkalun partikkelisysteemien tekemiseen, mutta tässä työssä keskitytään ainoastaan Niagara-työkaluun. Järjestelmä pitää sisällään kaiken tarpeellisen, mitä efektin rakentamiseen tarvitaan. Järjestelmä koostuu erilaisista rakennuspalikoista, joiden avulla lopullinen efekti saadaan toteutettua. Järjestelmän asetuksia voi säätää järjestelmäkomponentista käsin. Tällöin muutokset vaikuttavat efektiin kokonaisvaltaisesti. (Niagara Overview s.a.) Kuvassa 10 on selkeytetty työkalun rakennetta ja komponentteja.



Kuva 10. Niagara-järjestelmän komponentit

Emitterillä tarkoitetaan partikkelien lähdettä, josta partikkelit saavat alkunsa. Se sisältää eri vaiheita, ja siihen on mahdollista lisätä myös uusia vaiheita. Partikkelien syntyminen määritellään Emitter Update -vaiheessa. Particle Spawn -vaiheessa partikkeleille annetaan alustavat arvot. Tässä alustusvaiheessa voidaan asettaa partikkeleille esimerkiksi alustava koko, väri, elinaika, suuntautuminen ja rotaatio. Particle Update -vaiheeseen, eli partikkelien päivitysvaiheeseen lisätyt moduulit puolestaan vaikuttavat siihen, mitä partikkeleille tapahtuu niiden elinaikana. Render-vaihe määrittää miltä partikkelit näyttävät.

Emitteri siis säätelee partikkelien syntymistä, kuinka ne käyttäytyvät, miltä ne näyttävät ja mitä niille tapahtuu elinaikanaan. Emitterin vaiheisiin lisätään päällekkäin erilaisia moduuleita, jotka toteuttavat yksilöllisiä tehtäviä. Moduulit ovat efektin perusrakennuspalikoita ja niiden järjestyksellä on merkitystä, sillä ne käsitellään ylhäältä alas. (Niagara Overview s.a.) Moduuleita voidaan pitää tietynlaisina funktioina. Moduuliin syötetään partikkelien dataa, jota muokataan eri tavoin moduulin sisällä. Tämän jälkeen muokkauksen tulos siirtyy ulos moduulista, yleensä toiseen moduuliin. Moduuleihin voidaan syöttää erilaisia parametrejä, jotka toimivat datan abstraktioina Niagara-järjestelmässä (Niagara Overview s.a.). Parametrejä voi myös luoda itse lisää omalla välilehdellään Niagara-työkaluikkunan näkymässä. Esimerkiksi värille voi luoda erillisen parametrin ja asettaa sille jonkin tietyn oletusarvon. Luotua parametriä voi myöhemmin hyödyntää kaikkialla, missä tiettyä väriä halutaan käyttää. Näin samaa väriä ei tarvitse säätää uudestaan jokaiseen moduuliin.

4 EFEKTIT

Efektien kehittäminen alkaa niiden suunnittelusta. Ennen hahmon efektin suunnittelun aloittamista, on kuitenkin tiedettävä millaiseen animaatioon ja millaiselle hahmolle efekti tullaan kehittämään. Hahmon spesifikaatiot voivat määrittää efektien tyylejä esimerkiksi aseiden, värien ja teemojen mukaan. Koska tässä opinnäytetyössä ei keskitytä hahmon animointiin, efektit toteutetaan siten, että ne sopivat ennalta valikoituihin animaatioihin. Pelihahmojen taisteluliikkeiden ja kykyjen animaatiot ovat usein yksittäin melko lyhyitä, joten efekteillä voidaan laajentaa animaatiosta saatavaa liikkeen käsitystä.

Efektin toteuttamista varten on lisäksi tärkeä tietää, minkälaista tyyliä niissä tavoitellaan. Suunnitelmien tuominen näkyville piirretyn 2D-kuvamateriaalin muodossa esimerkiksi paperille tai digitaaliselle kankaalle auttaa hahmottamaan efektin kokonaiskuvaa ja eri osia, joista se koostuu. Suunnitelmien pohjalta tuotettua havainnollistavaa kuvamateriaalia kutsutaan konseptitaiteeksi. Sen tarkoituksena on yksinkertaisesti visualisoida efektiä kehityksen esivaiheessa. Lisäksi suunnitteluprosessiin olisi hyvä sisällyttää efektin tekninen suunnittelu, jonka avulla voidaan pohtia mahdollisia toteutustapoja efektille ennen kuin sitä lähdetään varsinaisesti toteuttamaan.

Konseptitaide

Kun efektin pääpiirteittäisestä suunnitelmasta on saatu muodostettua käsitys, voidaan sen pohjalta alkaa kehittämään konseptimateriaalia. Jotta efektin konsepti saadaan sovitettua hahmon animaatioon, on konseptikuvaan tärkeä sisällyttää hahmon animaation avainkohdat tai -asennot, jotka ovat efektin suunnittelun kannalta merkittävimpiä. Tällaisia avainkohtia ovat ne, joissa tapahtuu jotain uutta efektin kannalta. Esimerkiksi ennakoiva liike ja kohokohta ovat sellaisia animaation kohtia, joissa myös efektin vaiheet ovat erilaisia. Efektillä tulisikin olla valmisteluvaihe, kohokohta sekä hälvenemisvaihe.

Ennakoiva liike tapahtuu ennen merkityksellisempää liikettä. Tällaista ennakoivaa liikehdintää voi olla esimerkiksi hahmon käden liikkuminen taaksepäin

ennen varsinaisen iskun suorittamista. Ennakoiva liike ilmaisee hahmon valmistautumista toimintaan, mikä myös antaa pelaajalle tietoa siitä, mitä seuraavaksi on tapahtumassa. Samassa yhteydessä on sopiva hetki aloittaa visuaalisen efektin esittäminen. Ennen voimakkaan iskun tekemistä, hahmon ase voi esimerkiksi alkaa hehkua. Tällöin pelaaja ymmärtää miekan keräävän voimaa tulevaa iskua varten. Animaation kohokohta on puolestaan sen voimakkain hetki, jota efektillä halutaan painottaa. Miekan iskeytyminen kohteeseen aiheuttaa voimakkaimman vaikutuksen, jota painotetaan intensiivisemmällä efektillä. Efektin on myös hiivuttava pois aikanaan, sillä kykyjen vaikutukset eivät ole ikuisia.

Hahmon yksityiskohtiin ei ole tarpeellista kiinnittää huomiota efektin konseptia piirrettäessä. Liiallinen yksityiskohtiin panostaminen hahmon kohdalla voi viedä huomiota pois varsinaisen efektin konseptista, jolloin suunnitelman seuraaminen toteutusvaiheessa voi aiheuttaa epäselkeyksiä. Tämän vuoksi ainoastaan hahmon yksinkertaisen siluetin piirtäminen on sopivampi menettely, jolloin konseptitaide keskittyy olennaisimpaan suunnittelun kohteeseen.

Efektien toteuttaminen

Luotaessa uutta efektiä Unreal Engineen on vaihtoehtona valita Niagara-emitteri tai Niagara-järjestelmä. Niagara-emitteri on nimensä mukaan vain yksittäinen emitteri. Se ei toimi itsenäisenä efektinä, eikä sitä voi suoraan käyttää sellaisenaan. Toimiakseen efektinä erillinen Niagara-emitteri on aina sisällytettävä Niagara-järjestelmään. Niagara-järjestelmään voi luoda emittereitä myös suoraan. Erillisen Niagara-emitterin tekeminen voisi olla hyödyllistä, mikäli esimerkiksi useampi eri efekti käyttäisi osanaan jotain tiettyä, samaa emitteriä. Tällöin useaan Niagara-järjestelmään voidaan liittää sama yksittäinen emitteri, eikä sitä tarvitse tehdä erikseen uudestaan jokaiseen järjestelmään.

Järjestelmään voi sisällyttää useita emitterikomponentteja ja ne kaikki hoitavat oman osansa kokonaisuuden aikaansaamiseksi. Efektin osat täytyy monesti jakaa useampaan Niagara-järjestelmään. Samaan järjestelmään voi kuitenkin sisällyttää sellaiset efektin osat, jotka ovat liikkeeltään, sijainniltaan tai ajoitukseltaan samankaltaisia. Esimerkiksi tuliefektin liekit, savu ja kipinät voitaisiin tehdä omina emittereinään saman Niagara-järjestelmän sisälle.

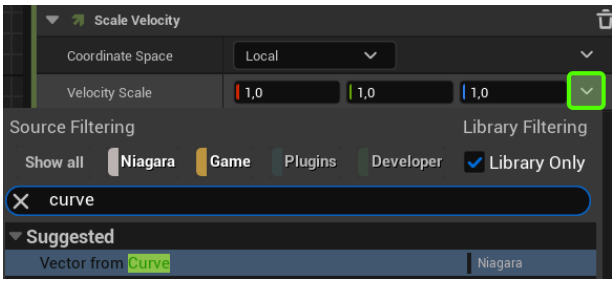
Yleiset moduulit

Partikkelien syntymistä säädellään tyypillisesti Spawn Rate tai Spawn Burst Instantaneous -moduulin avulla. Spawn Rate -moduulilla partikkeleita syntyy tasaisesti ajan kuluessa. Spawn Burst Instantaneous -moduuli puolestaan synnyttää kaikki partikkelit samalla kertaa. Location-tyyppiset moduulit määrittävät alustusvaiheessa partikkelien syntymispisteen. Esimerkiksi Shape Location -moduulin avulla partikkelit syntyvät valitun primitiivisen muodon mukaan. Primitiivisiä muotoja on valittavana vain muutama, mutta puolestaan Static Mesh Location -moduulin avulla voi muodoksi asettaa minkä tahansa 3D-mallin. Partikkelit syntyvät satunnaisiin pisteisiin muodon määrittämien rajojen sisälle. Muita moduuleja partikkelien syntymispisteen määrittämiseksi ovat muun muassa System Location ja Skeletal Mesh Location.

Toisinaan 2D-partikkelien suuntautumista on säädettävä, sillä niiden halutaan ehkä suuntautuvan aina kameraan päin, tai pysyvän maanpinnan suuntaisena. Tämä tapahtuu partikkelien alustusvaiheessa ja sitä varten on olemassa Sprite Facing and Alignment -moduuli. Samaan tapaan 3D-partikkelien suuntautumista säädetään Initial Mesh Rotation -moduulilla.

Partikkelien päivitysvaiheessa alustettuja arvoja voidaan muokata ja muutokset ilmenevät niiden elinaikana. Esimerkiksi nopeutta ja väriä voidaan säätää elinajan kuluessa Scale Velocity ja Scale Color -moduuleilla. Lisättäessä skaalaavia moduuleja päivitysvaiheeseen on pidettävä mielessä partikkelien alustavat arvot. Tämä tarkoittaa, ettei esimerkiksi nopeutta voida skaalata päivitysvaiheessa, ellei partikkeleille ole asetettu alustavaa nopeutta. Päivitysvaiheeseen voidaan lisätä muitakin kuin skaalaavia moduuleja. Tällaisia ovat muun muassa erilaisten voimien moduulit. Esimerkiksi pyörimisliike saadaan aikaan 3D-partikkeleille Mesh Rotation Force -moduulilla, ja 2D-partikkeleille Sprite Rotation Rate -moduulilla.

Useiden parametrien määrittelytapaa voidaan vaihtaa moduuleissa, ja esimerkiksi käyrien hyödyntäminen voi olla monesti kätevämpää etenkin päivitysvaiheen moduulien kohdalla. Kuvassa 11 on havainnollistettu, kuinka parametrien arvot vaihdetaan saamaan arvonsa käyrän mukaan.



Kuva 11. Parametrien määrittelytavan muuttaminen

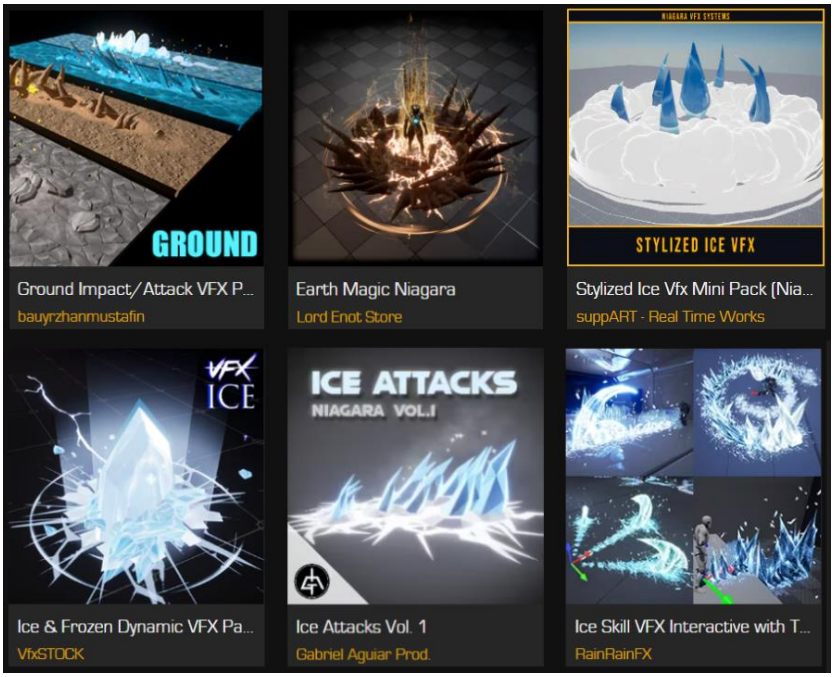
Mikäli parametrien vieressä siis näkyy alaspäin osoittava nuolimerkki, niiden määrittelytapaa on mahdollista vaihtaa. Vaihtoehtoja on useita, mutta tämän työn aikana käyrien hyödyntäminen todettiin yleiseksi ja hyödylliseksi käytännöksi.

Joitakin materiaalien parametrejä voidaan muokata suoraan emitteristä käsin, mikäli partikkelien materiaali sisältää Dynamic Parameter -solmun. Partikkelien päivitysvaiheeseen voidaan silloin lisätä Dynamic Material Parameters -moduuli, minkä kautta tiettyjä materiaalin arvoja saadaan säädettyä moduulin sisällä. Tämä helpottaa työskentelyä huomattavasti, sillä muutokset näkyvät automaattisesti Niagara-järjestelmän esikatseluikkunassa niiden partikkelien kohdalla, jotka hyödyntävät kyseistä materiaalia. Lisäksi materiaalin väriä voidaan säätää emitterissä, jos materiaali sisältää Particle Color -solmun.

4.1 Efekti 1: maahan kohdistuva isku

Maaisku on pelikentän maa-alueeseen kohdistuva hahmon hyökkäysliike. Tämänkaltaisissa efekteissä pelihahmon iskiessä maata, iskukohtaan aiheutuu esimerkiksi halkeamia, murtumia tai maan osittaista nousemista. Iskun efekti voi olla käytännössä mitä tahansa materiaalia tai ainesta, kuten esimerkiksi kivikkoa, hiekkaa tai jäätä. Toteutettavalle efektille valittiin materiaaliksi jää.

Efektin suunnittelua varten havainnoitiin Epic Games -kauppapaikalle julkaistuja maahan kohdistuvia iskuefektejä, joiden avulla voitiin alustavasti selvittää samantyyppisille efekteille yleisiä piirteitä. Kuvaan 12 on koostettu muutama esimerkki.



Kuva 12. Esimerkkikooste maahan kohdistuvista jäisistä iskuefekteistä

Kuten kuvasta voidaan havaita, piikikäs ulkomuoto on hyvin tavanomainen tämän tyyppisille efekteille etenkin jään kohdalla. Valkoinen ja sinertävä väri ovat johdonmukaisia värimaailman valintoja kuvastamaan jäätä. Lisäksi pienemmät partikkelit ja sumuisuus ovat useamman esimerkin kohdalla toteutuva käytäntö.

4.1.1 Suunnittelu

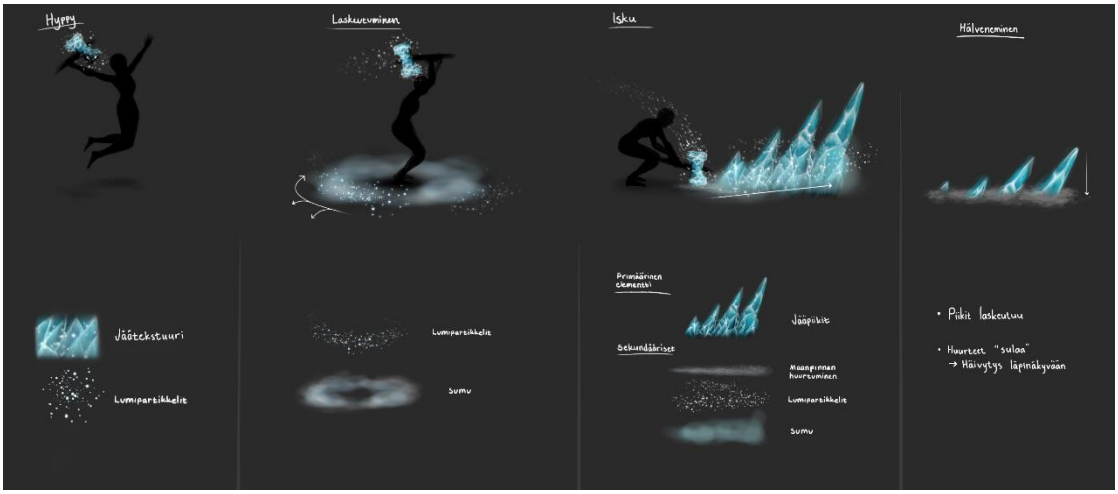
Efektiä varten valittiin animaatio, jossa hahmo juoksee hetkellisesti eteenpäin, hyppää ilmaan ja laskeutumisen jälkeen iskee aseella maanpintaa. Iskun on tarkoitus saada jääpiikit nousemaan hahmon edessä.

Aiemmin esitetystä kuvasta 6 (sivu 18), joka käsittelee värien ja tunnetilojen välisiä yhteyksiä, voidaan havaita valkoisen ja sinisen värin aiheuttavan mieltymyksiä kylmyyteen. Lisäksi jään sinertäminen on luonnossakin tavattu ilmiö. Kyseiset värit siis soveltuvat teemaan loogisesti ja ovat jäisissä efekteissä usein käytettyjä, kuten aiemmin havaintojen pohjalta voitiin todeta.

Koska kyseessä on pelihahmon vahingoittava hyökkäysliike, on efektin indikoitava vahingollisuutta. Täten efektin primäärimuodoksi voitiin valita terävä rakenne, sillä terävät asiat tunnetusti satuttavat. Terävä muoto sopii myös

efektin teemaan realistiselta kannalta, sillä jääpuikot ovat luonnollisestikin teräväkärkisiä.

Kuvassa 13 on esitetty efektin konsepti. Konseptikuvaan on eritelty hahmon avainasennot, ja efektin vaiheet on pyritty sovittamaan niihin. Valmistautuminen iskuun näkyy hahmon aseessa, joka alkaa animaation alkuvaiheessa jäätyä ja säteillä pieniä partikkeleita. Animaation keskivaiheella hahmo laskeutuu hypystä takaisin maanpinnalle. Hypyn aikana kyvyn voimakkuus on jo ehtinyt kasvaa jonkin verran, joten laskeutumisesta seuraa lyhytkestoinen ilmavirtaus hahmon ympärille tämän korostamiseksi. Kolmas avainasento käsittää kyvyn voimakkaimman kohokohdan, eli sen hetken, kun hahmo iskee aseella maanpintaa. Isku saa aikaan vahingoittavien jääpiikkien nousemisen maasta.



Kuva 13. Maahan kohdistuvan iskuefektin konsepti

Hälvenemisvaiheen osalta jääpiikit voisivat esimerkiksi sulaa vesilammikoksi tai pirstoutua jääsirpaleiksi efektin loppuvaiheessa. Kuitenkin yksinkertaisuuden säilyttämiseksi opinnäytetyön rajoissa, jääpiikit laskeutuvat takaisin maan tasalle viimeisessä vaiheessa, kun kyvyn vaikutus lakkaa.

Hahmon siluettien alapuolelle on eritelty, mistä osista kyvyn efekti kokonaisuudessaan koostuu missäkin liikkeen vaiheessa. Osien liikkeen suunnat on merkitty nuolien avulla kuvaan. Esimerkiksi jääpiikit nousevat aina toinen toistaan kauemmaksi ja suuremmaksi hahmosta poispäin. Efektin osien vaiheittainen erittely helpottaa myös teknistä suunnittelua, sillä osat on toteutettava erikseen.

Efektin primäärinen elementti on iskusta aiheutuvat jääpiikit. Jääpiikeillä on vahva, terävä muoto, selvästi näkyvät ääriviivat, kontrastiltaan korkeat väriarvot, eikä niissä ole läpinäkyvyyttä. Lisäksi jääpiikkien liike on intensiivistä ja niiden merkitys kyvyn kannalta on suurin, sillä ne ovat vahinkoa aiheuttava elementti. Piikkien ympärillä näkyvä luminen sumu, on puolestaan efektin sekundäärinen elementti. Sumun ääriviivat ovat hatarat, se on osittain läpinäkyvää, ja sen merkitys on ainoastaan korostaa jääpiikkien nousemista. Efektin kaikki muutkin osat ovat sekundäärisiä elementtejä, jotka valmistelevat ja tukevat primäärielementtiä liikkeen aikana.

Tekninen suunnittelu

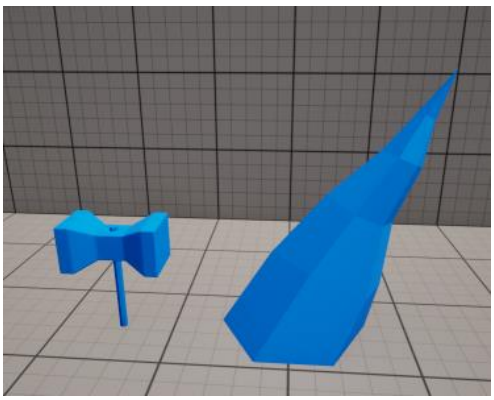
Efekti vaatii jäisen materiaalin jääpiikeille sekä myös aseelle. Jääpiikit ja ase on puolestaan ensin mallinnettava 3D-mallinnusohjelmalla. Aseen jäätyminen voidaan toteuttaa Unreal Enginessä hyödyntämällä dynaamista materiaali instanssia, eikä sen toteutus vaadi erityistä Niagara-järjestelmää. Aseesta ulospäin lentävät lumipartikkelit puolestaan vaativat. Lumipartikkeleiden nopeutta voidaan kasvattaa niiden alkupisteestä nähden, jolloin ne lentävät ulospäin. Jotta partikkelit seuraavat asetta jäljessä liikkeen aikana, on niiden oltava World Space -koordinaatistossa.

Sumuefektit voidaan toteuttaa yksinkertaisilla 2D-tekstuuripartikkeleilla, jotka ovat aina kameraan päin kääntyneenä. Sumupartikkelien rotaatiota voidaan satunnaistaa ja niille voidaan määrittää hyvin pieni pyörimisliike kaavamaisen toistuvuuden välttämiseksi. Laskeutumisvaiheen sumu- ja lumipartikkelien ympyräliikkeen toteuttamiseksi voidaan mahdollisesti hyödyntää Rotate Around Point -moduulia.

Iskuvaiheen efekti vaatii 2D- ja 3D-partikkeleita. Piikkien kohoaminen ja laskeutuminen voidaan toteuttaa 3D-partikkelien koon skaalaamisella. Maanpinnan huurtumiseen piikkien alla on mahdollista hyödyntää Location Event -tapahtumaa, jolloin 2D-huurrepartikkelit asettuvat aina jääpiikkien sijaintiin.

4.1.2 Toteutus

Toteutus aloitettiin tarvittavien yksinkertaisten 3D-mallien tekemisellä Blender-mallinnusohjelmalla. Efekti hyödyntää 3D-mallia jääpiikkien renderöintiin. Tämän lisäksi hahmolle mallinnettiin hyvin yksinkertainen ase, jota ei tullut Mixamosta valmiina ladatun 3D-hahmon mukana. Kuvassa 14 on esitetty efektiä varten tehdyt 3D-mallit.



Kuva 14. Aseen ja jääpiikin 3D-mallit

Mallintamisen jälkeen malleille luotiin jäinen materiaali Unreal Enginessä. Materiaalia varten tarvittiin jäätä kuvastava saumaton tekstuuri, joka luotiin piirto-ohjelmalla. Tekstuurin saumattomuuden ansiosta materiaali näyttää jatkuvalta erilaisten objektien päällä.

Iskuvaihe

Efektin kehittäminen aloitettiin haastavimmalta tuntuvasta osasta, eli jääpiikkien kohoamisesta. Jääpiikeille luotiin oma Niagara-järjestelmä. Järjestelmää luotaessa Unreal Engine ehdotti valmiita mallipohjia, mutta tässä tapauksessa pohjaksi valittiin tyhjä emitteri, joka sisälsi ainoastaan perusmoduulit partikkelien luomiseen ja renderöintiin. Oletuksena partikkelit renderöidään kuvina Sprite-renderöijällä. Kun partikkeleina haluttiin käyttää 3D-malleja, oli renderöijän tyyppi vaihdettava Mesh-renderöijään, johon sai valittua halutun 3D-mallin ja materiaalin.

Jotta efektistä saatiin mitään näkyviin Niagara-järjestelmän esikatseluikkunaan, lisättiin Emitter Update -vaiheen alle Spawn Rate -moduuli, joka määrittää kuinka monta partikkelia luodaan sekunnissa. Initialize Particle -moduuli oli

emitterissä valmiina ja siinä voitiin asettaa partikkeleille alustavia arvoja, kuten koko, elinaika ja rotaatio. Partikkelien elinajaksi asetettiin kiinteä arvo, sillä jokaisen partikkelin elinajan haluttiin olevan yhtä suuri. Jääpiikkien kooksi asetettiin aluksi yhtenäinen vakioarvo, mutta myöhemmin huomattiin, ettei se vaikuttanut efektiin halutulla tavalla. Jääpiikkien oli suunnitellusti tarkoitus kasvaa aina toinen toistaan suuremmaksi. Partikkelien koko alustettiin suhteutumaan kasvavan käyrän mukaan. Käyrä asetettiin tarkastelemaan järjestelmän elinikää, ja partikkelin koko valikoituu käyrältä järjestelmän ikään nähden. Mitä vanhempi järjestelmä on siis iältään, sen suurempi seuraavaksi syntyvästä partikkelista kasvaa.

Partikkelien alustusvaiheeseen lisättiin System Location -moduuli, jolloin partikkelit saatiin syntymään järjestelmän sijaintipisteestä. Koska jääpiikkien haluttiin kohoavan jonomaisena asetelmana, asetettiin partikkelien sijainti määräytymään kasvavan käyrän mukaan, jolloin aina seuraavan partikkelin etäisyys järjestelmän sijainnista nähden oli suurempi. Partikkelien kohoaminen toteutettiin niiden päivitysvaiheessa. Kohoamista varten vaiheeseen lisättiin Scale Mesh Size -moduuli, jonka avulla partikkelien koko asetettiin kasvamaan niiden elinaikana.

Jääpiikkien lisäksi iskuvaiheen efektiin tarvittiin vielä luminen sumu sekä maanpinnan huurtuminen. Huurre-efektiä varten oli suunnitelman mukaan tarkoitus hyödyntää emitterin sijaintitapahtumaa, joka loisi huurteen aina jääpiikin sijaintiin. Sijaintitapahtuman avulla huurrepartikkelit syntyivät kuitenkin aina samankokoisina, mutta niiden koon haluttiin sopivan paremmin jääpiikkien kokoon. Tätä ei lopulta ollut mahdollista toteuttaa sijaintitapahtuman avulla. Huurtumista varten jääpiikkien emitteri monistettiin, ja muutettiin renderöijän asetuksia sopimaan 2D-kuvatekstuurille. Muita arvoja muokattiin myös sopimaan 2D-huurrepartikkeleille, mutta partikkelien sijainti sekä liikkeen suunta ja nopeus toimivat samoin kuin jääpiikeillä.

Luminen sumu toteutettiin myös 2D-tekstuurin avulla. Tekstuuripartikkelit alustettiin asettumaan aina satunnaiseen rotaatioon, ja värin läpinäkyvyyttä vähennettiin, jotta partikkelit olisivat läpikuultavia. Lisäksi partikkelien rotaatiota skaalattiin Update Particle -vaiheessa, jolloin ne saatiin tekemään elinaikanaan hidasta pyörimisliikettä realistisemman vaikutuksen aikaansaamiseksi.

Partikkeleille lisättiin lopuksi vielä Add Velocity -moduuli, joka lisäsi partikkelien nopeutta koordinaattiakselien suuntaan. Nopeutta kasvatettiin jääpiikkien etenemissuuntaa kohti sekä myös osittain ylöspäin.

Laskeutumisvaihe

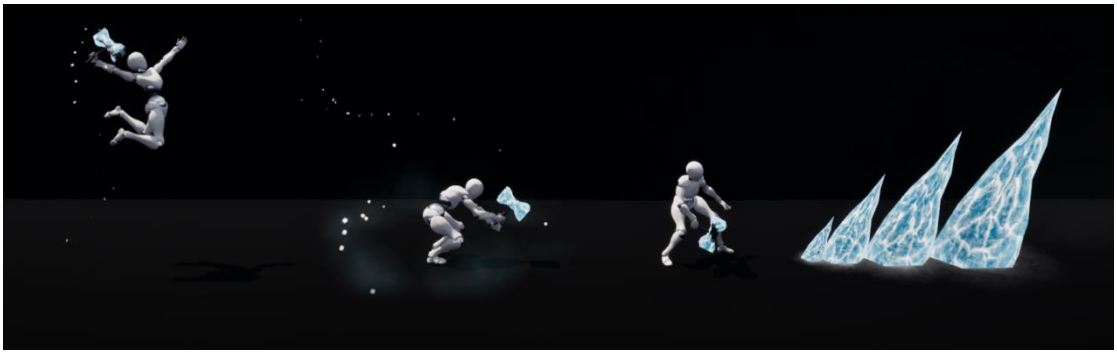
Laskeutumisvaiheen efektin oli tarkoitus kuvastaa hahmoa ympäröivää lumista ilmavirtaa. Toteutuksessa käytettiin tekstuureja sumun ja lumipartikkelien renderöintiin. Aiemmin teknisessä suunnittelussa pohdittiin Rotate Around Point -moduulin käyttöä partikkelien ympyräliikkeen toteuttamiseksi. Konseptissa liikkeen oli ajateltu pyörimisen lisäksi myös ulkonevan hahmosta poispäin. Tätä ei ollut mahdollista toteuttaa edellä mainitun moduulin avulla. Sen sijaan emitteristä löytyi Vortex Force -moduuli, jolla saatiin yksinkertaisemmin aikaan ulkoneva, pyörremäinen liike.

Hyppyvaihe

Hyppyvaiheessa aseen oli tarkoitus jäätyä, ja partikkelien säteillä siitä ulospäin. Aseen jäätymistä ei toteutettu Niagara-järjestelmällä, vaan pelkästään materiaalilla. Materiaalissa hyödynnettiin Noise-, Smoothstep- ja Lerp-solmuja, joiden avulla aseen materiaali saatiin vaihtumaan hiljalleen objektin eri puolilta jäiseksi materiaaliksi. Noise-solmun avulla materiaali saatiin jäätymään laajentuvina saarekkeina objektin eri puolilta. Smoothstepillä saarekkeiden reunoja voitiin häivyttää. Lopuksi kasvattamalla Lerp-solmun, eli lineaarisen interpoloinnin alfa-arvoa animaation aikana, jäinen materiaali saatiin laajenevasti peittämään koko aseen.

Aseesta ulospäin säteileville partikkeleille puolestaan tehtiin oma yksinkertainen Niagara-järjestelmä. Tällä kertaa kokeiltiin hyödyntää valmista suihkulähde-mallipohjaa, sillä se sisälsi valmiiksi joitakin yleisesti tarvittavia moduuleja. Partikkelien nopeuden tilaksi asetettiin *From Point*, jolloin ne liikkuvat keskipisteestä ulospäin satunnaisiin suuntiin. Lisäksi suihkulähde-emitterin mukana tulleet tarpeettomat moduulit poistettiin. Lopuksi partikkelien alustusarvoja säädettiin sopiviksi aseeseen nähden. Jotta partikkelit saatiin jäämään jälkeen aseen liikkeen mukana, emitterin oli oltava World Space -tilassa.

Tämä oli emitterin ominaisuuksissa oletuksena päällä. Lopputulos on nähtävissä kuvassa 15.



Kuva 15. Jäinen iskuefekti

Efekti noudattaa suunnitelmaa melko tarkasti. Kokonaisuudessaan efekti koostuu kolmesta eri Niagara-järjestelmästä. Aseen partikkelit toteutettiin yhden emitterin avulla, kehämäiseen sumuun tarvittiin kaksi emitteriä, ja jääpiikeille tehtiin neljä emitteriä.

4.2 Efekti 2: parannusloitsu

Parannusloitsulla on nimensä mukaan parantava vaikutus. Se on pelihahmon taianomainen kyky, joka palauttaa elinvoimaa. Tyypillisesti parannusloitsu näkyy hahmon kehon ympärillä. Parannusloitsun efektiä varten havainnoitiin Artstation-palveluun julkaistuja efektiesimerkkejä, joista muutamia on koostettu kuvaan 16.

Kuva 16. Esimerkkikooste parannusloitsun efekteistä

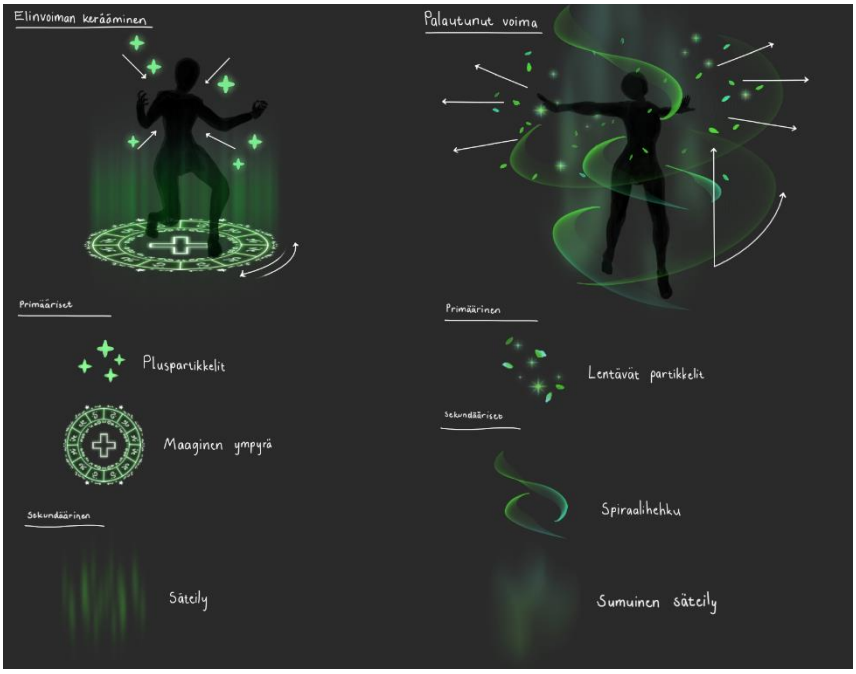
Parannusloitsuille tyypillinen värimaailma tulee koosteesta selkeästi esille. Värit ovat hyvin luonnonläheisiä, sekä lisäksi efekteissä on hyödynnetty muita luonnon elementtejä, kuten soljuvia muotoja ja lehden muotoisia partikkeleita. Esimerkkien avulla havaitaan myös, että parannusloitsuefektit voivat sisältää säihkyviä ja hohtavia elementtejä. Lisäksi loitsuefekteille on yleistä maata vasten näkyvä maaginen ympyräkuvio, kuten muutamassa esimerkkikuvassa näkyy. Parannusloitsuille ominainen symboliikka tuli havainnoinnin aikana vahvasti ilmi. Videopeleissä yleisin terveyteen yhdistettävä symboliikka käsittää plusmerkin muotoiset kuviot, millä on yhteys myös tosielämään.

4.2.1 Suunnittelu

Valitussa animaatiossa hahmo kyykistyy ja vetää käsiään hiljaa yhteen, mikä sopii elinvoiman keräämisen merkiksi. Lopulta hahmo levittää kädet sivulle uusin voimin.

Aiemman havainnoinnin lisäksi värien ja tunnereaktioiden kaaviosta nähdään, että vihreä väri voidaan mieltää terveyden ja elämän tunnetiloihin. Vihreä on lisäksi luontoon liitettävä väri, ja luonto itsessäänkin elämänläheinen aihe.

Valitusta animaatiosta voitiin tässä tapauksessa erottaa kaksi avainasentoa kuvan 17 mukaan. Animaation alkupuolella hahmo ikään kuin kerää elinvoimaa itseensä, mikä voidaan esittää partikkelien vetäytymisenä hahmoon päin. Elinvoiman keräämisen korostamiseksi efektiin on hyvä sisällyttää sen merkityksen kannalta loogista symboliikkaa. Vetäytyvät partikkelit voivat siten olla muodoltaan plusmerkkisymboleja. Ne ilmaantuvat hahmon ympärille ja vetäytyvät hiljalleen sen yläkehon keskipistettä kohti. Kun hahmon elinvoima on palautunut, lehden muotoiset partikkelit lennähtävät ulospäin sen ympärillä tämän merkiksi ja hiipuvat vähitellen pois. Lopuksi spiraalimainen hehku liikkuu hahmon ympärillä ylöspäin kuvastaen loitsun poistumista.



Kuva 17. Parannusloitsuefektin konsepti

Primääri- ja sekundäärielementtien tunnistaminen loitsuefektissä voi olla haastavampaa, sillä kyky ei ole suoraan sidonnainen taisteluun ja intensiivisten iskujen tekemiseen. Loitsuefektin elementit voivat olla merkitykseltään samankaltaisia, eikä kyvyllä välttämättä ole yhtä selkeää kohokohtaa verrattuna hyökkäysliikkeisiin. Videopeleissä parantavien kykyjen tarkoitus on usein tukea taistelukohtauksia ja antaa pelihahmolle taistelua tehostavaa voimaa. Konseptikuvassa parannusloitsun primäärielementeiksi on kuitenkin pystytty osoittamaan ne efektin osat, jotka ilmaisevat sen merkitystä eniten. Näihin osiin kuuluu plusmerkkisymbolin sisältävät partikkelit sekä liikkeeltään intensiivisemmät lentävät partikkelit. Efektin muut osat lukeutuvat sekundäärisiin elementteihin, sillä niiden merkitys ei ole yhtä keskeinen.

Tekninen suunnittelu

Efektissä on hyödynnettävä 2D- ja 3D-tekniikoita. Suurin osa efektistä on mahdollista toteuttaa 2D-tekstuurien avulla, mutta spiraali vaatii 3D-mallia. Spiraalia varten täytyy erillisellä mallinnusohjelmalla mallintaa muotoisensa runko, jonka päälle tehdään ja asetetaan liikehtivä materiaali pelimoottorissa.

Ylöspäin säteilevä efekti voidaan yksinkertaisimmillaan toteuttaa pystysuuntaisesti venytettyjen 2D-partikkelien avulla. Efektin loppuvaiheessa säteilevien partikkelien nopeutta ja kokoa voidaan kasvattaa, jolloin ne saavat sumuisen

vaikutuksen. Maaginen ympyrä on jaettava useampaan eri tekstuuriin, sillä sen renkaiden halutaan mahdollisesti pyörivän eri suuntiin.

Pienet plusmerkkisymboliset partikkelit voidaan asettaa syntymään eripuolille pallomaista pintaa ja säätää niiden nopeutta siten, että ne liikkuvat pallon keskipistettä kohti. Ulospäin lentävät lehtipartikkelit voidaan toteuttaa lähes samanlaisella, mutta käänteisellä menetelmällä. Partikkelit asetetaan syntymään samasta pisteestä ja kasvattamalla niiden nopeutta pisteestä katsottuna, ne lentävät siitä poispäin satunnaisiin suuntiin.

4.2.2 Toteutus

Efektin kehittäminen aloitettiin yksinkertaisten osien, eli pluspartikkelien ja ulospäin lentävien partikkelien tekemisellä. Pluspartikkelit vaativat 2D-kuva-tekstuurista tehdyn materiaalin renderöintiä varten. Partikkelien haluttiin syntyvän tasaisesti vähitellen ajan kuluessa, joten niiden luomiseen käytettiin Spawn Rate -moduulia. Partikkelit asetettiin syntymään pallomaiseen muotoon Shape Location -moduulin avulla, kuten teknisessä suunnittelussa oli pohdittu. Partikkelit syntyivät aluksi pallon volyymin sisälle, mutta niiden haluttiin ilmaantuvan ainoastaan pallon pinnalle. Tätä varten partikkelien jakaumaa muotoon nähden oli säädettävä Sphere Surface Distribution -säätimen avulla. Arvo oli oletuksena 0, mutta säätämällä se arvoon 1, partikkelit syntyivät pallon pinnalle. Tämän jälkeen partikkeleille oli asetettava nopeus. Nopeuden suunnan haluttiin olevan kohti pallon keskipistettä, joten nopeuden arvo oli asetettava negatiiviseen lukuun. Partikkelien haluttiin lisäksi kutistuvan niiden lähentyessä pallon keskipistettä. Tämä toteutettiin partikkelien päivitysvaiheessa, johon lisättiin niiden kokoa säätelevä Scale Sprite Size -moduuli. Moduulin arvot muunnettiin käyräksi, jolloin arvojen muutos saatiin tapahtumaan käyrän mukaan partikkelien elinaikana. Partikkelien koon arvoksi asetettiin laskeva käyrä, jolloin niiden koko saatiin pienenemään tasaisesti elinajan loppua kohti.

Ulospäin lentävät partikkelit toteutettiin hyvin samankaltaisesti. Efektissä hyödynnettiin kahta erilaista partikkelia: lehden- ja ympyränmuotoisia, joille tehtiin omat emitterit. Lennähtävien partikkelien haluttiin syntyvän kerralla samanai-

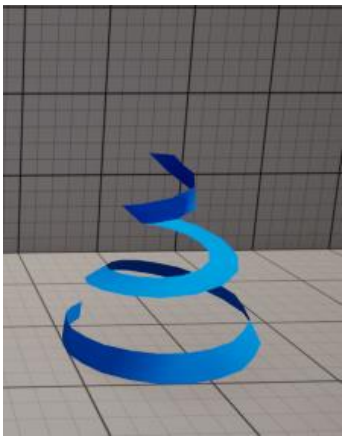
kaisesti, joten niille asetettiin Spawn Burst Instantaneous -moduuli. Partikkeleille asetettiin teknisen suunnitelman mukainen alustava nopeus, joka oli tyyppiä From Point. Lehdenmuotoisten partikkelien päivitysvaiheessa niiden rotaatiota säädettiin, jolloin ne saivat hitaan pyörimisliikkeen ja liike vaikutti realistisemmalta. Lehtipartikkelien kokoa pienennettiin niiden elinajan kuluessa, jolloin ne lopulta kutistuvat pois. Niiden liikkeen haluttiin olevan alussa räjähtävän voimakasta ja hidastuvan loppua kohti, joten päivitysvaiheeseen lisättiin Scale Velocity -moduuli. Nopeus asetettiin kasvamaan käyrän mukaan nopeasti ja laskemaan epätasaisesti hidastuen. Lisäksi ympyränmuotoisten partikkelien kokoa skaalattiin päivitysvaiheessa aaltoilevan käyrän mukaan, millä ne saatiin välkkymään.

Seuraavaksi toteutettiin maaginen ympyrä. Ympyrää varten piirrettiin aluksi kaksi kuvatekstuuria piirto-ohjelmalla, sillä ympyrän sisä- ja ulkokehän haluttiin liikkuvan eri suuntiin. Ympyrän keskustaa varten piirrettiin kuitenkin vielä erillinen plusmerkin kuva, jonka ei haluttu liikkuvan muun ympyrän mukana. Jokaisesta tekstuurista tehtiin oma materiaalinsa ja jokaiselle tehtiin oma emitteri Niagara-järjestelmään. Koska maagiselle ympyrälle tarvittiin vain yhdet partikkelit, lisättiin emittereihin Spawn Burst Instantaneous -moduuli, jossa syntyvien partikkelien määrä asetettiin yhteen. Ympyräpartikkeleille oli myös lisättävä Sprite Facing and Alignment -moduuli, jotta niiden suuntautuminen voitiin asettaa maata vasten. Sisä- ja ulkoympyrän pyörimisliikkeen säätämistä varten niille lisättiin päivitysvaiheeseen Sprite Rotation Rate -moduuli. Ympyröiden pyörimisliikkeille asetettiin vastakkaiset suunnat positiivisin ja negatiivisin arvoin, jolloin ne saatiin pyörimään eri suuntiin. Maagisen ympyrän plusmerkkipartikkelin päivitysvaiheessa sen koko asetettiin mukautumaan aaltomaisen käyrän mukaan, joka sai sen sykähtelemään pulssimaisesti. Sen väriä skaalattiin myös vaihtelemaan sinivihreän ja vihreän välillä pulssimaisen liikehdinnän korostamiseksi.

Seuraavaksi luotiin uusi emitteri ylöspäin säteileville partikkeleille. Partikkelien haluttiin asettuvan ympyrän muotoon maagisen ympyrän mukaan, joten emitteriin lisättiin Shape Location -moduuli, jonka muodoksi valittiin kehä. Partikkelien jakaumaa säädettiin moduulissa taas sopimaan paremmin kehän reunoille, kuten aiemmin tehtiin pallomaisen muodon kanssa plusmerkkipartikke-

lien kohdalla. Partikkelien alustuksessa niiden kokoa venytettiin pystysuunnassa, mikä sai ne näyttämään pitkiltä, ylöspäin osoittavilta säteiltä. Alustuksessa partikkelien sijaintia nostettiin hieman ylöspäin moduulin Transform-osiossa, jotta ne eivät ilmaantuisi huomattavasti maagisen ympyrän alapuolelle.

Lopuksi efektiin toteutettiin hehkuvat spiraalimaiset osat. Spiraaleja varten niille tehtiin kuvan 18 mukainen 3D-malli. Mallin kokoa voitiin muokata emitterissä, ja efektissä hyödynnettiinkin kahta erikokoista spiraalia.



Kuva 18. Spiraalin 3D-malli

Pelimoottorissa spiraaleille tehtiin uusi materiaali, jossa tekstuurien liike toteutettiin Panner-solmulla. Itse Niagara-systeemi oli loppujen lopuksi verrattain yksinkertainen. Spiraalien emittereissä partikkeleita luotiin ainoastaan yksi ja renderöitäväksi partikkeliksi valittiin spiraalin 3D-malli. Partikkelin päivitysvaiheessa partikkelin väriä ja kokoa skaalattiin. Värin skaalaamisella voitiin säätää partikkelin läpinäkyvyyttä sen elinajan alku- ja loppupäässä, jolloin efekti saatiin näyttämään sulavammalta sen alkamis- ja hiipumisvaiheessa. Puolestaan kasvattamalla partikkelin kokoa pystysuunnassa sen elinajan kuluessa, saatiin korostettua spiraalin liikettä ylöspäin. Koko efekti on esitetty kuvassa 19.

Kuva 19. Parannusloitsun efekti

Efektin toteutukseen tarvittiin kokonaisuudessaan neljä Niagara-järjestelmää. Järjestelmien emittereiden määrä pysyi yhden ja viiden välillä, joten efektit saatiin pidettyä suhteellisen yksinkertaisina. Efekti mukailee myös suunnitelmaa tarkasti.

4.3 Efekti 3: maaginen hyökkäys

Myös maaginen hyökkäys perustuu taianomaiseen kuvitelmaan, eikä siihen löydy suoraan realistisia vastineita todellisesta maailmasta. Hyökkäysliikkeen efektissä voidaan siitä huolimatta soveltaa esimerkiksi räjähdykselle ominaista partikkelien liikehdintää. Maaginen hyökkäys voi ilmentyä lukemattomilla erilaisilla efekteillä. Tässä tapauksessa maagisen hyökkäyksen efektissä yhdistyy heitettävä projektiili sekä suuren vaikutusalueen jälkeensä jättävä räjähdys. Vaikutusalueen jättäviä efektejä kutsutaan usein lyhenteellä AOE, eli *Area of Effect*. AOE-Efektin suunnittelua varten havainnoitiin taas samantyyppisten efektien ominaisuuksia Artstation-palveluun julkaistujen teosten avulla, joita on koottu kuvaan 20.

Kuva 20. Havainnoitujen AOE- ja projektiiliefektien esimerkkikooste

Koska kehitettävässä efektissä yhdistyy sekä projektiili että AOE-räjähdys, oli havainnointi kohdistettava molemman tyyppisiin efekteihin. Projektiilien efektit usein pitävät sisällään useamman osan, kuin pelkän lentävän ammuksen. Projektiilille on ominaista, että se jättää jälkeensä liikettä seuraavan juovan, sekä pienempiä partikkeleita.

Maagisille AOE-räjähdyksille osoittautui tyypilliseksi pallomainen muoto, ulospäin lentävät pienet partikkelit, sekä maan vaikutusalueen korostus. Väreistä ei havainnoinnin pohjalta voitu tehdä mitään johtopäätöstä. Maagisia efektejä voi olla hyvin monenlaisia eri merkityksineen, joten standardinomaisia värejä on hankala osoittaa niille. Lisäksi räjähdyksissä on usein hyödynnetty vääristävää vaikutusta (kuva 21).

Kuva 21. Ympäristön vääristymä räjähdyksen ympärillä

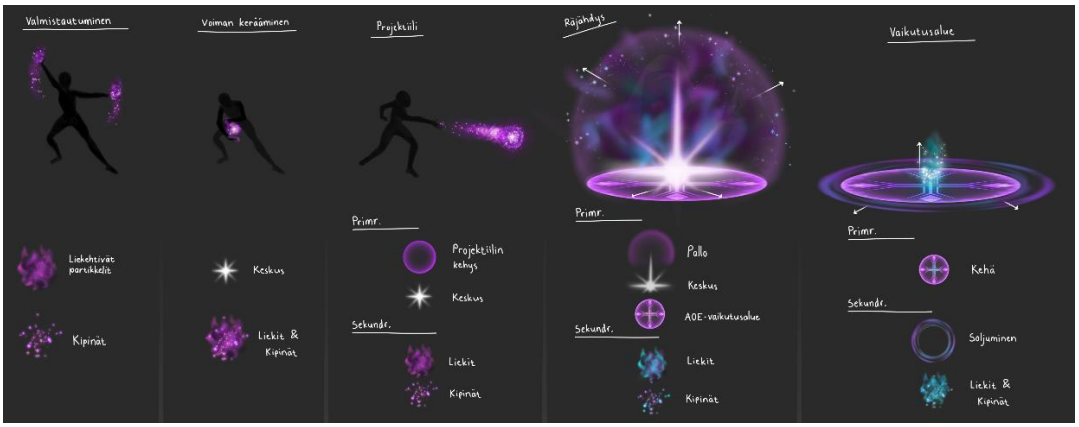
Vääristymät korostavat räjähdyksen voimakkuutta kuvastaen ikään kuin siitä aiheutuvaa paineaaltoa. Vääristymä vaikuttaa efektin lähiympäristöön esimerkiksi venyttämällä tai laajentamalla sitä.

4.3.1 Suunnittelu

Efektiä varten valittiin animaatio, jossa hahmo pyörittää käsiään maagisen voiman keräämiseksi ja lopulta heittää voiman eteenpäin.

Vaikka havainnoinnin perusteella ei voitu selvittää tunnusomaisia värejä maagiselle efektille, voitiin primääriväriksi valita violetti, sillä värikaavion mukaan se edustaa mystisyyden ja taikuuden mielleyhtymiä. Efektiä varten oli myös sopivaa valita sekundäärinen aksenttiväri, joka esiintyy efektissä pienemmissä määrin. Sekundääriväriksi valittiin syaanin sininen, sillä se tuo sopivaa kontrastia pääväriin.

Animaatiosta voitiin eritellä kolme vaihetta. Ensimmäisessä vaiheessa hahmo tekee käsillään ennakoivan liikkeen tuoden kädet eteensä lähelle toisiaan. Tämän jälkeen hahmo pyörittää käsiään lähekkäin kerätäkseen maagista voimaa. Lopulta hahmo ojentaa toisen kätensä eteen voiman vapauttamiseksi. Vaiheet on esitetty konseptikuvassa 22.



Kuva 22. Maagisen AOE-hyökkäyksen konsepti

Ensimmäisessä vaiheessa hahmon käsien kohdalle syntyy liekehtivä hehku, joka merkitsee valmistautumista voiman keräämiseen. Maagisen voiman kerääminen ilmenee samantyyppisenä hehkuna käsien keskiössä. Lopuksi

hahmo heittää maagisen projektiilin eteenpäin, joka maahan osuessaan aiheuttaa sijaintiinsa AOE-räjähdyksen. Räjähdyksessä yhdistyy havainnoinnin avulla selvitettyjä ominaisia elementtejä. Vaikutusalue hiipuu lopulta pois näkymättömiin ja sen vaikutukset lakkaavat.

Kokonaisuudessaan efekti koostuu useista vaiheista. Merkityksellisimmät efektin vaiheet ovat projektiili, räjähdys ja sen jälkivaihe. Niiden aikana olisi tarkoitus aiheuttaa vahinkoa pelissä, joten pelattavuuden kannalta ne kommunikoivat eniten pelaajan kanssa. Projektiilin intensiivinen liike sekä maata lähestyvä liikerata kertovat pelaajalle, että siitä aiheutuu jokin suurempi vaikutus. Räjähdys on intensiivisin vaihe, joka aiheuttaisi suurinta vahinkoa sen kohdalla oleviin vihollisiin. Räjähdyksen jälkeen vaikutusalue elää hetken aiheuttaen lisävahinkoa tai muita vaikutuksia sen sisällä oleviin vihollisiin.

Tekninen suunnittelu

Kuten konseptikuvasta nähdään, useassa efektin vaiheessa hyödynnetään liekehtiviä partikkeleita. Tässä tapauksessa liekeille voidaan tehdä oma erillinen emitteri, joka sitten sisällytetään muihin Niagara-järjestelmiin. Liekkipartikkeleita varten voidaan käyttää useampaa erilaista tekstuuria, jotta liekkeihin saadaan aikaan parempaa satunnaisuutta. Tätä varten emitterin renderöintivaiheeseen voidaan lisätä eri tekstuureja vastaava määrä renderöijiä, joihin asetetaan halutut tekstuureista tehdyt liekkimateriaalit. Tämän jälkeen partikkeleita voidaan satunnaistaa niiden alustusvaiheessa VisibilityTag-parametrin avulla, jolloin seuraavaksi renderöitävä partikkeli valikoituu aina satunnaisesti asetettujen vaihtoehtojen väliltä.

Ensimmäisessä vaiheessa ja projektiilissa liekkipartikkelien halutaan jättävän juovan jälkeensä, kun taas räjähdyksessä ja jälkiefektissä liekkien liike on staattisempaa. Lisäksi liekkien syntymistapa, koko ja väri vaihtelee efektin osien välillä, mutta erillisen emitterin alkuperäisiä moduuleja ja parametrejä voi kuitenkin muokata Niagara-järjestelmässä tarpeiden mukaan.

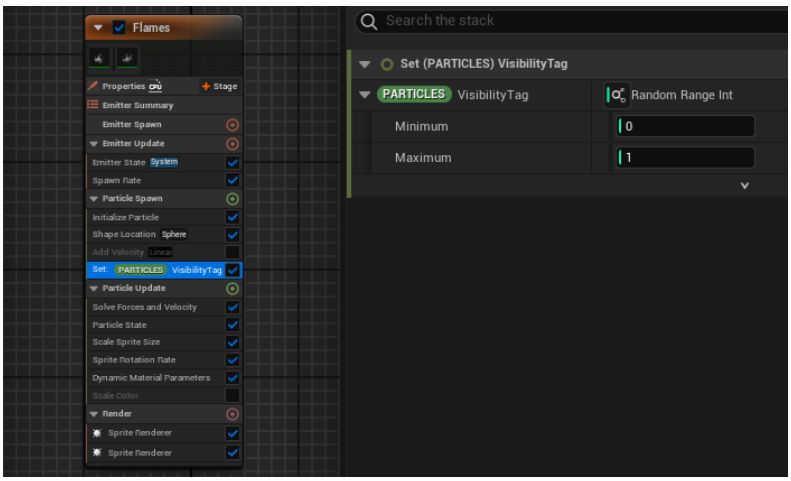
Kuten aiemmissakin efekteissä, myös tämän efektin kohdalla on osittain turvauduttava 3D-partikkeleihin. Muun muassa räjähdyksen pallomainen kupu voidaan toteuttaa 3D-pallon avulla, jolloin siihen saadaan parempaa syvyyttä

aikaan. Pallon materiaaliin voidaan tehdä taustaa vääristävä ominaisuus materiaalin päätesolmun Refraction-syötteen avulla, joka lisää räjähdyksen realistista vaikutelmaa.

Jälkivaiheen aikana maagisen vaikutusalueen ympärille ilmaantuu soljuva efekti. Tätä varten voidaan tehdä materiaali, joka liikuttaa tekstuureja ulospäin tason keskipisteestä. Emitterin kannalta tässä kohtaa on hyödynnettävä Mesh-renderöijää, jotta materiaalin liikkumapinnaksi saadaan asetettua litteä taso.

4.3.2 Toteutus

Efektin työstäminen aloitettiin liekkipartikkelien toteuttamisella, sillä niiden oli tarkoitus tulla käyttöön useammassa efektin vaiheessa. Teknisen suunnitelman mukaan liekeille tehtiin yksi emitteri, jonka renderöintivaiheeseen sisällytettiin kaksi renderöijää erilaisille liekeille. Renderöijien Visibility-arvoiksi asetettiin eri luvut, jotka vaikuttavat partikkelien satunnaisuuden arpomiseen. Seuraavaksi partikkeleille oli Niagara-järjestelmän parametrivälilehdellä tehtävä uusi parametri nimeltä VisibiltyTag. Jotta liekkipartikkelit lopulta saatiin syntymään satunnaisesti näiden kahden vaihtoehdon väliltä, lisättiin partikkelien alustusvaiheeseen uusi moduuli nimeltä *Set new or existing parameter directly*, johon VisibilityTag-parametri raahattiin parametrivälilehdeltä. Sitten parametri muutettiin saamaan satunnaisen arvon aiemmin määriteltyjen renderöijien Visibility-arvojen väliltä (kuva 23).



Kuva 23. Renderöitävien partikkelien satunnaistaminen yhden emitterin sisällä

Liekkien materiaalissa käytettiin Voronoi Noise -tyyppistä tekstuuria yhdessä varsinaisten liekkitekstuurien kanssa luomaan realistisuutta. Voronoi-tekstuurin arvoihin yhdistettiin materiaalissa Dynamic Parameter -solmu. Tämän avulla kyseisiä arvoja voitiin muokata emitteristä käsin Dynamic Material Parameters -moduulin avulla partikkelien päivitysvaiheessa. Näin partikkelien ilmettä saatiin helposti mukautettua sopimaan erilaisiin tarkoituksiin.

Valmistautumisvaihe

Seuraavaksi kehitettiin ensimmäisen vaiheen efekti, jonka oli tarkoitus kuvastaa valmistautumista maagisen voiman keräämiseen. Efektille tehtiin oma Niagara-järjestelmä, johon sisällytettiin aiemmin tehty liekkiemitteri. Tämän lisäksi järjestelmään tehtiin emitterit pienille kipinöiville partikkeleille, sekä sumuisemmille hehkupartikkeleille. Kaikki emitterit olivat World Space -tilassa, sillä niiden oli tarkoitus jättää juova jälkeensä hahmon käsien liikkeen mukana.

Voiman kerääminen

Toisessa vaiheessa hahmo kerää voimaa pyörittämällä käsiä lähekkäin. Tätä vaihetta varten ei oikeastaan tarvinnut tehdä paljon mitään uutta. Ensimmäisen vaiheen Niagara-järjestelmä monistettiin, mutta tällä kertaa sen emitterit asetettiin Local Space -tilaan. Tällöin partikkelit saatiin pysymään staattisemmin hahmon käsien läheisyydessä. Järjestelmään lisättiin vielä yksinkertainen tähtimäinen partikkeli, joka toimi voiman keräämisen keskipisteenä.

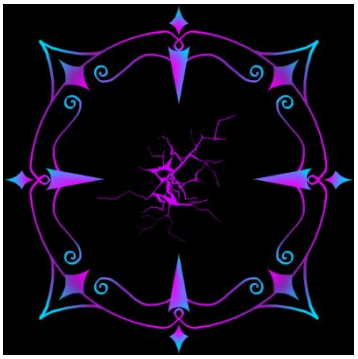
Projektiili

Seuraavaa vaihetta varten kehitettiin projektiilin efekti. Projektiilin pää koostui tähtimäisestä keskustasta sekä pallosta, jonka pinnalla on pieniä kimaltavia partikkeleita. Projektiilin päähän hyödynnettiin 3D-palloa. Pallolle tehtiin Fresnel-materiaali, joka sai aikaan palloon läpinäkyvän keskustan. Projektiilin juovaa varten hyödynnettiin tässäkin efektissä aiemmin tehtyä liekkiemitteriä. Liekkipartikkelien kokoa, määrää sekä materiaalin dynaamisia arvoja muutettiin sopimaan projektiilin kokoon ja liikkeeseen. Kimaltaville partikkeleille tehtiin kaksi emitteriä, sillä niiden haluttiin pysyvän sekä pallon pinnalla että jää-

vän projektiilin liikkeen jälkeen. Täten pinnalla pysyvät partikkelit asetettiin Local Space -tilaan, ja jälkeen jäävät partikkelit puolestaan World Space -tilaan. Projektiilin keskustaa varten tehtiin taas hyvin yksinkertainen yhden partikkelin emitteri, johon asetettiin tähtimäinen materiaali.

AOE-räjähdys

Lopuksi oli jäljellä enää AOE-räjähdyksen efekti. Räjähdys jälkiefekteineen sisälsi monta osaa. Vaikutusalueelle piirrettiin kuvatekstuuri erillisellä piirto-ohjelmalla. Vaikutusalueen tyyliä muutettiin kuitenkin hieman erilaiseksi, mitä konseptikuvassa oli alustavasti ajateltu (kuva 24).



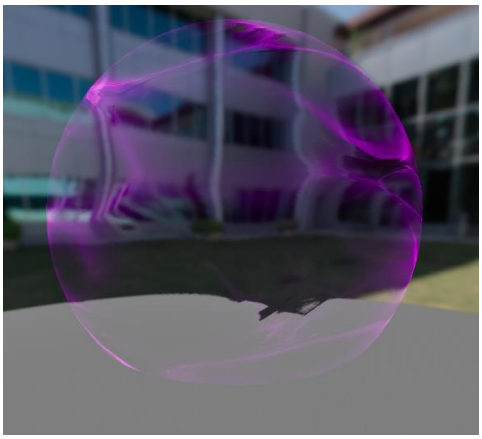
Kuva 24. Vaikutusalueen tyyli

Kuvan kehys sekä keskellä näkyvä murtuma toteutettiin erillisinä tekstuureina ja emittereinä AOE-järjestelmään. Vaikutusalue syntyy räjähdyksen seurauksena, ja murtuma toimii visuaalisesti ikään kuin jälkivaiheen liekkien lähteenä. Niagara-järjestelmässä molemmille tehtiin yhden partikkelin emitterit. Partikkelien hälveneminen toteutettiin läpinäkyvyyden kasvattamisella eliniän loppua kohti.

Räjähdystä varten AOE-järjestelmään sisällytettiin liekkiemitteri, joka muutettiin näyttämään räjähtävämmältä. Alkuperäinen Spawn Rate -moduuli vaihdettiin Spawn Burst Instantaneous -moduuliin, jotta useampi liekkipartikkeli saatiin pelmahtamaan samanaikaisesti. Partikkelien kokoa oli kasvatettava huomattavasti ja niiden väri satunnaistettiin violetin ja syaanin välille. Lisäksi emitteriin lisättiin nopeuden moduuli, jotta nopeutta saatiin skaalattua partikkelien eliniän loppuvaiheella. Tämän ansiosta liekkipartikkelit saatiin liikkumaan ylöspäin. Räjähdyksen keskustaan kuului myös tähtimäinen partikkeli sekä pitkä

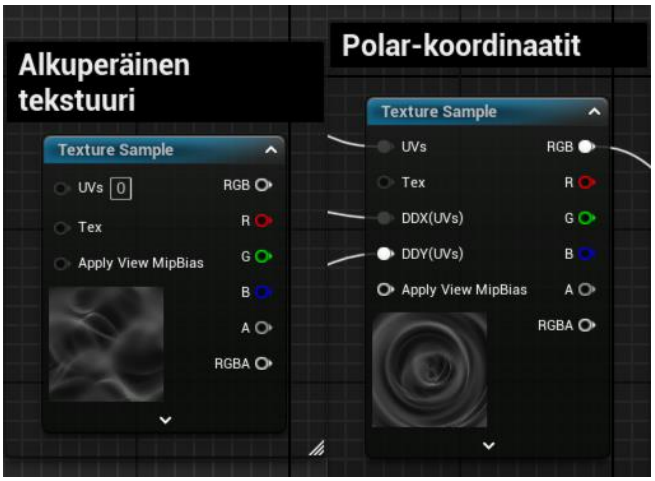
pystysuuntainen partikkeli, joiden avulla räjähdyksestä saatiin tyylitellympi ja sarjakuvamaisempi.

Räjähdyksessä käytettiin 3D-palloa, joka laajenee nopeasti alkuvaiheessa. Pallolle tehtiin erityinen materiaali, jossa hyödynnettiin liikkuvia tekstuureja sekä Refraction-syötettä. Läpinäkyvä pallo saatiin näin aiheuttamaan havainnoinnin perusteella räjähdyksille ominaista ympäristön vääristymää (kuva 25).



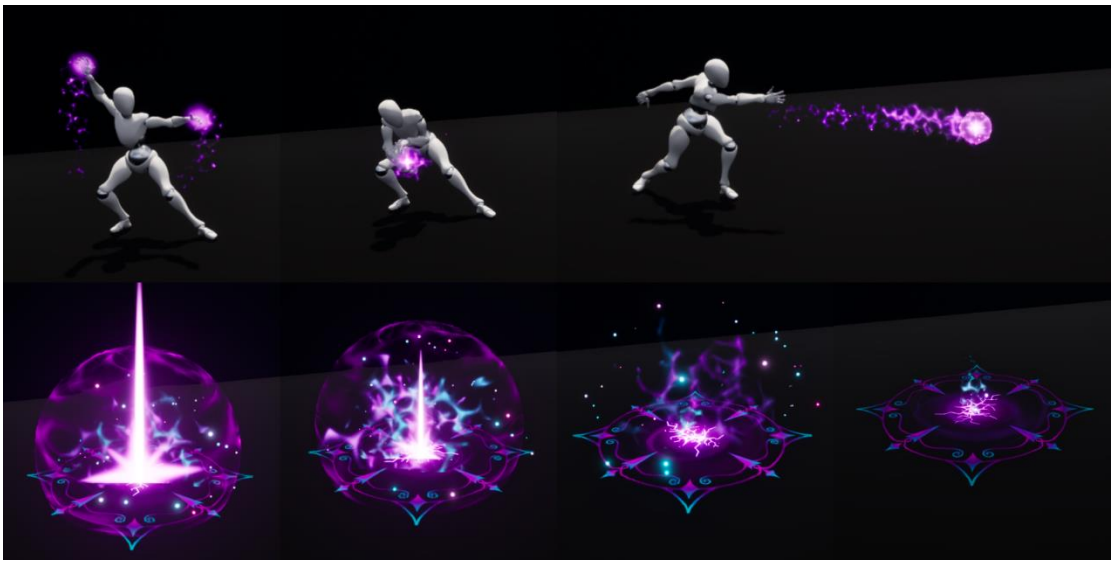
Kuva 25. Materiaalin vääristävä vaikutus

Räjähdyksen jälkiefektiä varten järjestelmään lisättiin vielä yksi liekkiemitteri, joka liekehti ylöspäin kuten nuotio. Jälkiefektiin kuului myös maan soljuva efekti, joka toimi lähes kokonaan materiaalin voimin. Materiaaliin sisällytettiin tekstuuri, jolle lisättiin liikettä Panner-solmulla siten, että se liikkuu tason keskipisteestä ulospäin. Jotta liike saatiin syntymään tason keskipisteestä käsin, oli tekstuurin UV-koordinaatit muutettava ensin Polar-koordinaateiksi. Koordinaattimuutoksen vaikutusta on havainnollistettu kuvassa 26.



Kuva 26. Tekstuurikoordinaattien muutoksen havainnollistus

Lopputuloksena saatiin kokonaisuudessaan toteutettua kuvan 27 mukainen efekti. Kuvasta nähdään, että efektin vaiheet myötäilevät suunnitelmaa hyvin tarkasti. Ainoastaan vaikutusalueen tyyli on hieman erilainen kehityksen aikana tapahtuneiden muutosten takia.



Kuva 27. Maaginen AOE-hyökkäysefekti

Efekti koostuu kokonaisuudessaan neljästä Niagara-järjestelmästä: valmistautumisen efektistä, voiman keräämisestä, projektiilista sekä AOE-räjähdyksestä jälkivaiheineen. Näistä eniten emittereitä osakseen vaati AOE-efekti, johon tarvittiin yhteensä 10 emitteriä. Yksinkertaisimmillaan järjestelmä puolestaan toimi kolmen emitterin voimin.

5 EFEKTIEN LIITTÄMINEN HAHMOON

Visuaalisia efektejä voidaan liittää suoraan hahmon animaatioon Unreal Enginessä animaation sekvenssissä, joka sisältää animaation datan. Sekvenssi pitää sisällään animaation kaikki avainkohdat, jotka määrittävät hahmon asennon, sijainnin ja rotaation tiettyinä ajan hetkinä. Animaatiosekvenssiin on mahdollista lisätä Notify-ilmoituksia, joiden avulla animaation aikana voidaan suorittaa erilaisia tapahtumia, kuten ääniefektejä, Blueprint-koodia tai visuaalisia efektejä (Animation Notifies s.a.).

Kun Notify-ilmoitus lisättiin sekvenssiin, oli valittava animaation aikana suoritettavan tapahtuman tyyppi. Koska tarkoituksena oli toistaa visuaalinen efekti,

valittiin ilmoituksen tyypiksi *Play Niagara Particle Effect*. Kyseistä tyyppiä oli hyvä käyttää lyhyiden, kerran toistettavien efektien kohdalla, jotka eivät vaadi kummoisempaa toiminnallisuutta.

Notify-ilmoitus tuli näkyviin animaatiosekvenssiin. Tarkastelemalla ilmoituksen ominaisuuksia löydettiin erilaisia siihen vaikuttavia asetuksia. Ominaisuuksissa ilmoitukseen sai valittua halutun Niagara-järjestelmän, joka toistuisi animaation aikana. Efektien haluttiin tietenkin toistuvan tietyillä animaation hetkillä, joten Notify-ilmoituksia oli siirrettävä raahaamalla ne oikeille animaation kohdille. Kun animaatio käynnistettiin, efektit alkoivat toistumaan niissä kohdissa, joihin ilmoitukset oli asetettu.

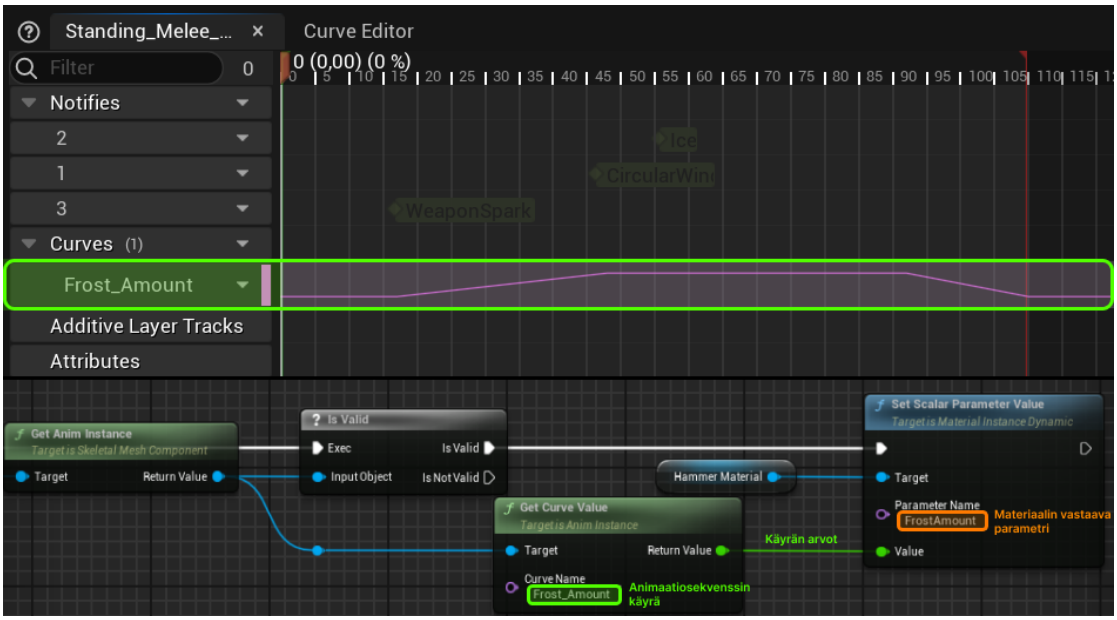
Notify-ilmoituksen ominaisuuksissa valittiin tietty hahmon luurangon osa, johon efekti haluttiin kiinnittää. Näin Niagara-järjestelmä saatiin syntymään valitun ruumiinosan kohdalle. Jotta Niagara-järjestelmä myös seuraisi hahmoa liikkeen mukana, oli Notify-ilmoituksen ominaisuuksista otettava vielä käyttöön Attached-ominaisuus, jolla järjestelmä kiinnittyy valittuun luurangon osaan. Tällöin Niagara-järjestelmä perii hahmon luurangon osan sijainnin, rotaation ja skaalan.

Muun muassa maaiskuefektin kohdalla osa Niagara-järjestelmistä oli kiinnitettävä hahmon luurangon osaan Attached-ominaisuudella, sillä animaatio liikkui eteenpäin, ja efektien oli seurattava hahmon sijaintia. Hahmon luurangon rakenteeseen oli mahdollista lisätä uusia Socket-osia, joihin Niagara-järjestelmän sai myös kiinnitettyä samaan tapaan kuin luihin. Socket-osat toimivat hahmon luiden irrallisina jatkeina ja niiden paikkaa sekä rotaatiota voitiin muokata. Niiden avulla efekti siis saadaan toistumaan mukautetussa luurangon sijainnissa.

Mixamosta ladatun hahmon luurangon rakenne ei sisältänyt staattista juuriluuta, jonka rotaatio pysyisi muuttumattomana animaation aikana. Socket-osat täytyi tällöin lisätä muiden rakenteessa olevien luiden alle, jolloin ne liikkuivat ja kääntyilivät animaation aikana muun luurangon mukana. Myös niihin kiinnitetyt Niagara-järjestelmät tekivät samoin. Tämän takia maaiskuefektin kohdalla törmättiin ongelmaan, jossa Socketiin kiinnitetyt jääpiikit kallistuivat täysin väärään suuntaan tai ilmestyivät vääriin kohtiin. Staattisen juuriluun avulla

Socketit olisi voitu lisätä suoraan sen alle, jolloin niiden rotaatio ei olisi välttämättä muuttunut animaation aikana, ja siten efektikin olisi suuntautunut paremmin. Ongelman ratkaisemiseksi hahmolle olisi voitu lisätä juuriluu itse hyödyntämällä 3D-mallinnusohjelmaa tai Unreal Enginen lisäosaa luurangon muokkaamiseen. Tämä olisi kuitenkin vaatinut edistyneempää työtä 3D-mallinnuksessa käytettyjen tekniikoiden parissa. Hahmon 3D-mallinnuksen keinot oli pitkälti rajattu opinnäytetyön ulkopuolelle, eikä ongelman ratkaisemista siten lähestytty siltä kannalta. Toinen tapa ratkaista ongelma oli säätää Socketin rotaatiota oikeassa animaation vaiheessa, jotta efekti saatiin suuntautumaan oikein. Tämä ei toki ollut yhtä tehokas ratkaisu, mutta toimiva vaihtoehto siitä huolimatta.

Maahan kohdistuvan iskuliikkeen aikana tapahtuvaan aseen jäätymiseen hyödynnettiin animaatiosekvenssin käyräominaisuutta. Koska ase ei ollut osa hahmon rakennetta, vaan erillinen objekti, ei käyräominaisuutta voitu kuitenkaan hyödyntää materiaalin muutokseen suoraan. Käyrän avulla voitiin silti määrittää, kuinka aseen tulisi jäätyä animaation aikana. Tämän jälkeen määriteltyä käyrää voitiin hyödyntää Blueprint-koodissa. Animaatiosekvenssissä määritelty käyrä ja sen hyödyntäminen koodissa on havainnollistettu kuvassa 28.

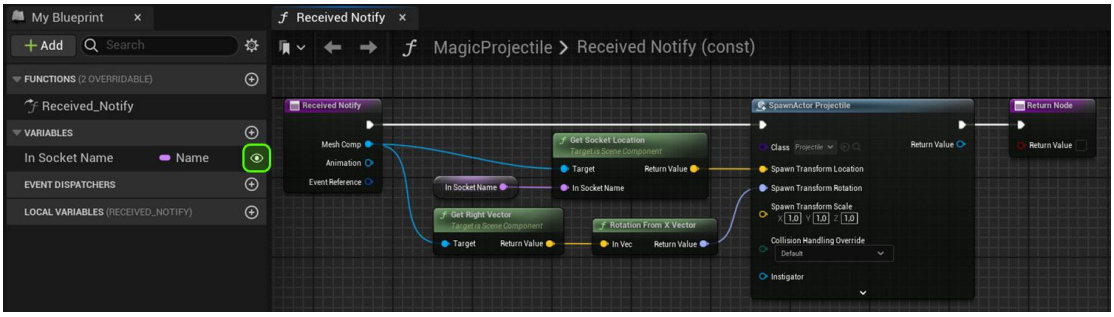


Kuva 28. Animaatiosekvenssin käyrän hyödyntäminen Blueprint-koodissa

Koodissa luetaan animaatiosekvenssiin luodun käyrän arvot Get Curve Value -funktiolla. Tämän jälkeen käyrän arvot asetetaan Set Scalar Parameter Value -funktiolla aseen materiaalin vastaavalle parametrille. Näin materiaali vaihtuu jäiseksi animaation aikana käyrän mukaan.

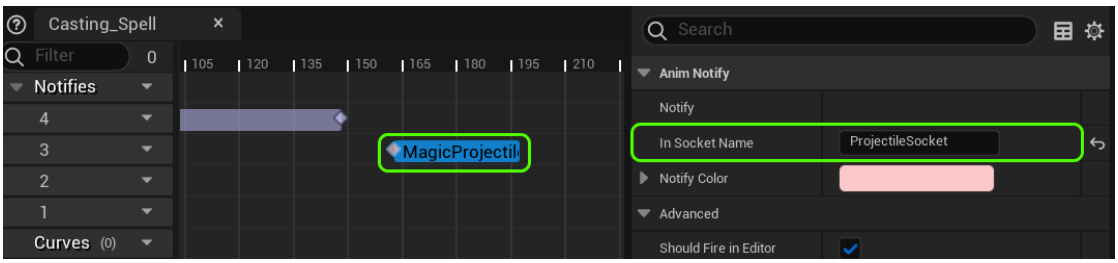
Sekvenssiin oli mahdollista lisätä myös Notify State-ilmoituksia, eli tilailmoituksia, jolloin tyypiksi voitiin valita ajastettu *Timed Niagara Effect*. Ajastetun tilailmoituksen avulla efektin kesto voidaan määrittää paremmin animaatiosekvenssissä, mikäli efektin käyttäytyminen on asetettu Niagara-järjestelmässä äärettömäksi. (Animation Notifies s.a.) Ajastettua efektin Notify-ilmoitusta käytettiin maagisen hyökkäyksen valmistautumisvaiheessa ja voiman keräämisen vaiheessa. Efektin kesto voitiin siten helpommin määrittää sopimaan animaatioon.

Maagisen hyökkäysefektin liittäminen hahmon animaatioon tuotti ongelmia projektiilin ja AOE-räjähdyksen osalta. Koska efektin oli tarkoitus sisältää toiminnallisuutta, kuten törmäyksen havaitsemisen sekä projektiilin liikkumisen, ei sitä voitu liittää animaatiosekvenssiin aiemmin mainituin Notify-ilmoituksin. Projektiilin efektille oli ensin tehtävä Actor-tyyppinen Blueprint-luokka, jolloin siitä saatiin helpommin hallittava objekti. Actorille voitiin sitten lisätä tarvittavia komponentteja, kuten Niagara-komponentti projektiilin efektiä varten, törmäyskomponentti sekä Projectile Movement -komponentti, jonka avulla projektiilille voitiin helposti antaa liikeominaisuus. Projektiilille voitiin tämän jälkeen tehdä Blueprint-koodia, jossa tarkasteltiin sen törmäystä maan kanssa. Mikäli törmäys tapahtuu, törmäyksen sijainnissa kutsutaan AOE-räjähdyksen efektiä suoraan koodista käsin. Tämän jälkeen täytyi tehdä vielä toinen Blueprint-koodi, jotta projektiili saatiin ilmestymään hahmon käden kohdalle ja lentämään siitä eteenpäin. Blueprintiä luotaessa sille on valittava jokin yläluokka. Tässä tapauksessa yläluokaksi haettiin AnimNotify-luokka, jotta koodin kutsuminen animaatiosekvenssissä olisi myöhemmin mahdollista. Override-funktioksi valittiin Received Notify, jotta koodi suoritetaan, kun ilmoitusta kutsutaan animaatiosekvenssissä. Tämän jälkeen toteutettiin tarvittava koodi (kuva 29).



Kuva 29. Projektiilin koodi

Koodiin lisättiin Spawn Actor from Class -funktio, johon Actoriksi valittiin aiemmin tehty projektiilin Blueprint-luokka. Projektiilin haluttiin syntyvän hahmon oikeaan käteen lisätyn Socketin kohdalta, joten koodiin lisättiin Get Socket Location -funktio, johon tarvittiin muuttujaksi Socketin nimi sekä referenssi hahmoon. Hahmon referenssi saatiin suoraan Received Notify -solmusta, ja Socketin nimi muutettiin julkiseksi muuttujaksi, jotta sitä voitiin käsitellä koodin ulkopuolella. Projektiilin liikkeen tuli suuntautua aina hahmosta eteenpäin, joten lopuksi projektiilille täytyi vielä selvittää oikea rotaatio. Tämän jälkeen animaatiosekvenssiin voitiin lisätä kyseinen MagicProjectile-niminen Notify-ilmoitus kuvan 30 mukaan, kun sen yläluokaksi oli aiemmin valittu AnimNotify.



Kuva 30. Projektiilin koodia kutsutaan ilmoituksen avulla animaatiosekvenssissä.

Ilmoituksen ominaisuuksissa näkyi julkiseksi muutetun Socketin nimen muuttuja, johon voitiin kirjoittaa halutun Socketin nimi. Nimimuuttujan perusteella efekti saatiin ilmestymään oikean Socketin kohdalle animaation aikana koodista käsin.

6 OPTIMOINTI

Kun visuaaliset efektit alkavat monimutkaistua, on sillä heikentävä vaikutus pelin suorituskykyyn. Siksi efektien optimointi on tärkeä osa kehitysprosessia. Optimoinnin suunnittelu olisi hyvä aloittaa jo efektin suunnittelun yhteydessä,

jolloin voidaan alustavasti pohtia teknisiä ratkaisuja optimaalisemman lopputuloksen saavuttamiseksi. Tässä opinnäytetyössä optimoinnin keinoja sovellettiin kuitenkin toteutuksen jälkeen, jotta optimoinnin vaikutuksia voitiin havaita paremmin.

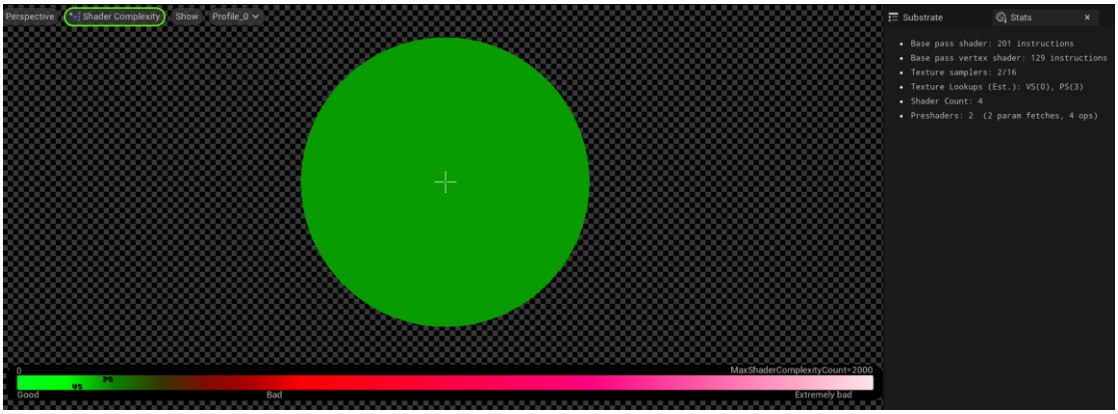
6.1 Optimoinnin kohteet

Niagara-järjestelmän emittereissä partikkelit on oletuksena asetettu simuloitumaan CPU:ssa, eli keskussuorittimessa. Tuhansien partikkelien simulointi CPU:n voimin heikentäisi suorituskykyä huomattavasti, sillä CPU:ssa tapahtuu paljon muutakin laskentaa. Simuloinnin prosessointi ja laskenta voidaan vaihtaa myös tapahtumaan GPU:ssa, eli grafiikkasuorittimessa, joka puolestaan mahdollistaa useidenkin tuhansien partikkelien simuloinnin ilman siitä aiheutuvia huomattavia suorituskyvyn ongelmia. (CPU and GPU Sprite… s.a.) Yleisesti ottaen GPU-simulointi on tehokkaampaa, mutta esimerkiksi mobiilialustoilla se voi muodostaa pullonkaulan, kun grafiikkamuistia on vähemmän käytettävissä. (Optimizing Niagara… 2024.) Pelihahmojen efektit tässä opinnäytetyössä ovat lyhytkestoisia, eivätkä yksittäiset emitterit hyödynnä tuhansia partikkeleita, joten simulointi CPU:ssa oli riittävä ratkaisu. On kuitenkin pidettävä mielessä, kuinka monta partikkeliefektiä pelissä simuloitaisiin samanaikaisesti. Pelin aikana simuloitavien partikkelien kokonaismäärä voi kasvaa suureksi, vaikka yksittäiset efektit olisivatkin yksinkertaisia.

Toinen huomioitava seikka efektien optimoinnissa on *Overdraw*, eli päällekkäin piirtyvät kerrokset. Tätä tapahtuu usein sellaisten partikkelien kohdalla, jotka hyödyntävät läpinäkyvyyttä. Läpinäkyvät partikkelit piirtyvät enemmän tai vähemmän toistensa päälle, mikä voi merkittävästi heikentää suorituskykyä. Päällekkäinpiirrosta aiheutuvat ongelmat riippuvat myös vaikuttavien pikselien määrästä. (VFX Optimization Guide s.a.) Mitä suuremman alueen päällekkäinpiirto siis kattaa ruudusta, sen heikommaksi tehokkuus laskee. Partikkelien läpinäkyvyys säädetään niiden materiaalissa. Materiaaleista laskennallisesti kalleimpia ovat Translucent- ja Additive-tyyppiset, sillä ne mahdollistavat partikkelien läpinäkyvyyden. Kyseisiä materiaalityyppejä tulisi välttää aina mahdollisuuksien mukaan ja niiden sijaan käyttää Opaque- tai Masked-tyyppisiä materiaaleja. Mikäli partikkelin läpinäkyvyyttä ei siis tarvitse skaalata efektissä, eikä

tekstuuri itsessään sisällä läpinäkyvyyttä, on parempi käyttää kustannustehokkaampia materiaalityyppejä.

Materiaalit Unreal Enginessä toimivat shadereina. Materiaaleissa voi tapahtua paljon laskentaa, kun tavoitteena on saada jotain yksityiskohtaista visuaalisuutta aikaan. Tämä voi tehdä materiaalista kuitenkin monimutkaisen ja siten kalliin. Materiaalin kompleksisuudella on vaikutusta suorituskykyyn ja sitä tulisi pyrkiä kontrolloimaan. Materiaalien kustannuksia, partikkelien lukumäärää sekä efektien etäisyyttä ruudulla olisi tärkeä analysoida optimoinnin kannalta. (VFX Optimization Guide s.a.) Unreal Enginessä materiaalien monimutkaisuutta kuvataan käskyjen määrällä eli Instruction Countilla. Korkea käskyjen määrä tarkoittaa, että materiaali on monimutkainen ja sitä tulisi pyrkiä yksinkertaistamaan. Jo tyhjät materiaalit sisältävät yli 200 käskyä riippuen materiaalityypistä. Kalleimman Translucent-materiaalin käskyjen määrä on lähempänä kolmeasataa. Materiaalin kompleksisuutta voidaan tarkastella sen esikatseluikkunassa Shader Complexity -tilan avulla (kuva 31).



Kuva 31. Materiaalin Shader Complexity -tarkastelutila

Materiaalin kompleksisuuden aste nähdään esikatseluikkunan alareunassa näkyvältä väriasteikolta. Valkoinen väri tarkoittaa korkeinta käskyjen määrää ja siten monimutkaista materiaalia. Puolestaan vihreä väri kertoo materiaalin olevan yksinkertainen ja tehokas. Käskyjen määrän lukema asettuu väriasteikolla nollan ja kahdentuhannen välille.

Monimutkaisten materiaalien kohdalla voidaan yrittää vähentää ohjeistusten määrää yksinkertaistamalla matemaattisia laskutoimituksia tai käyttämällä materiaalin tarjoamia valmiita funktioita. Lisäksi ehtohaarojen välttäminen sekä

materiaalin tarjoamien vektorioperaatioiden käyttäminen on suositeltavaa. (Shestakova 2024.) Optimaalisemman materiaalin ohjeistusten määrä voidaan työn osalta asettaa kahden- ja viidensadan välille, jolloin lukema asteikolla pysyttelee vihreän ja punertavan välimaastossa.

Efektejä voidaan myös optimoida eri etäisyyksien perusteella säätämällä yksityiskohtaisuuden tasoa eli Level of Detail (LOD) -tasoa. Kauempana näkyvien efektien yksityiskohtaisuus voidaan asettaa matalaksi, kun taas lähempänä olevien korkeammaksi. (VFX Optimization Guide s.a.) Opinnäytetyön aikana kehitettävät pelihahmon efektit pysyvät lähes samoilla etäisyyksillä hahmon lähettyvillä, minkä vuoksi LOD:n lisääminen ei ole tämän työn kannalta kovin tarpeellinen toimenpide. Varsinaisissa peleissä LOD:n lisääminen on olennaisempaa, kun efektien etäisyys vaihtelee.

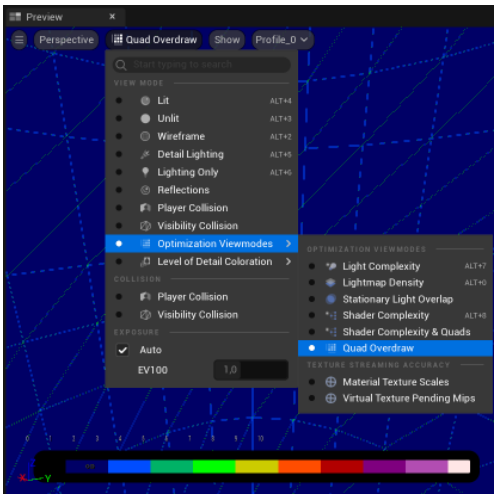
Unreal Engine 5.4 esitteli myös ensimmäistä kertaa uutena ominaisuutena tilattomat emitterit, jotka ovat valmiiksi optimoidumpia tavallisiin emittereihin verrattuna. Tilattomat emitterit tulevat olemaan useimmissa tapauksissa huomattavasti tehokkaampia, sillä niiden lukumäärällä järjestelmän sisällä on vähemmän vaikutusta tehokkuuteen. Järjestelmän sisällä voi samanaikaisesti toimia sekä perinteisiä että tilattomia emittereitä, mutta jälkimmäiset ovat tehokkaampia. Tehokkuuden ylläpitämiseksi tilattomat emitterit voivat hyödyntää ainoastaan tiettyjä moduuleja. (Niagara Lightweight Emitters… s.a.) Näihin moduuleihin kuitenkin lukeutuu hyvin monet tämänkin työn aikana käytetyistä moduuleista.

6.2 Efektien optimointi

Lähes jokaisen efektejä varten tehdyn materiaalin käskyjen määrä saatiin pysymään kahdensadan puolella. Materiaaleista kuitenkin vaativin oli jäämateriaali, jonka käskyjen määrä oli 346. Materiaalin kompleksisuus pysytteli väriasteikolla punaisen alapuolella, joten sen todettiin olevan tarpeeksi tehokas työn osalta.

Materiaalit olivat siis yksinkertaisia itsessään, eikä niissä ollut paljon varaa optimoinnille. Työn osalta ainoastaan läpinäkyvien materiaalien hyödyntäminen partikkeleissa aiheutti kompleksisuutta päällekkäinpiirron osalta.

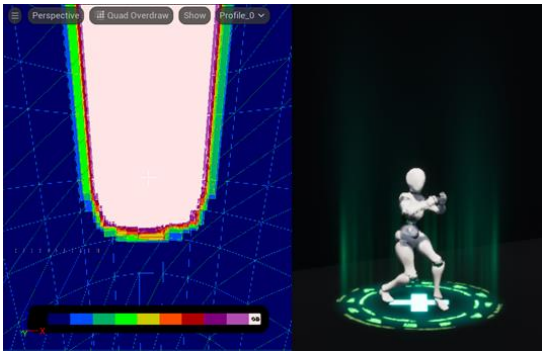
Päällekkäinpiirto oli eräs aiemmin mainituista huomioitavista optimoinnin seikoista. Niagara-työkalun esikatseluikkunassa laitettiin päälle Quad Overdraw -tarkastelutila, jonka avulla pystyttiin tarkastelemaan, kuinka efektin eri kuvakerrokset piirtyivät toistensa päälle (kuva 32).



Kuva 32. Päällekkäinpiirron tarkastelutila Niagara-työkalun esikatseluikkunassa

Myös tässä tarkastelutilassa väriasteikko kuvastaa päällekkäinpiirron määrää. Valkoinen väri tarkoittaa suurinta päällekkäinpiirron määrää ja sininen vastaavasti pienintä. Tarkastelutilan avulla havaittiin muutaman efektin kohdalla tapahtuvan voimakasta päällekkäinpiirtoa. Tällaisia olivat juuri läpinäkyviä materiaaleja hyödyntävät efektit, kuten liekit, sumupartikkelit sekä parannusloitsun säteily.

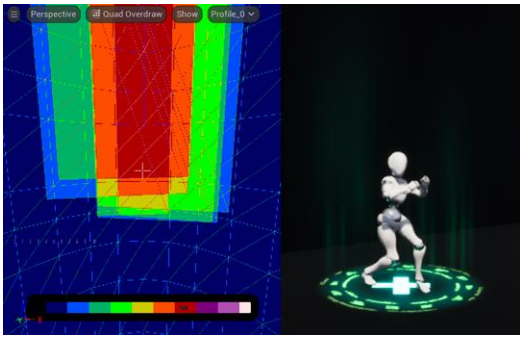
Parannusloitsun säteilevät partikkelit aiheuttivat hiipuessaan huomattavan määrän päällekkäinpiirtoa, sillä läpinäkyvien partikkelien koko oli asetettu kasvamaan niiden elinajan lopulla. Lukuisat suuret ja läpinäkyvät partikkelit siis piirtyivät toistensa päälle, ja lukema väriasteikolla nousi valkoiselle alueelle (kuva 33).



Kuva 33. Säteilyefektin päällekkäinpiirto ennen optimointia

Säteilevien partikkelien lukumäärä oli alun perin sata. Lukumäärää oli vähennettävä huomattavasti, jotta päällekkäinpiirto saatiin siedettävämpään lukemaan. Vähennyksen jälkeen partikkeleita oli 13. Partikkelien lukumäärän vähentäminen paransi päällekkäinpiirtoa, mutta muutti kuitenkin efektin suunniteltua visualista ilmettä. Säteilyä haluttiin näkyvän enemmän maagisen ympyrän lähettyvillä, mutta liiallinen partikkelien määrä aiheutti aiemmin mainitusti ongelmia niiden elinajan lopulla. Efektin visuaalisen ilmeen palauttamiseksi partikkelien vähennyksen jälkeen pohdittiin toisen emitterin tekemistä säteille. Uusien säteiden lukumäärä olisi voitu asettaa useampaan kymmeneen, mutta partikkelit olisi säädetty hiipumaan paikallaan kasvattamatta niiden kokoa ja nopeutta elinajan lopulla. Säteilyä saataisiin siten näkymään enemmän halutussa kohdassa, mutta samalla pidettyä päällekkäinpiirto kurissa efektin loppuvaiheella. Emitterien lukumäärää tulisi kuitenkin optimoinnin periaatteiden mukaisesti pyrkiä vähentämään, eikä lisäämään. Lisäksi säteily oli ainoastaan sekundäärinen efektin osa, joka ei lopulta tarvinnutkaan niin merkittävää korostusta.

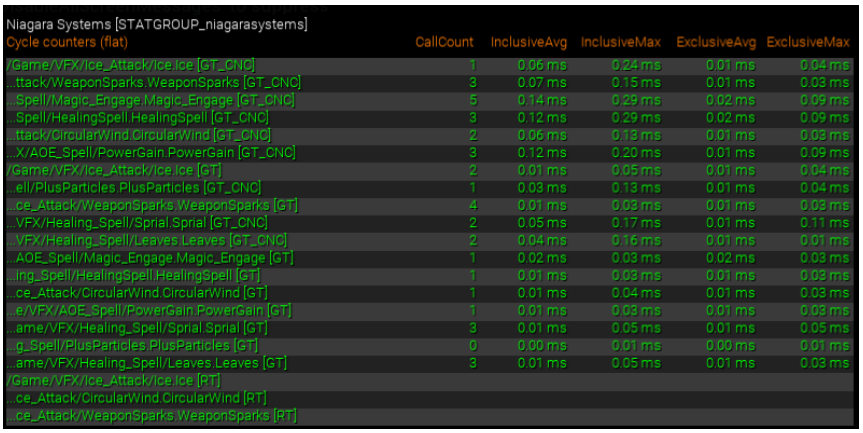
Uusi emitteri jätettiin siis toteuttamatta. Siitä huolimatta visuaalisen ilmeen ero muutosten seurauksena oli loppujen lopuksi melko pieni, mutta päällekkäinpiirron kannalta muutoksella oli parantava vaikutus. Kuvasta 34 voidaan nähdä muutosten merkitys verrattaessa edelliseen kuvaan.



Kuva 34. Säteilyefektin päällekkäinpiirto optimoinnin jälkeen

Kuvan efekti on kuvattu samana ajan hetkenä suhteessa sen elinikään kuin aiempi kuva 33. Päällekkäinpiirron määrää saatiin vähennettyä merkittävästi, ja väriasteikon valkoinen lukema saatiin lähestulkoon poistettua efektistä. Samaan tapaan partikkelien määrää vähennettiin muissakin efekteissä, joissa havaittiin suurta päällekkäinpiirtoa.

Niagara-järjestelmien tehokkuutta voitiin vielä tarkastella niiden tilastojen perusteella. Unreal Enginen komentoriville kirjoitettiin *stat niagarasystems*, jolloin järjestelmien tilastot saatiin auki pelin tason ollessa käynnissä (kuva 35).



Niagara Systems [STATGROUP_niagarasystems]
Cycle counters (flat)

	CallCount	InclusiveAvg	InclusiveMax	ExclusiveAvg	ExclusiveMax
/Game/VFX/Ice_Attack/Ice.Ice [GT_CNC]	1	0.06 ms	0.24 ms	0.01 ms	0.04 ms
..ttack/WeaponSparks.WeaponSparks [GT_CNC]	3	0.07 ms	0.15 ms	0.01 ms	0.03 ms
..Spell/Magic_Engage.Magic_Engage [GT_CNC]	5	0.14 ms	0.29 ms	0.02 ms	0.09 ms
..Spell/HealingSpell.HealingSpell [GT_CNC]	3	0.12 ms	0.29 ms	0.02 ms	0.09 ms
..ttack/CircularWind.CircularWind [GT_CNC]	2	0.06 ms	0.13 ms	0.01 ms	0.03 ms
..X/AOE_Spell/PowerGain.PowerGain [GT_CNC]	3	0.12 ms	0.20 ms	0.01 ms	0.09 ms
/Game/VFX/Ice_Attack/Ice.Ice [GT]	2	0.01 ms	0.05 ms	0.01 ms	0.04 ms
..ell/PlusParticles.PlusParticles [GT_CNC]	1	0.03 ms	0.13 ms	0.01 ms	0.04 ms
..ce_Attack/WeaponSparks.WeaponSparks [GT]	4	0.01 ms	0.03 ms	0.01 ms	0.03 ms
..VFX/Healing_Spell/Sprial.Sprial [GT_CNC]	2	0.05 ms	0.17 ms	0.01 ms	0.11 ms
..VFX/Healing_Spell/Leaves.Leaves [GT_CNC]	2	0.04 ms	0.16 ms	0.01 ms	0.01 ms
..AOE_Spell/Magic_Engage.Magic_Engage [GT]	1	0.02 ms	0.03 ms	0.02 ms	0.03 ms
..ing_Spell/HealingSpell.HealingSpell [GT]	1	0.01 ms	0.03 ms	0.01 ms	0.03 ms
..ce_Attack/CircularWind.CircularWind [GT]	1	0.01 ms	0.04 ms	0.01 ms	0.03 ms
..e/VFX/AOE_Spell/PowerGain.PowerGain [GT]	1	0.01 ms	0.03 ms	0.01 ms	0.03 ms
..ame/VFX/Healing_Spell/Sprial.Sprial [GT]	3	0.01 ms	0.05 ms	0.01 ms	0.05 ms
..g_Spell/PlusParticles.PlusParticles [GT]	0	0.00 ms	0.01 ms	0.00 ms	0.01 ms
..ame/VFX/Healing_Spell/Leaves.Leaves [GT]	3	0.01 ms	0.05 ms	0.01 ms	0.03 ms
/Game/VFX/Ice_Attack/Ice.Ice [RT]					
..ce_Attack/CircularWind.CircularWind [RT]					
..ce_Attack/WeaponSparks.WeaponSparks [RT]					

Kuva 35. Niagara-järjestelmien tilastot

Efektien tehokkuus on kuitenkin suhteellista pelin tavoiteltuun kuvataajuuteen nähden. Jos tavoiteltu kuvataajuus on esimerkiksi 60 kuvaa sekunnissa, yksi kuva eli *frame* esitetään aina noin 16,6 millisekunnin välein. Tilaston seuraamisen avulla voidaan havaita, mikäli jokin järjestelmä kuluttaa liian suuren osan kyseisestä ajasta. Kuvan 35 tilaston perusteella jokainen Niagara-järjestelmä kuluttaa hyvin pienen osuuden esimerkin mukaisesta 16,6 millisekunnista. Työn osalta efektejä voidaan pitää tehokkaina, koska kuvataajuuden

määrittämää saatavilla olevaa aikaa ei kulu paljon muuhun peleille tyypilliseen grafiikan piirtämiseen tai pelilogiikan pyörittämiseen.

Mikäli jokin järjestelmä havaitaan tilaston avulla kuormittavaksi, voidaan *stat niagaraemitters* konsolikomennolla tarkastella vielä yksittäisten emitterien tehokkuutta. Tällöin on mahdollista nähdä tarkemmin, mitkä järjestelmän emittereistä aiheuttavat kuormitusta.

7 TULOKSET

Työn tavoitteena oli toteuttaa 3 erilaista visuaalista efektiä pelihahmolle ja luoda selkeämpi kuva kehitysprosessista. Tämä tavoite saavutettiin, ja kehityksen aikana löydettiin vastauksia tutkimusongelmasta johdettuihin tutkimuskysymyksiin.

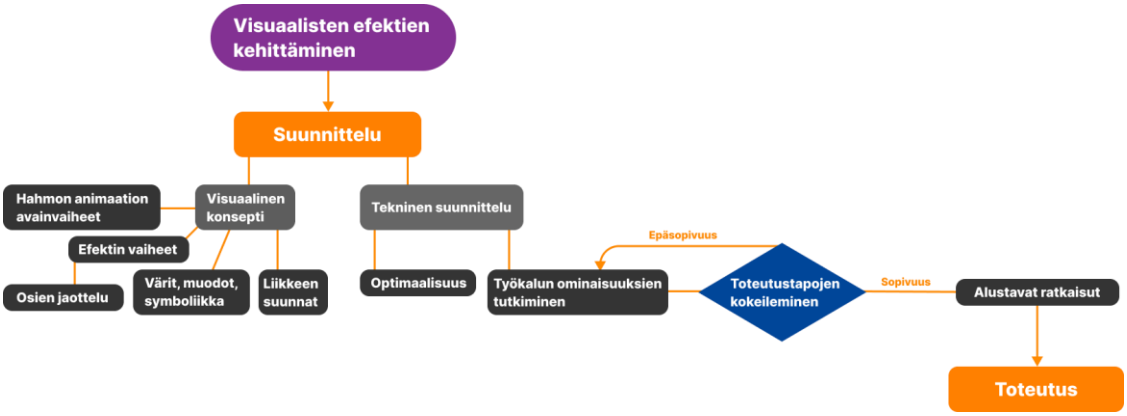
Mitä suunnittelussa olisi hyvä huomioida?

Efektin visuaalinen suunnitelma kiteytyy kuvitettuun konseptimateriaaliin. Konseptista on käytävä ilmi värit ja muotojen kieli sekä efektin vaiheet. Suunniteltaessa efektiä hahmolle on efektin vaiheiden lisäksi konseptiin syytä sisällyttää hahmon animaation eri vaiheita tai asentoja. Näin efektin vaiheet voidaan helpommin sovittaa animaatioon ja siten havainnollistaa kehityksen esivaiheessa, kuinka efektin tulisi kulkea animaation mukana.

Efektit koostuvat monesti useammasta eri osasta, joita voidaan erotella konseptissa. Osien jaottelussa voidaan mennä partikkelien tasolle, mikä helpottaa efektin teknistä suunnittelua, kun tiedetään millaisista osista efektin tulisi koostua. Osittelun pohjalta voidaan myöhemmin toteutusvaiheessa helpommin tehdä emittereitä, joista efekti koostuu.

Suunnittelun on myös katettava efektin tekninen suunnittelu. Teknisestä suunnittelusta on käytävä ilmi toteutuksen alustavat ratkaisut, jotta myöhemmin toteutusvaiheessa olisi helpompi edetä. Teknistä suunnittelua varten on tarkasteltava työkalun ominaisuuksia ja mahdollisesti etsittävä niistä dokumentaatiota soveltuvuuden selvittämiseksi. Teknisen suunnittelun vaiheesta edetään toteutusvaiheeseen, jossa suunnitelmat laitetaan käytäntöön.

Työn perusteella suunnitteluprosessista voitiin tehdä kuvan 36 mukainen havainnollistava kaavio.



Kuva 36. Suunnitteluvaiheen havainnollistus

Efektien kehittäminen lähtee siis käyntiin niiden suunnittelusta. Varsinaisessa prosessissa optimaalisten valintojen pohtiminen olisi hyvä aloittaa teknisen suunnittelun yhteydessä. Näin ei kuitenkaan toimittu työn osalta, sillä tehokkuuteen vaikuttavia tekijöitä halutiin tutkia ja vertailla valmiiden efektien avulla.

Miten kehitysympäristön työkalulla voidaan toteuttaa yksinkertaisia pelihahmon efektejä?

Niagara-työkalu tarjoaa efektien kehittämistä varten tarvittavat komponentit. Niagara-järjestelmään lisätään emittereitä, jotka sisältävät erilaisia vaiheita. Emitterin vaiheet huolehtivat partikkelien syntymisestä, alustamisesta, päivittämisestä sekä renderöinnistä. Vaiheiden alle lisätään moduuleja, jotka lopulta määrittävät partikkeliefektin käyttäytymisen ja ulkonäön. Efektien renderöintiä varten tarvitaan usein spesifejä materiaaleja, joita voidaan luoda Unreal Enginessä.

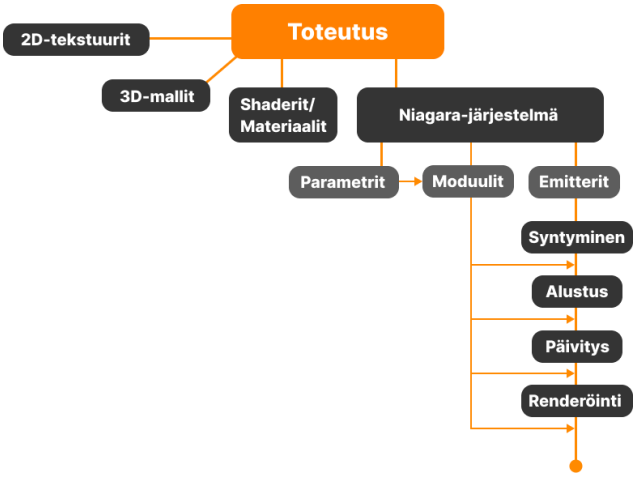
Useiden samojen moduulien huomattiin toistuvan lähes kaikissa työn aikana toteutetuissa efektien emittereissä. Partikkelien täytyy aina syntyä sopivalla Spawn-moduulilla. Jokaisen emitterin täytyy myös aina sisältää Initialize Particle -moduuli, jossa asetetaan partikkeleille alustavia perustason arvoja. Tämän lisäksi alustusvaiheessa asetetaan tyypillisesti partikkelien nopeus, suuntautuminen sekä syntymispisteet. Spesifien syntymispisteiden määrittämiseksi

emitteriin voidaan lisätä erilaisia Location-moduuleja. Näistä työn perusteella yleisimmäksi havaittiin ShapeLocation, mutta partikkelien käyttötarkoitus määrittelee niille parhaan syntymispisteen moduulin.

Päivitysvaiheeseen lisätään tyypillisimmin Scale-moduuleja, joilla skaalataan alustettuja arvoja, kuten nopeutta, kokoa tai väriä. Tällöin muutokset ilmenevät partikkelien elinaikana. Päivitysvaiheessa voidaan myös määrittää partikkelien pyörimisliike tai lisätä partikkeleille erilaisia vaikuttavia voimia Force-moduuleilla.

Renderöintivaiheeseen täytyy lisätä oikeanlainen renderöijä partikkelien tyypin perusteella. Renderöijiä on useampia eri tarkoituksiin, mutta työn aikana tarvittiin ainoastaan 2D- ja 3D-partikkelien renderöijiä.

Kehitysprosessin havainnollistavaa kaaviota jatkettiin suurpiirteisesti toteutuksen kuvaamisella (kuva 37).



Kuva 37. Toteutusvaiheen havainnollistus

Efektit vaativat osakseen muutakin, kuin pelkän Niagara-järjestelmän. Partikkelien ulkonäkö määritellään tekstuurien ja materiaalien avulla sekä usein myös 3D-malleja hyödyntäen.

Miten efektit saadaan integroitua hahmon animaatioihin?

Efektien liittäminen pelihahmoon tapahtuu Unreal Engine -pelimoottorissa hahmon animaatiosekvenssissä Notify-ilmoituksia hyödyntäen. Hahmon luurangon rakenteeseen voidaan lisätä uusia mukautettuja Socket-osia, jotka toimivat ikään kuin hahmon luiden jatkeina. Määritellyn luun tai Socketin perusteella efekti saadaan näkymään kyseisen osan kohdalla, ja Attached-ominaisuuden ollessa päällä efekti myös liikkuu osan mukana.

Play Niagara Particle Effect -tyyppinen Notify-ilmoitus liittää efektin animaatioon yksinkertaisesti, mutta se ei sovellu käytettäväksi jokaisen efektin kohdalla. Kyseiset partikkeliefektin Notify-ilmoitukset käytännössä vain asettavat efektin haluttuun animaation kohtaan, mutta pelkästään niiden avulla ei voida saada aikaan tiettyihin efekteihin niiden tarvitsemaa toiminnallisuutta. Kyseinen Notify-ilmoitus sopiikin staattisempiin efekteihin, mutta dynaamisemmat efektit vaativat usein koodia sisältäviä Blueprint-tyyppisiä Notify-ilmoituksia toiminnallisuutensa vuoksi.

Miten efektejä voidaan optimoida?

Efektien tehokkuuteen vaikuttaa muun muassa päällekkäinpiirron määrä, materiaalit sekä simuloiva suoritin. Simuloivan suorittimen osalta päätettiin työn osalta pysyä CPU-simuloinnissa, sillä partikkelimäärät eivät olleet minkään efektin kohdalla merkittäviä, eikä työn taustalla ollut mitään varsinaista peliä. GPU-simulointi on kuitenkin yleisesti ottaen hieman tehokkaampaa, mutta joissain tapauksissa puolestaan CPU-simulointi on parempi ratkaisu. Valintaa varten on arvioitava tehokkuuden kustannuksia etukäteen.

Päällekkäinpiirto on eräs merkittävimpiä efektin tehokkuuteen vaikuttavia tekijöitä, joka tapahtuu läpinäkyvien kerrosten piirtyessä toistensa päälle. Ongelmaan voidaan vaikuttaa muun muassa partikkelien lukumäärän vähentämisellä sekä partikkelien koon rajoittamisella. Lisäksi on syytä pyrkiä vähentämään läpinäkyvien materiaalityyppien käyttämistä ja sen sijaan käyttää kustannustehokkaampia materiaalityyppejä silloin kun partikkelien läpinäkyvyyden hyödyntäminen ei ole välttämätöntä.

Partikkelien määrän ja koon vähentäminen voivat kuitenkin vaikuttaa efektin haluttuun visuaaliseen ilmeeseen, minkä vuoksi optimointi olisi tärkeä huomioida jo suunnitteluvaiheessa. Efektin kustannuksia kannattaa pohtia etukäteen, sillä mitä enemmän saa aikaan vähemmällä, sitä parempi.

8 JOHTOPÄÄTÖKSET

Työn tavoitteena oli selvittää visuaalisten efektien kehityksen vaiheita. Tavoite jaettiin suunnittelu- ja toteutusvaiheeseen. Suunnitteluvaiheessa oli tarkoitus soveltaa jo olemassa olevia suunnittelumenetelmiä, mutta myös pohtia olisiko suunnitteluprosessiin syytä sisällyttää jotain muutakin. Muodot, värit ja ajoitus olivat tunnettuja suunnittelun kohteita, ja niitä voidaan pitää efektien suunnittelun peruspilareina. Näiden lisäksi efektin suunnitelmaan sisällytettiin aina hahmon animaation avainkohdat, jotta efektin vaiheita olisi helpompi sovittaa animaatioon. Jokaisessa vaiheessa efekti pilkottiin vielä pienemmiksi osiksi partikkelien tasolle, jotta voitiin helpommin havaita mistä osista efekti kokonaisuudessaan koostuu sekä paremmin suunnitella teknistä toteutusta. Johtopäätöksenä voidaan siis sanoa, että animaation avainkohtien ja efektien osittelun sisällyttäminen konseptiin selkeyttävät suunnittelutyötä sekä toteutusta.

Efektit toteutettiin lopulta suunnitelmien pohjalta. Kaikkia osia ei voitu toteuttaa kuten teknisen suunnittelun vaiheessa oli ajateltu. Joidenkin alustavien toteutustekniikoiden epäsopivuus huomattiin kehityksen aikana, ja ratkaisua oli pohdittava uudestaan. Niagara-työkalun ominaisuudet eivät olleet entuudestaan täysin tuttuja, eikä teknisen suunnittelun aikana tehty kartoitus ollut täysin perusteellista, mikä johti pariin virheelliseen suunnitelmaan. Siitä huolimatta efektien toteuttamiseksi löydettiin uudet ratkaisut, ja efektit saatiin lopulta toteutettua lähestulkoon täysin suunnitelmien mukaan. Näin ollen Niagara-työkalu siis soveltuu efektien tekemiseen.

Toteutuksen aikana etsittiin keinoja liittää efektit osaksi hahmon animaatioita. Ongelmaan löydettiin aluksi yksinkertainen ja helppo ratkaisu. Myöhemmin kuitenkin huomattiin, ettei ainoastaan tämä yhdenlainen ratkaisu sovi kaikille efekteille. Vaikka efektejä voitiin siis pääsääntöisesti liittää helposti animaatiosekvenssiin, oli mukautettujen ratkaisujen tekeminen Blueprint-koodin

avulla lopulta väistämätöntä. Efekteillä on peleissä usein erilaisia toiminnallisuuksia ja niiden tarvitsee kommunikoida pelin muiden osien kanssa, minkä vuoksi ne tarvitsevat koodia toimiakseen kunnolla. Erilaisten efektien toimintaa ja tarpeita varten täytyy siis usein pohtia yksilöllisiä keinoja.

Efektien optimointia tarkasteltiin työssä pääsääntöisesti päällekkäinpiirron osalta. Päällekkäinpiirtoa voidaan vähentää kontrolloimalla partikkelien määrää ja kokoa, mutta sitä on hankala eliminoida täysin, sillä läpinäkyvyyden käyttäminen on efekteissä monesti välttämätöntä. Kuormittavimmat osat on kuitenkin syytä havaita, ja päällekkäinpiirtoa pyrittävä vähentämään siellä missä se on mahdollista. Tämän seurauksena aiheutuvien mahdollisten visuaalisten muutosten tapahtuessa on pohdittava kyseisten osien merkittävyyttä muuhun efektiin nähden. Mikäli muutokset koskevat primääristä elementtiä, voi olla parempi selvittää tehokkaampi ratkaisu visuaalisuuden säilyttämiseksi. Puolestaan sekundääristen elementtien kohdalla visuaalinen menetys ei välttämättä ole niin merkittävä seikka.

Jatkokehitysmahdollisuudet

Tässä opinnäytetyössä keskityttiin lähinnä partikkeliefekteihin, mutta jatkokehitys avaa ovet monienkin erityyppisten visuaalisten efektien tutkimiselle. Tällaisia voi olla esimerkiksi pelkillä shadereilla eli Unreal Enginessä materiaaleilla toteutetut tehosteet, joilla voidaan saada erilaisia spesifejä vaikutuksia aikaan. Työn aikana toteutettiin muun muassa aseen jäätyminen lähes pelkästään materiaalin avulla. Lisää erilaisia peleissä tavattuja visuaalisia efektejä jäsenneltiin tarkemmin luvussa 3.2.

Efektit toimivat yksinkertaisesti hahmon kanssa, mutta pelimekaanisesti niillä ei ole toiminnallisuutta, sillä kehityksen taustalla ei ollut mitään varsinaista peliä. Parannusloitsun tulisi palauttaa pelihahmolle elämäpisteitä, iskuefektien olisi tarkoitus tehdä vahinkoa vihollisiin, ja efektien täytyisi yleisestikin huomioida vuorovaikutus muun ympäristön kanssa. Vihollisten tulisi puolestaan reagoida efektiin oikealla tavalla. Nämä seikat jäivät opinnäytetyön ulkopuolelle, joten efektien integroiminen varsinaisiin pelimekaniikkoihin voisi olla seuraava askel jatkokehityksen kannalta.

Lisäksi voitaisiin mennä vielä syvemmälle optimointiin. Optimointiin vaikuttavia seikkoja on varmasti enemmänkin, mitä tämän opinnäytetyön aikana saatiin selvitettyä. Materiaaleja voitaisiin tutkia tarkemmin ja selvittää kuinka käskyjen määrää voitaisiin konkreettisesti vähentää esimerkiksi yksinkertaistamalla joitakin matemaattisia operaatioita tai poistamalla mahdollisia turhia haaroja. Efektejä voitaisiin myös profiloida ja analysoida tarkemmin niiden tehokkuuden selvittämiseksi.

Myöhemmin tutkimusta voitaisiin kohdistaa myös tilattomiin emittereihin, jotka ovat Unreal Engine 5.4 -pelimoottorin esittelemä uusi ominaisuus. Kyseinen ominaisuus oli työn kehittämisen aikana sen verran uusi, että aihetta sivuttiin vain erittäin pintapuolisesti. Jatkokehityksenä voitaisiin esimerkiksi yrittää muuttaa työn aikana kehitettyjä perinteisiä emittereitä tilattomiksi emittereiksi, ja tutkia muutosten vaikutusta efektien tehokkuuteen.

9 POHDINTA

Työn toteutus oli pääosin hyvin onnistunutta. Suunnitelmien mukaisesti saatiin toteutettua kolme efektiä sekä tuotettua vastauksia tutkimuskysymyksiin. Työn aikana saatiin ratkaistua tutkimusongelma keräämällä teoriapohjaa valitusta työkalusta, sekä keräämällä aineistoa tutkimuskysymysten ja varsinaisen toteutuksen avulla.

Teoriaosuus tarjoaa hyödyllistä tietoa työkalusta, jonka moduuleihin keskitytään tarkemmin efektien toteutuksessa. Työn aikana kehitetyt efektit edustavat omaa tyylilajiaan ja niitä varten tehdyt tekniset ratkaisut soveltuvat hyvin tarkoitukseensa. Koska tutkimuksen aikana kehitettiin vain saman tyylilajin efektejä, ei työssä esitettyjä ratkaisuja voida pitää yleispätevinä aivan kaikkien tyylilajien partikkeliefekteille. Esimerkiksi realismia edustavat efektit vaatisivat tarkempaa ilmiöiden tutkimista ja yksityiskohtaisempia toteutustekniikoita.

Teoriaosuus myös selventää suunnitteluun vaadittuja tunnettuja osa-alueita, jonka lisäksi efektien suunnitteluvaiheessa esitetään niiden rinnalle muita hyödyllisiä suunnittelun keinoja. Suunnittelu olisi silti voinut olla joiltain osin perusteellisempaa. Esimerkiksi efektien konseptisuunnittelua olisi voitu laajentaa suunnittelemalla monipuolisemmin vaihtoehtoisia tyylejä ja muotoja erilaisille

efektin osille. Tämän avulla olisi voitu välttää tarve muuttaa joitakin tyylejä toteutuksen aikana. Lisäksi efektien hälvenemisvaiheet jäivät melko yksipuolisiksi. Teknisen suunnittelun vaiheessa saatiin avattua efektien alustavia toteutustapoja hyvin. Suunnitelmille olisi kuitenkin voitu toteuttaa havainnollistavampi esitystapa, joka varmasti toimisi selkeämmin myös todellisessa kehitysprosessissa yrityksen sisällä.

Maahan kohdistuvan iskuefektin liittäminen hahmon animaatioon aiheutti ongelmia, jotka ratkaistiin hieman heikoin keinoin. Myöhemmin tutkimuksen aikana kuitenkin selvitettiin, että animaatiosekvenssiin on mahdollista lisätä koodia sisältäviä Blueprint-ilmoituksia. Keinoa olisi mahdollisesti voitu soveltaa kyseisen ongelman ratkaisemiseen, mutta ensimmäisen efektin kohdalla tätä vaihtoehtoa ei ollut vielä osattu tutkia.

Efektien optimointiin ei työn aikana onnistuttu panostamaan niin laajasti kuin olisi toivottu. Efektejä saatiin optimoitua päällekkäinpiirron osalta, mutta esimerkiksi materiaalien optimointi jäi vähemmälle huomiolle. Materiaalien ohjeistusten määrän perusteella ne oli kuitenkin onnistuttu pitämään yksinkertaisina, sekä materiaalityyppien osalta oli tehty kustannustehokkaita valintoja mahdollisuuksien mukaan. Materiaalien tarkempi analysointi ja optimointi vaatisi kuitenkin laajempaa tietämystä niistä, mitä ei tämän opinnäytetyön rajoissa voitu saavuttaa.

Työn aikana saatiin kuitenkin kerättyä aineistoa, jota on mahdollista hyödyntää yksinkertaisten partikkeliefektien kehityksessä. Kaikkea ei ollut mahdollista käsitellä yhden opinnäytetyön mitoissa, mutta kerätyn aineiston avulla voidaan kuitenkin hahmottaa Niagara-työkalua, kehitysprosessia sekä sille hyödyllisiä ominaisuuksia selkeämmin.

LÄHTEET

Aguiar, G. 2022. Unity VFX Graph – Sword Slash Tutorial. Videoleike. Saatavissa: https://youtu.be/Er99e0OOBgc?si=Asq2uXzdHfKqm_C2 [viitattu 25.4.2024].

Aguiar, G. 2023. Unity VFX Graph – Arrow Projectile – Tutorial. WWW-dokumentti. Saatavissa: https://www.patreon.com/posts/unity-vfx-graph-83095149 [viitattu. 25.4.2024].

Animation Notifies s.a. Epic Games. WWW-dokumentti. Saatavissa: https://dev.epicgames.com/documentation/en-us/unreal-engine/animation-notifies-in-unreal-engine [viitattu 5.8.2024].

Anttila, P. 1998. Tutkimuksen taito ja tiedonhankinta. WWW-dokumentti. Saatavissa: https://metodix.fi/2014/05/17/anttila-pirkko-tutkimisen-taito-ja-tiedonhankinta/#top [viitattu 7.5.2024].

Archetti, S. 2024. The Meaning of Color: How to Use Color in Your Art. Blogi. Saatavissa: https://www.serenaarchetti.com/blog/the-meaning-of-colors-how-to-use-colors-in-your-art [viitattu 25.4.2024].

CPU and GPU Sprite Particles Comparison s.a. Epic Games. WWW-dokumentti. Saatavissa: https://dev.epicgames.com/documentation/en-us/unreal-engine/1.1---cpu-and-gpu-sprite-particles-comparison?application_version=4.27 [viitattu 4.8.2024].

Kananen, J. 2012. Kehittämistutkimus opinnäytetyönä: Kehittämistutkimuksen kirjoittamisen käytännön opas. Jyväskylä: Jyväskylän ammattikorkeakoulu.

Kananen, J. 2017. Kehittämistutkimus interventiotutkimuksen muotona: Opas opinnäytetyön ja pro gradun kirjoittajalle. Jyväskylä: Jyväskylän ammattikorkeakoulu.

Madeinshoreditch. 2023. Creating Stunning Game VFX: Insights from Industry Experts and Practitioners. WWW-dokumentti. Saatavissa: https://madeinshoreditch.co.uk/2023/02/17/creating-stunning-game-vfx-insights-from-industry-experts-and-practitioners/ [viitattu 25.4.2024].

Mad VFX. 2022. VFX Art Style Guide. Blogi. Saatavissa: https://www.mad-vfx.com/blogs/vfx-art-style-guide [viitattu 25.4.2024].

Mad VFX. 2023. VFX In Games. Blogi. Saatavissa: https://mad-vfx.com/visual-effects-in-games/ [viitattu 25.4.2024].

Marshall, J. s.a. What is VFX? A Guide to Visual Effects in Film. WWW-dokumentti. Päivitetty 16.2.2023. Saatavissa: https://www.backstage.com/magazine/article/create-digital-effects-film-project-13748/ [viitattu 13.10.2024].

Niagara Lightweight Emitters Overview s.a. Epic Games. WWW-dokumentti. Saatavissa: https://dev.epicgames.com/documentation/en-us/unreal-engine/niagara-lightweight-emitters-overview [viitattu 15.10.2024].

Niagara Overview s.a. Epic Games. WWW-dokumentti. Saatavissa: https://dev.epicgames.com/documentation/en-us/unreal-engine/overview-of-niagara-effects-for-unreal-engine?application_version=5.0 [viitattu 25.4.2024].

Optimizing Niagara | Scalability and Best Practices. 2024. Epic Games. WWW-dokumentti. Päivitetty 17.6.2024. Saatavissa: https://dev.epicgames.com/community/learning/tutorials/15PL/unreal-engine-optimizing-niagara-scalability-and-best-practices [viitattu 15.10.2024].

Reddit. 2019. Custom Toon Smoke Shader Using a Particle System in Unity. WWW-dokumentti. Saatavissa: https://www.reddit.com/r/gamedev/comments/eehm42/custom_toon_smoke_shader_using_a_particle_system/ [viitattu 18.10.2024].

Riot Games. 2017. The Complete Guide to Creating Visual Effects Within League of Legends. PDF-dokumentti. Saatavissa: https://nexus.leagueoflegends.com/wp-content/uploads/2017/10/VFX_Styleguide_final_public_hidpjqwx7lqyx0pjj3ss.pdf [26.4.2024].

Riot Games. 2018. So You Wanna Make Games? | Episode 7: Game VFX. Youtube. Videoleike. Saatavissa: https://youtu.be/3QKK2o5rWSQ?si=T96R88YEuGr2cewx [viitattu 25.4.2024].

Shestakova, K. 2024. Materials compilation: shader complexity and optimisation. WWW-dokumentti. Saatavissa: https://kseniia-shestakova.medium.com/materials-compilation-shader-complexity-and-optimisation-f60be9a9357a [viitattu 15.10.2024].

Steam. 2016. Enter the Gungeon. WWW-dokumentti. Saatavissa: https://store.steampowered.com/app/311690/Enter_the_Gungeon/ [viitattu 8.11.2024].

Steam. 2019. Red Dead Redemption 2. WWW-dokumentti. Saatavissa: https://store.steampowered.com/app/1174180/Red_Dead_Redemption_2/ [viitattu 8.11.2024].

Steam. 2024. God of War Ragnarök. WWW-dokumentti. Saatavissa: https://store.steampowered.com/app/2322010/God_of_War_Ragnark/ [viitattu 8.11.2024].

The Difference Between Game and Movie VFX s.a. Magicmedia. WWW-dokumentti. Saatavissa: https://magicmedia.studio/news-insights/the-differences-between-game-and-movie-vfx/ [viitattu 22.4.2024].

The Ultimate Guide to VFX for Gaming: From Explosions to Environments s.a. Magicmedia. WWW-dokumentti. Saatavissa: https://magicmedia.studio/news-insights/guide-to-vfx-for-gaming/ [viitattu 18.4.2024].

Veselinovikj, B. 2024. VFX in Gaming: The Ultimate Guide to Visual Effects in Video Games. Blogi. Saatavissa: https://borisfx.com/blog/vfx-in-gaming-ultimate-guide-visual-effects-games/ [viitattu 13.10.2024].

VFX Optimization Guide s.a. Epic Games. WWW-dokumentti. Saatavissa: https://dev.epicgames.com/documentation/en-us/unreal-engine/vfx-optimization-guide?application_version=4.27 [viitattu 4.8.2024].

Wirtz, B. s.a. The Power of Experience: The Wonders of Video Game Immersion. Gamedesigning. WWW-dokumentti. Päivitetty 4.7.2023. Saatavissa: https://www.gamedesigning.org/learn/game-immersion/ [viitattu 22.4.2024].