Niklas Taipalinen

Visuaaliset tehosteet videopeleissä

Tradenomi

Tietojenkäsittely

Kevät 2020

KAMK • University of Applied Sciences

Tiivistelmä

Tekijä(t): Taipalinen Niklas

Työn nimi: Visuaaliset tehosteet videopeleissä

Tutkintonimike: tradenomi, tietojenkäsittely

Asiasanat: 3D**,** vfx, varjostin, partikkeli, UV-kartta

Opinnäytetyön tavoitteena oli perehtyä visuaalisiin tehosteisiin ja niiden tekemiseen. Tutustumisen pohjalta tehtiin Unity-pelimoottorissa moniosainen meteoritaikaisku.

Tehosteita voidaan hyödyntää melkein kaikkialla videopeleissä. Aina tunnelmaa luovista huomaamattomista tehosteista valtaviin pelin päättäviin räjähdyksiin. Ennen tehosteiden tekemistä tulee perehtyä väriteoriaan, ajoitukseen, optimointiin ja peliin sopivuuteen. Tehosteet voivat koostua monista eri osista, joihin kuuluvat partikkelit, 3D-objektit, varjostimet, tekstuurit, sprite-arkit ja animaatio. Tehosteiden luomiseen käytetään tyypillisesti monia eri sovelluksia työtehtävän mukaan. Yleisimpiä sovelluksia ovat pelimoottorit kuten Unity, kuvanmuokkausohjelmat kuten Adobe Photoshop ja 3D-mallinnusohjelmat kuten Blender. Jokaisella graafikolla on kuitenkin omat suosikkiohjelmat eri työtehtäviin, koska ei ole vain yhtä oikeaa tapaa tehdä tehoste.

Opinnäytetyön soveltavassa osuudessa toteutettiin meteoritaikaisku, jossa hyödynnettiin kaikkia opinnäytetyössä käytyjä tekniikoita ja työtapoja. Meteoritaikaiskussa on monta eri osaa ja kaikki niistä soveltavat eri tekniikoita. Tehosteessa myös tutkitaan Unity High Definition Render-putkea, joka on tehty auttamaan realismin tuontiin projektissa.

KAMK • University of Applied Sciences

Abstract

Author(s): Taipalinen Niklas

Title of the Publication: Visual Effects in Videogames

Degree Title: Bachelor of Business Information Technology, Game Development

Keywords: 3D, vfx, shader, particle, UV-map

The goal of this thesis was to take a look at visual effects and how they are made. Based on the knowledge from that a meteor strike magic was created in the Unity game-engine.

Visual effects can be used almost everywhere in video games, from setting the mood with near unnoticeable effects, to effects that end the whole game with a massive explosion. Before creating visual effects one has to learn about color theory, timing, optimization and game suitability. Visual effects can consist of many different parts, including particles, 3D-objects, shaders, textures, sprite sheets and animation. Many different programs, depending on the task, are often used in the creation of visual effects. Most common programs are game engines like Unity, photomanipulation programs like Adobe Photoshop and 3D-modeling programs like Blender. However every artist has their own preferred programs as there is no one correct way to create a specific visual effect.

In the practical part of this thesis a meteor strike magic will be created, which uses all techniques mentioned in this thesis. The meteor strike magic has multiple parts and all of them use a different technique from the thesis. The thesis will also be looking into Unity High Definition Render pipeline, which was created to help bring realism into projects

Sisällys

Symboliluettelo

3D-malli

Kolmiulotteinen virtuaalimalli, jota voidaan käyttää tehosteiden pohjana.

HDR-putki

High Definition Render Pipeline Unity-pelimoottorissa. Putki hallitsee miten objektit näytetään pelissä ja miten pelimoottorin eri osat käyttäytyvät. HDR-putki on Unityn realismiin keskittyvä putki.

Immersio

Tila, jossa pelaaja pystyy samaistumaan pelimaailmaan ja unohtaa hetkellisesti kyseessä olevan vain virtuaalinen maailma tai asia.

Proseduraalinen

Työtapa, jossa tietokone auttaa asian luomisessa algoritmien avulla, esimerkiksi luomalla ja yhdistämällä monta kuvaa sekä liikuttamalla niitä.

Sprite-arkki

Kuva, joka sisältää tehosteen eri vaiheet pieninä kuvina. Kaikki kuvat ovat tasaisin välein aseteltuna ruudukkoon, jotta tehoste osaa lukea ne oikeassa järjestyksessä ja näyttää kuvat yksitellen.

Tekstuuri atlas

Iso kuva, joka sisältää monta pienempää tekstuuria, josta pelimoottori laitetaan valitsemaan oikea tekstuuri käyttötarkoitukseen. Tämän avulla pelimoottorin ei tarvitse ladata montaa eri kuvaa samanaikaisesti, säästäen muistia.

Varjostin

Koodi, jolla voidaan säädellä eri asioita tehosteista. Kommunikoi näytönohjaimelle, että miltä asian kuuluisi näyttää pelissä.

1 Johdanto

Visuaaliset tehosteet ovat kuin kakun kuorrutus ja koristeet. Kakku maistuu hyvältä ilmankin niitä, mutta niiden kanssa kokemus on aivan erilainen. Visuaaliset tehosteet tehostavat pelin tuntuman, maailman ja pelaajan nautinnon paljon paremmaksi oikein käytettynä. Sitä voidaan hyödyntää melkein kaikkialla, tunnelman luontiin maailmassa puolihuomaamattomilla tehosteilla, valtaviin ruudun täyttäviin räjähdyksiin. Kuitenkin aina tehosteita tehdessä täytyy ottaa huomioon niiden käyttötarkoitus ja sopivuus muiden tehosteiden kanssa. Jos näyttö on koko ajan täynnä valtavia monimutkaisia tehosteita, niin pelaaja ei tiedä, mihin hänen pitäisi katsoa tai keskittyä.

Tämän opinnäytetyön tarkoitus on antaa eri käytännöistä ja työkaluista hyvä pohjatieto, joka toimii alustana tulevalle oppimiselle. Opinnäytetyö sisältää myös paljon teoriaa, joka on hyvä tietää ennen tehosteiden tekemistä. Opinnäytetyössä näytetään useita eri työtapoja ja työkaluja, jotta niistä voi valita tehtävään sopivan. Kaikkia ei pysty, eikä tarvitse käyttää jokaisessa tehosteessa, mutta niiden olemassaolosta on hyvä tietää, sillä ne voivat tulevaisuudessa auttaa.

Opinnäytetyössä tullaan keskittymään enimmäkseen Unity-pelimoottorilla tehosteiden tekemiseen, mutta samat käytännöt toimivat myös useilla muilla pelimoottoreilla.

Opinnäytetyön tavoitteena on tutkia visuaalisia tehosteita videopeleissä ja niiden tekemistä. Mikä tekee tehosteesta hyvän ja mitä asioita kannattaa välttää? Lopuksi tietoa käytetään luomaan moniosainen meteoritaikaisku, joka hyödyntää kaikkea opinnäytetyössä käytyä.

2 Visuaaliset tehosteet

Visuaaliset tehosteet ovat tapa tehdä pelistä näyttävämpi, ohjata pelaajaa, antaa hänelle tietoa, sekä paljon muuta. Tehosteet usein koostuvat partikkeleista, varjostimista, 3D-malleista ja animaatiosta, mutta on muitakin tapoja tehdä niitä. Kaikilla graafikoilla on omat tavat tehdä tehosteita. Jotkut tekevät kaiken käsin, mutta jotkut nojaavat enemmän proseduraaliseen eli algoritmeilla luotuun työtapaan. Tehosteisiin tarvittavat työtavat usein myös vaihtelevat paljon eri pelien välillä. Realistisessa pelissä todennäköisesti käytetään enemmän proseduraalisesti tehtyjä tehosteita tai jopa valokuvien tai videoiden pohjalta tehtyjä, kun taas maalatussa 2D-pelissä todennäköisesti käytetään käsin maalattuja sprite-arkkeja.

2.1 Visuaalisten tehosteiden hyödyt

Visuaaliset tehosteet tuovat pelin kokemuksena seuraavalle tasolle ja saavat pelin näyttämään paljon paremmalta oikein tehtynä. Ilman tehosteita peli ei näyttäisi yhtä hyvältä, ja esimerkiksi hyökkäykset eivät tuntuisi yhtä voimakkailta.

Tehosteiden vaikutuksesta on hyvänä esimerkkinä peli Wolfenstein: Enemy Territory. Pelissä on kaksi eri puolta, jotka taistelevat keskenään. Vaikka molemmilla puolilla on erilaiset aseet, niin niiden kaikki arvot, kuten rekyyli ja tehty vahinko, ovat kuitenkin samat. Aikoinaan pelaajat kuitenkin luulivat, että eri puolien aseilla oli isojakin eroja, esimerkiksi että Thompson ampuu hitaammin, mutta tekee enemmän vahinkoa, mutta MP40 on nopeampi, mutta tekee vähemmän vahinkoa. Tämä kuitenkaan ei ollut totta, sillä aseet olivat täysin identtiset ulkonäköä lukuun ottamatta. Miksi sitten pelaajat luulivat niitä erilaisiksi? Aseiden äänet olivat liian erilaiset. Thompsonilla oli hitaampi, mutta voimakkaampi ääni, toisin kuin MP40 aseella. Tämän takia pelaajat luulivat Thompsonin olevan voimakkaampi. Vaikka tämä esimerkki oli ääniefekteistä eikä visuaalisista tehosteista, niin sama asia pätee niihinkin [1].

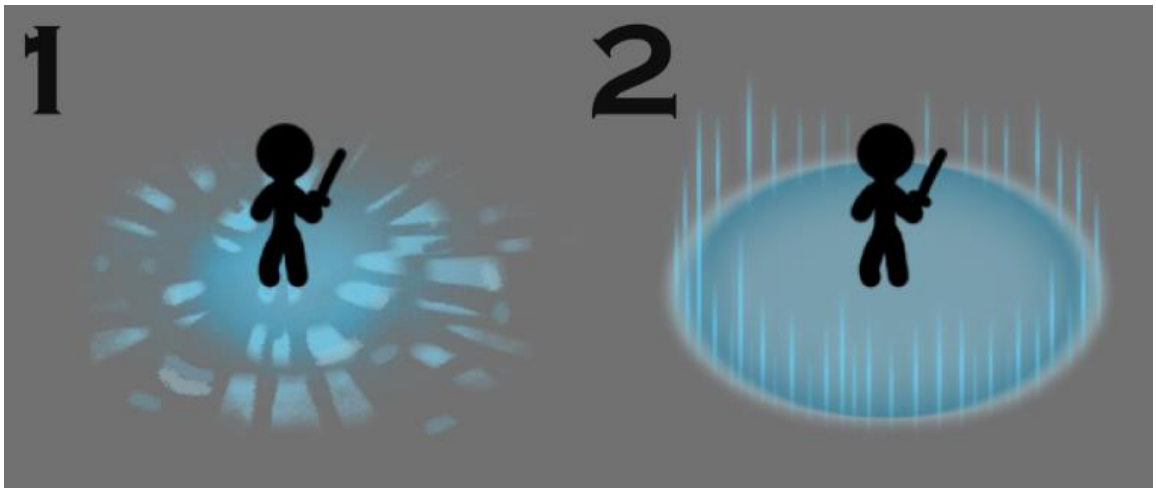
Sama asia huomattiin tehdessämme CONTINUE-peliä, jossa toimin tehostegraafikkona. CONTINUE on tietokoneelle tehty 3D Hack n’ Slash peli, jossa pelaajalla oli kaksi hyökkäystä: kevyt hyökkäys, joka oli nopea mutta ei tehnyt niin paljoa vahinkoa, sekä voimakas ladattava hyökkäys, joka teki enemmän vahinkoa latausajan mukaan. Aivan alussa ladattava hyökkäys oli aivan liian voimakas verrattuna kevyeen hyökkäykseen, mutta ladattavalla hyökkäyksellä ei ollut kunnollista tehostetta, kuten kevyellä hyökkäyksellä. Tämä johti pelaajat uskomaan, että se ei ollut niin hyvä,

ellei sitä ladannut pitkään. Toisella eri testauskerralla ladattavan hyökkäyksen tehoste oli kuin räjähdys jo ensimmäisellä tasolla, vaikka tehtyä vahinkoa oli vähennetty. Tämä johti pelaajat käyttämään sitä paljon enemmän kevyeen hyökkäykseen verrattuna, joka tässä versiossa oli paljon parempi.

Näistä esimerkeistä voidaan havaita, kuinka tärkeitä tehosteet peleissä ovat. Pelaajat olettavat, että jos hyökkäyksessä on voimakas tehoste, niin silloin hyökkäys on voimakas. Eli on todella tärkeää, että tehoste vastaa käyttötarkoitustaan, eikä ole liian näyttävä tai liian huomaamaton.

Visualliset tehosteet ovat myös erittäin tärkeitä työkaluja pelaajan opastamisessa. Esimerkiksi jos pelaajalla on monta eri hyökkäystä, niin tehosteella voi viestiä, kuinka voimakkaita ne ovat. Sama pätee myös vihollisiin. Tehosteilla voi viestiä, että milloin vihollinen hyökkää ja minne hyökkäys osuu, jotta pelaaja ehtii väistämään. Esimerkiksi jos kesken taistelun maahan ilmestyy punaisia alueita, pelaaja automaattisesti ajattelee, että ne ovat vaarallisia, kun taas vihreä tai sininen alue saattaisi saada heidät ajattelemaan, että siinä seisominen voisi auttaa taistelussa. Jos pelissä käytetään tehostetta näyttämään alueita, niin tehosteen rajojen täytyy olla selkeät [2].

Kuten kuvasta 1 voi huomata, ensimmäisen version aluetta on todella vaikea hahmottaa, etenkin jos sitä pitäisi osata varoa kesken nopeatempoisen taistelun, kun taas toisessa kuvassa alueen pystyy näkemään todella nopeasti.



Kuva 1. Esimerkki tehosteiden rajoista [2]

2.2 Visuaalisten tehosteiden käyttötarkoitus

Visuaalisia tehosteita voidaan käyttää melkeinpä kaikkialla. Aina valokeilassa näkyvistä pölyhiukkasista valtaviin ydinräjähdyksiin.

Tehosteita voidaan käyttää realismin tuontiin ja tunnelman luontiin. Puolihuomaamattomat tehosteet ympäristössä, esimerkiksi sadepisarat vesilammikossa tai tulikärpäset pimeässä metsässä, ovat helppo tapa saada peli näyttämään paremmalta. Nämä tehosteet oikein tehtynä auttavat todella paljon tunnelman luomisessa ja saavat pelimaailman tuntumaan paljon immersiivisemmältä ja aidolta. Esimerkiksi pienet aallot vedessä ja heiluva ruoho ovat hyviä tapoja osoittaa pelaajalle, että pelimaailmassa on kevyttä tuulta ja luoda aitouden tuntua.

Hyökkäyksissä tehosteet ovat erittäin tärkeässä osassa. Niillä viestitään pelaajalle, millainen hyökkäys on kyseessä, mitä se tekee, kuinka ison alueen se kattaa, kuinka voimakas hyökkäys on kyseessä ja paljon muuta. Tehosteilla ja animaatiolla voidaan myös näyttää, milloin vihollinen on hyökkäämässä, tai jopa lataamassa isoa hyökkäystä, joka pelaajan täytyy väistää [3].

Tehosteella voidaan viestittää monia eri asioita. Esimerkiksi jos pelaaja näkee alueen täynnä punaisena hehkuvaa ja kuumuudesta höyryävää laavaa, niin todennäköisesti pelaaja osaa yhdistää, että sinne ei kannata mennä ja että se on vaarallinen. Sen sijaan outo laite, joka hohtaa vihreänä, vaikuttaa siltä, että se voisi auttaa pelaajaa.

3 Tärkeitä huomioita tehosteita tehdessä

Tehosteita tehdessä täytyy ottaa monta eri asiaa huomioon. Esimerkiksi heikolle hyökkäykselle ei kannata tehdä pelin hienointa ja isointa tehostetta ja tunnelmaa tuovalle taustatehosteille ei kannata tehdä näyttävää tehostetta.

On todella tärkeää jo ennen tehosteen konseptointia puhua suunnittelijalle ja varmistaa, että molemmilla on hyvä käsitys tehosteen käyttötarkoituksesta. Tämä auttaa paljon ylimääräiseltä päänvaivalta ja helpottaa suunnittelemista.

Tehostetta tehdessä täytyy ottaa monia eri asioita huomioon, mutta kaikista tärkein on tehosteen luettavuus. Jos kyseessä on esimerkiksi hyökkäys, pelaajan täytyy heti saada selville, millainen hyökkäys on kyseessä, mihin se osuu, kuinka isolle alueelle, ja kuinka voimakas se on. Näitä voidaan säätää esimerkiksi muodolla ja värillä. Jos kyseessä on voimakas hyökkäys, niin se voi esimerkiksi hohtaa kirkkaana. Myös osumisalue täytyy olla helposti hahmotettavana, jotta pelaajan ei tarvitse arvata, missä on turvallista ja missä ei [4].

On myös tärkeää, ettei käytä liikaa aikaa yhteen tehosteeseen. Alussa täytyy laittaa tehosteet tärkeysjärjestykseen ja päättää, kuinka paljon aikaa minkäkin tehosteen tekemiseen käytetään. Jos säästää tärkeimmän tehosteen viimeiseksi ja alussa käytti liikaa aikaa huomaamattomiin taustatehosteisiin, niin aika voi hyvinkin loppua kesken, johtaen huonompaan laatuun tärkeissä tehosteissa.

Monet aloittelijat eivät usein ota huomioon, kuinka vaativa tehoste on kohdelaitteen tehojen kannalta ja vain keskittyvät ulkonäköön. Tämä voi johtaa siihen, että peli alkaa pätkimään tai jopa kaatuu, kun kyseinen tehoste tapahtuu pelin aikana.

3.1 Tehosteen tärkeys

Ensimmäinen asia, joka tulee ottaa huomioon tehostetta suunnitellessa, on tehosteen käyttötarkoitus ja tärkeys. Näiden mukaan päätetään, kuinka näyttävä tehosteen tulee olla ja kuinka paljon aikaa sen tekemiseen kannattaa varata. Jos kyseessä on vain pieni taustatehoste, jota pelaaja ei tule edes huomaamaan todennäköisesti, niin sen ei kannata olla todella näyttävä, eikä sen teke-

miseen kannata käyttää paljoa aikaa. Jos sen sijaan kyseessä on esimerkiksi pelaajahahmon kaikista voimakkain hyökkäys, niin sen hiomiseen kannattaa käyttää paljon aikaa ja huomiota. Vaikka sitä tulisi näkemään harvemmin, niin pelaajan täytyy tuntea iskun voimakkuus ja olla innoissaan, kun pääsee käyttämään sitä. Jos voimakkaimmalla iskulla on heikko tehoste, niin pelaaja ei todennäköisesti sitä käytä paljoa tai ajattele sen olevan voimakas [5].

Heikoin isku, jota pelaaja käyttää valtaosan ajasta, ei kuulu olla näyttävä. Sen täytyy viestiä pelaajalle, että kyseessä on vain heikko isku, jota käytetään, kun muut iskut eivät ole valmiina käytettäväksi. Tämä ei kuitenkaan tarkoita, että se kannattaisi vain tehdä pois alta puoleen tuntiin. Vaikka tehoste on heikko, niin pelaaja tulee näkemään sitä usein. Eli vaikka tehoste ei ole todella näyttävä, niin sen silti täytyy näyttää silmää miellyttävältä.

Merkityksettömimmille tehosteille, esimerkiksi taustoissa käytettäville, kannattaa luoda tapa tehdä niitä nopeasti, jotta tarvittaessa uusiin tehosteisiin ei menisi paljoa aikaa. Esimerkiksi jos tekee tulitehosteen, niin sen voi pilkkoa eri osiin. Tästä yhdestä tehosteesta saa tehtyä monia taustatehosteita kuten savua, kipinöitä, soihdun liekki tai vaikka käyttää osana isompia iskuja. Jos täytyy tehdä paljon erilaisia tehosteita yhteen peliin, niin aina kannattaa etsiä tapoja nopeuttaa työtä hyödyntämällä osia vanhoista efekteistä. Ei kuitenkaan ole hyväksi, jos kaikki tehosteet näyttävät samalta, eli kannattaa yhdistää uutta ja vanhaa.

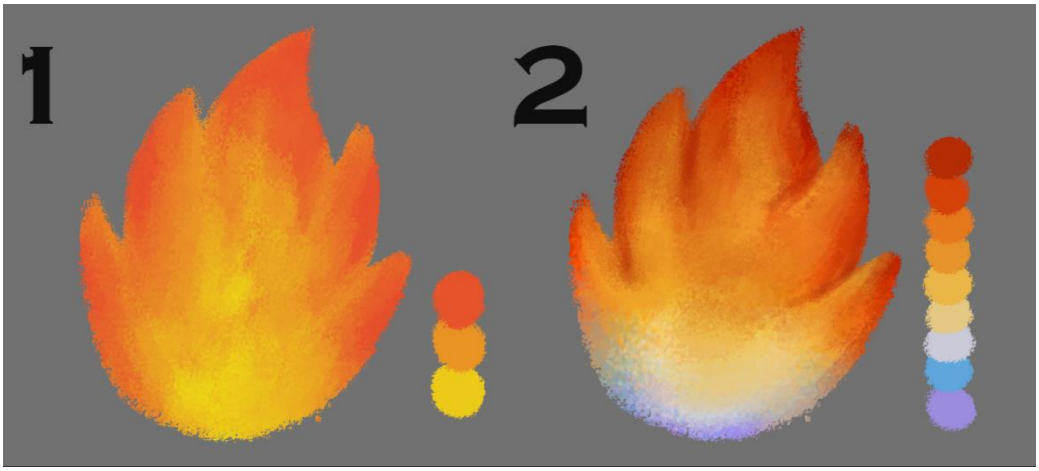
3.2 Värit

Värit ovat yksi tärkeimmistä asioista tehosteita tehdessä, mutta myös yksi vaikeimmista tehdä oikein [6]. Väriteorian hallitseminen on erittäin tärkeää, ja sitä kannattaa jo opetella ennen tehosteen tekemistä. Värillä voidaan säätää monia asioita tehosteessa, kuten miten pelaaja ymmärtää sen. Esimerkiksi vihreä on usein auttava väri, kun taas punainen on vaarallinen [7]. Tämä kuitenkin voi vaihdella eri kulttuurien välillä, ja jos peli käyttää värejä symboloimaan asioita, niin kannattaa värien tarkoituksiin eri kulttuureissa tutustua paremmin [8].

Tehosteen värikylläisyydellä myös voidaan säätää tärkeyttä. Tehosteita kannattaa tutkia mustavalkoisena ennen peliin laittamista. Jos pelaajan kuuluu huomata tehoste heti ja reagoida siihen, niin tehosteen täytyy olla hyvin näkyvä myös mustavalkoisena, samoin kuin erivärisiä taustoja vasten. Esimerkiksi osat tehosteesta, jotka voivat osua pelaajaan, kannattaa olla korkealla värikylläisyydellä ja visuaalista kiinnostavuutta tuovat osat vähäisemmällä. Esimerkiksi jos pelissä on taika, joka ampuu tulipallon, niin itse tulipallo kannattaa olla korkealla kontrastilla, mutta takana

lentävä savu ja kipinät kannattaa olla tummemmalla kontrastilla. Jos esimerkiksi tehosteen jälkeen jättämän hännän tekee myös kirkkaalla kontrastilla, niin pelaaja voi ajatella, että sekin tekee vahinkoa häneen. Tämä myös auttaa värisokeita pelaajia näkemään tärkeät tehosteet. Myös eri saturaatioiden käyttäminen tehosteessa tekee siitä paljon kiinnostavamman kuin se, että kaikki osat tehosteesta olisi samalla värikylläisyysarvolla tehtyä.

Tehosteissa kannattaa myös käyttää eri värejä ja pieniä sävyeroja. Tämän avulla tehosteista tulee paljon kiinnostavamman näköisiä. Esimerkiksi lisäämällä keltaista, valkoista tai sinistä liekkiin siitä saadaan paljon kuumemman ja kiinnostavamman näköinen (kuva 2). Myös muissa tehosteissa tummien värien lisääminen korostaa tehosteen kirkkaita osia.



Kuva 2. Esimerkki kiinnostavista sävy-yhdistelmistä

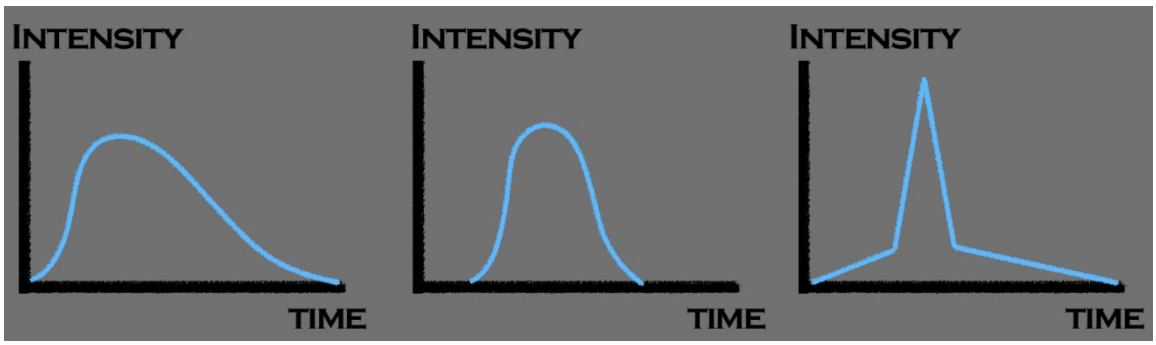
Kannattaa kuitenkin välttää käyttämästä samassa tehosteessa kahta eri sävyä pääroolissa, jotka ovat väriympyrän vastapuolilla, esim. punainen ja vihreä (kuva 3). Tämä luo tehostein sekavuutta ja taistelee pelaajan huomiosta.


Kuva 3. Väriympyrä [9]

Myös pienillä värieroilla voidaan vaihtaa tehosteen elementtiä, esimerkiksi jos pelissä on taikaohjus ja jääohjustaika ja molemmat niistä ovat sinisiä, niin taikaohjuksen voi erottaa jäästä lisäämällä siihen hieman purppuraa tai keltaista. Myös vihreä, joka normaalisti on positiivinen väri, voidaan saada näyttämään vaaralliselta tekemällä siitä hieman tummempaa ja lisäämällä purppuraa. Tehosteissa kannattaa kuitenkin välttää aivan puhdasta valkoista ja mustaa, sillä ne eivät ole niin silmäystävällisiä ja usein säästetään käyttöliittymälle.

3.3 Ajoitus

Tehosteen ajoitukset ovat erittäin tärkeitä pelin sekä visuaalisen kiinnostavuuden kannalta. Jos kyseessä on hyökkäys, joka pelaajan pitäisi pystyä väistämään, niin tehoste ei voi alkaa täydellä teholla. Tehosteessa täytyy olla jokin alkuosa, jonka pelaaja pystyy näkemään ja tulkitsemaan tulevaksi hyökkäykseksi. Toisaalta loitsu, joka auttaa pelaajaa, voi alkaa voimakkaana ja hitaasti pienentyä tai haihtua olemattomiin. Tehosteet, jotka ovat tasaisen voimakkaita koko ajan tai katoavat täysin tasaisella voimakkuudella, eivät yleensä ole yhtä kiinnostavia. Esimerkiksi jos kauhupelissä on rikkinäinen kattolamppu, niin on paljon kiinnostavampaa, jos siitä tulevan valon voimakkuus hyppää pimeästä täyteen valoisuuteen heti ja samalla nopeudella sammuisi, kuin se, että se hitaasti menisi täyteen valoisuuteen ja alkaisi taas himmentymään yhtä tasaisesti. Kuvassa 4 nähdään kolme eri esimerkkiä eri ajoituksista. Ensimmäinen tehoste on suhteellisen pehmeä ja on näytöllä koko elinajan. Toinen on suhteellisen intensiivinen, mutta ei kestä kovin kauaa. Kolmas kestää koko elinajan, mutta itse tehosteen osumisaika on todella räjähtävä, mutta alku ja loppu käytetään vain visuaalisen kiinnostavuuden tuontiin.



Kuva 4. Esimerkkejä tehosteen ajoituksista

Tehosteen täytyy aina vastata pelin toimivuutta. Jos pelaaja tai vihollinen hyökkää miekalla ja siinä on tehoste, niin tehoste ei voi olla miekkaa edellä tai paljon jäljessä, vaan sen täytyy mennä

miekan mukana. Jotkut animoijat kuitenkin animoivat hahmot siten, että miekka menee alusta loppuun niin nopeasti, ettei sitä ehdi nähdä (tai kokonaan jättävät välivaiheet animoimatta) tehden animaatiosta todella napakan. Tässä tilanteessa tehosteen voi väläyttää, kun miekka on loppuasennossa. Tällaisten tilanteiden takia tehostegraafikon pitää aina tehdä yhteistyötä etenkin animoijan kanssa, jotta tehosteet sopivat hahmojen liikkeisiin. Jos ajoitus ei ole oikein, niin se voi pistää silmään pelatessa ja hajottaa immersion.

3.4 Peliin sopivuus

Pelin tyyliin sopivan tehosteen tekeminen voi usein olla haastavaa. Jos koko peli on tehty maalatulla tyylillä, niin ei ole suositeltavaa tehdä realistisia tehosteita. Jos itse ei esimerkiksi osaa piirtää samalla tyylillä kuin mitä pelissä on, niin hyvä vaihtoehto on tehdä pohja tehostelle ja pyytää tiimin muita artisteja nopeasti maalaamaan päälle. Jos kyseessä on minimalistinen peli, niin usein voi vahingossa tehdä liian monimutkaisia tehosteita. Tässä tilanteessa täytyy vain aina pitää mielessä rajoitukset ja yrittää tehdä sopivia tehosteita. Väreissä kannattaa myös ottaa huomioon koko peli. Jos peli on tehty täysin pastellivärеillä, niin tehosteita ei kannata tehdä kirkkailla neonvärеillä, mutta värit eivät saa täysin hukkua taustaan myöskään. Tyylitellyssä grafiikassa kannattaa rikkoa tyyliä ja yrittää jakaa tehosteita yksinkertaisiin muotoihin [10].

Jos tehosteet ovat liian hienoja, niin ne voivat viedä liikaa huomiota pois itse pelistä. Sama pätee, jos peli on todella realistinen, mutta tehosteet on tehty maalatulla tyylillä. Aina kannattaa puhua tiimin tyylivastaavalle tehosteen ulkonäöstä, sillä hänellä usein on paras näkemys siitä, miltä pelin pitäisi lopussa näyttää. Jos kyseessä on pieni tiimi, niin tietenkin voi kaikilta artisteilta kysyä sitä myös, tai pyytää heiltä apua suunnittelussa.

Tehosteen täytyy myös vastata pelissä tapahtuvia asioita. Keraamisen maljakon särkymisessä ei kannata käyttää isoa räjähdystä tehosteena, ellei sillä yritetä saada pelaaja nauramaan.

3.5 Tehosteen optimointi

Monet aloittelijat usein unohtavat kokonaan optimoinnin ja vain lisäävät partikkeleja tehosteisiin, kunnes koko peli ei toimi. On suositeltavaa aina varmistaa ohjelmoijilta, kuinka vaativia tehosteita

voidaan käyttää tietokoneen tehoon verraten. Jos kyseessä on yksinkertainen mobiilipeli, niin optimointi tulee entistä tärkeämmäksi. Myös jos kyseessä on toimintapeli, missä on paljon pelaajia, niin tehosteet kannattaa suunnitella pelaajamäärän mukaan, koska näytöllä tulee olemaan paljon tehosteita samaan aikaan. Optimointiin on monia eri käytäntöjä, esimerkiksi kaikkien tekstuurien laittaminen samalle atlakselle, eli isolle kuvalle, josta voidaan rajata tietty pienempi kuva. Tällöin pelimoottorin tarvitsee ladata vain yksi kuva muistiin. Myös iso apu on, ettei käytä rajattomasti partikkeleja, vaan vähentää niitä, jos teho alkaa kärsimään. Myöskään ei kannata käyttää valtavaa määrää läpinäkyviä partikkeleita päällekkäin. Tällöin pelimoottori joutuu tekemään enemmän laskuja, jotta läpinäkyvyys toimisi oikein, sillä sen täytyy laskea erikseen, mitä jokaisen partikkelin takana näkyy [11].

3.6 Jälkikäsittely

Jälkikäsittely eli Post-Processing on pelimoottorissa kameraan lisättävä tehoste, jolla voi lisätä erilaisia tehosteita. Monet pelit käyttävät jälkikäsittelyä lisäämään esimerkiksi valoihin tai kirkkaisiin pintoihin enemmän hohtoa, taustaobjekteihin sumennusta tai näytön reunoille tummennusta.

Jälkikäsittely on erittäin helppo tapa saada peli näyttämään paremmalta, jos sen tekee oikein. Monet kuitenkin laittavat tehosteiden arvot aivan liian korkealle, jolloin peli alkaa näyttämään sekavalta, koska aivan kaikki hohtaa, välkkyy tai sumenee. Täytyy löytää täydellinen tasapaino tehosteille, jotta ne eivät pistä silmään, mutta silti tuovat silmänruokaa peliin.

Unity-pelimoottorissa helppo tapa lisätä näitä tehosteita on asentamalla Post-Processing Stack projektiin. Tämä on Unityn omatekemä lisäosa, joka voidaan liittää kameraan. Siihen voidaan sitten luoda profiili, josta on helppo lisätä eri tehosteita ja säätää niiden arvoja.

Näitä tehosteita pystyy myös koodilla säätämään pelin aikana, eli esimerkiksi kun pelaajahahmo ottaa vahinkoa, niin jälkikäsittelyllä voi laittaa näytön reunat välähtämään punaisella ja pelin sumenemaan hieman.

4 Tehosteen tekeminen

Melkein kaikilla tehostegraafikoilla on omat tavat tehdä tehosteita, mutta tässä osiossa koitetaan käydä mahdollisimman laajasti läpi eri työkaluja ja työtapoja, jotta niistä voi valita työtehtävään sopivat vaihtoehdot.

Tehosteita pystyy tekemään käyttämällä vain yhtä työkalua, mutta laaja tietämys ja eri työkalujen hallinta auttaa monipuolisten ja uniikkien tehosteiden luomisessa. Monet aloittelijat käyttävätkin alussa pelkkää partikkelisysteemiä, mutta tämä on paljon rajoittavampaa kuin eri työtapojen yhdistäminen.

4.1 Työkalut

Pelimoottori vaikuttaa paljon työtapoihin. Yleisimmät pelimoottorit ovat Unity ja Unreal Engine 4, mutta on paljon muitakin pienempiä tai yksityisiä pelimoottoreita. Valtaosa esimerkeistä on Unity-pelimoottorista, mutta samat käytännöt usein pätevät muillakin pelimoottoreilla.

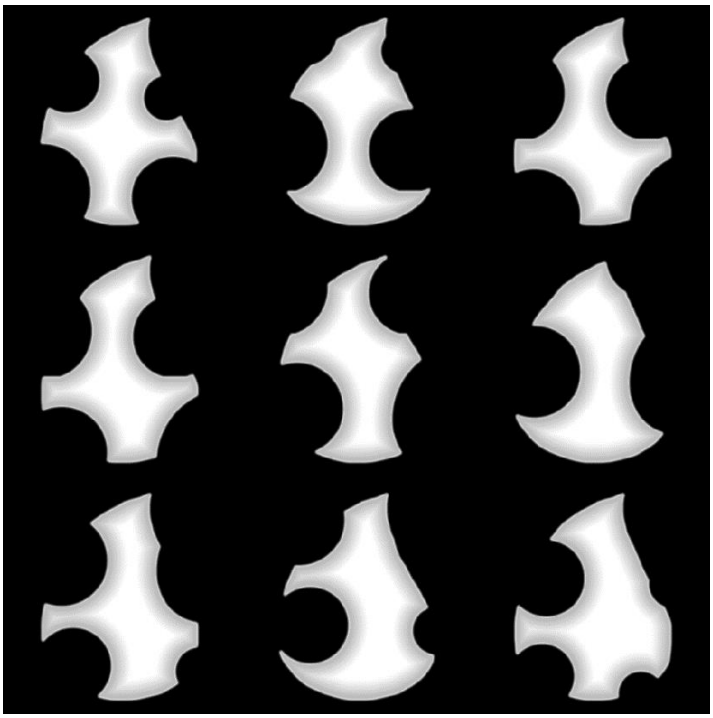
Pelimoottorin lisäksi tehosteissa voidaan käyttää monia eri työkaluja. Tehosteita voi tietenkin tehdä käyttäen ainoastaan pelimoottoria, mutta parhaan tuloksen saavuttamiseksi ulkoiset työkalut ovat hyödyllisiä. Ei ole vain yhtä oikeaa sovellusta tehosteiden tekemiseen, sillä kaikilla graafikoilla on omat työkalut, joihin he ovat tottuneet. Ulkoiset sovellukset voivat olla kevyitä ja ilmaisia piirustusohjelmia tekstuureiden tekemiseen tai ammattikäytöstä löytyviä monimutkaisia ja kalliita proseduraaliseen tekemiseen käytettäviä sovelluksia, jotka luovat esimerkiksi räjähdyksiä koodatulla simulaatiolla.

Yleisin käyttötarkoitus ulkoisille sovelluksille on tekstuurien luominen. Tekstuurien piirtämiselle suosituin sovellus on Adobe Photoshop, mutta se on maksullinen. Hyvä ilmainen vaihtoehto on esimerkiksi Krita niminen piirustusohjelma. Tehosteisiin tarvittavat tekstuurit eivät kuitenkaan ole usein monimutkaisia, joten graafikon kannattaa vain käyttää sovellusta, joka on hänelle tutuin. Sprite-arkkien tekeminen on kuitenkin paljon haastavampaa, joten niihin usein käytetään sovellusta, joka on luotu animoimiseen. Suosittuja vaihtoehtoja ovat Adobe After Effects ja Animate.

Ulkoisia sovelluksia käytetään myös proseduraaliseen simulaatioon. Tämä kuitenkin on erittäin raskas ja haastava prosessi, joten tämä on usein varattu vain huippurealistisiin tehosteisiin ja vaatii kalliita sovelluksia, kuten Houdini-sovelluksen, jota käytetään myös elokuvien tekemiseen.

4.2 Sprite-arkki

Sprite-arkki on kuva, johon on pakattu paljon pienempiä kuvia tasaiseen ruudukkoon tehosteen eri vaiheista (kuva 5). Näitä usein käytetään 2D-tehosteissa tai todella tyylitellyissä tehosteissa. Pelimoottori lukee kuvan vasemmalta oikealle, ylhäältä alas, ja näyttää kuvat yksitellen graafikon asettamalla nopeudella. Sprite-arkkien hyvä puoli on se, että ne antavat täyden artistisen vapauden ja hallinnan tehosteen ulkonäöstä, mutta niiden tekemiseen usein menee paljon aikaa.



Kuva 5. Sprite-arkki

Sprite-arkkeja voidaan tehdä monella eri tavalla, mutta niitä tehdessä täytyy aina muistaa tietyt asiat. Kuvan tulisi aina olla neliö, jonka koko on jaollinen neljällä. Tämä auttaa muistinhallinnassa, sillä kuva on helpompi kompressoida. Kuvien täytyy olla siinä järjestyksessä kuin niiden halutaan näkyvän pelissä, alkaen ylävasemmalta ja loppuen alaoikealle. On myös suositeltavaa, että sprite-arkissa on joko läpinäkyvä tai musta tausta. Jos kaikkien tekstuurien taustat ovat mustia, niin varjostimen voi laittaa etsimään sen värin ja olla näyttämättä sitä.

Sprite-arkin kuvissa voidaan käyttää värejä, ja niin kannattaakin tehdä, jos halutaan juuri tietynlaiset värit tehostein. Jos kyseessä on tehoste, josta tarvitsee tehdä eri versioita esimerkiksi vaihtamalla väriä, niin silloin kuvat kannattaa tehdä mustavalkoisena. Tällöin tehosteen väriä voidaan säätää helposti pelimoottorin sisällä. Vaikka tekee värillisen version, niin on myös suositeltavaa tehdä mustavalkoversio sprite-arkkista, jotta sitä voidaan käyttää emission eli tehosteen kirkkauden säätämiseen.

Sprite-arkkien tekemiseen löytyy monia eri työkaluja, mutta yleisimpiä ovat Photoshop, After Effects ja Flash. Photoshop on yleisesti käytetty, jos haluaa itse piirtää kaikki spritet. Tällä saa tehtyä juuri sellaista jälkeä kuin haluaa, mutta jokaisen kuvan piirtämiseen menee paljon aikaa.

Photoshopissa tehosteen kaikki eri vaiheet kannattaa tehdä erilliselle tasolle. Hyvä apu kuvien piirtämisessä on se, että piilottamisen sijaan tekee edellisestä kuvasta läpinäkyvän ja piirtää päälle. Tämä auttaa paljon yhtenäisyyden tuomisessa tehostein. Photoshopissa ei ole valmiina mitään tapaa tehdä kuvista sprite-arkkeja, mutta internetistä voi ladata koodin, joka tekee tämän. Internetistä myös löytyy palveluita, joihin voi ladata kaikki kuvat yksittäin ja yhdistää ne sprite-arkiksi. Nämä palvelut usein kuitenkin kompressoivat kuvia, jolloin kuvan laatu kärsii, joten tämä ei ole suositeltavaa.

Adobe After Effects on myös hyvä sprite-arkkien luontiin, mutta täysin erilainen Photoshopista. After Effectsillä pystyy luomaan tehosteita proseduraalisesti yhdistämällä eri osia. Esimerkiksi animoitu savupartikkeli voidaan tehdä käyttämällä generoitua kuvaa pilvistä, laittamalla se liikkumaan ja käyttämällä maskia poistamaan osat, joita ei haluta. Tämä on erittäin nopea tapa tehdä sprite-arkkeja, jotka liikkuvat sulavasti. Jos kyseessä on ei niin tärkeä taustatehoste tai pieni osa isompaa tehostetta, niin tämä on hyvä tapa tehdä se. Vaikka tehoste ei olisi ihan niin hyvä kuin mitä saisi tehtyä muilla keinoin, niin säästää paljon aikaa, joka voidaan käyttää tärkeiden tehosteen hiomiseen.

4.3 Partikkelit

Partikkeli on usein neliön muotoinen 3D-objekti, joka usein tehdään osittain läpinäkyväksi materiaalilla ja laitetaan osoittamaan aina kameraa päin. Partikkelina voi kuitenkin käyttää esimerkiksi muitakin 3D-objekteja, esim. aikaisemmin mallinnettuja kiviä. Suuntaakin voi muuttaa, jos sen ei halua aina osoittavan kameraa päin. Partikkeleiden käyttäytymistä usein säädetään pelimoottorin

sisällä, joten työtavat saattavat olla täysin erilaiset eri pelimoottoreiden välillä. Tämä ei kuitenkaan tarkoita, että tarvitsee kaikkea opetella alusta alkaen, jos joutuu vaihtamaan pelimoottoria. Samat periaatteet ja käytännöt pätevät aina partikkelisysteemejä tehdessä, vaikka kaikki asetukset, joilla niitä säädetään, olisivatkin eri paikoissa.

Partikkelit ovat todella monipuolisia, ja niitä voidaan käyttää melkeinpä kaikissa tehosteissa. Tämä ei kuitenkaan tarkoita, että niitä täytyisi käyttää aina tai että kaikki pelin tehosteet koostuisivat pelkästään partikkeleista. Esimerkiksi jos tekee miekaniskutehosteen käyttämällä 3D-mallia, niin iskun jälkeen voi laittaa muutaman partikkelin ilmaan leijumaan vähäksi aikaa visuaalisen kiinnostavuuden lisäämiseksi. Partikkeleissa voidaan tehdä pölyä lentämään valossa, kiviä lentämään kovan iskun osuessa maahan, taikaohjuksia tai melkein mitä vain.

Partikkeliin lisätään haluttu materiaali. 3D-objekteja käyttävät partikkelit voivat käyttää normaaleja materiaaleja, mutta yleisesti ottaen partikkeleissa käytetään niille tehtyjä erikoisia materiaaleja. Nämä ovat usein läpinäkyviä ja niihin lisätään haluttu tekstuuri tai jopa sprite-arkki. Käytettävät tekstuurit ovat usein mustavalkoisia ja niiden väriä, kokoa ja kirkkautta säädellään partikkelisysteemin kautta, mutta jos on varma, ettei halua väriä vaihtaa, niin sen voi lisätä jo itse tekstuuriin.

Partikkeleilla voi todella helposti tuoda monipuolisuutta tehosteisiin, sillä partikkeleiden käyttäytyminen on paljon satunnaisempaa kuin esimerkiksi animoitu 3D-objekti, ellei partikkelisysteemissä ole käytetty erillistä koodia tai asetuksia, jotka ohjaavat kaikkia partikkeleja.

4.4 Varjostimet

Varjostimet ovat monien mielestä vaikein asia oppia hyvin tehosteen teossa, ja monet eivät koskaan edes opettele itse niitä tekemään. Varjostin on koodi, joka kertoo näytönohjaimelle, miten valon, varjojen, värin ja muiden asioiden kuuluisi käyttäytyä materiaalissa. Koska kyseessä on erittäin vaativaa koodia, graafikot usein jättävät tämän tiimin ohjelmoijille. Jos vain on mahdollista, niin oma varjostin on usein paras valinta, sillä sen avulla graafikko pystyy tekemään juuri omaan käyttötarkoitukseen sopivat ominaisuudet tehosteeseen.

Tämä kuitenkaan ei aina ole mahdollisuutena, mutta lähivuosina solmupohjaiset varjostintyökalut ovat alkaneet yleistymään. Nämä ovat erittäin graafikkoystävällisiä työkaluja, joilla varjosti-

mien tekeminen on kuin legopalikoilla rakentamista. Jokainen palikka on osa koodia, jotka yhdistetään keskenään vetämällä viivoja palikasta toiseen. Tästä ajan mittaan koostuu iso verkko palikoita, jotka kaikki säätävät tiettyjä asioita materiaalista. Tämä ei välttämättä anna niin paljoa vapautta kuin itse kirjoitettu koodi, mutta moniin tarkoituksiin se on tarpeeksi ja säästää paljon aikaa.

Varjostimella voi myös muokata esimerkiksi 3D-objektin muotoa tai animoida sen. Esimerkiksi jos graafikko mallintaa perhosen, niin vertex offsetin ja maskien avulla perhosen siivet saa liikkumaan kuin se lentäisi. Vertex offset liikuttaa 3D-objektin osia ja maskilla tietyt osat objektista voidaan lukita paikoilleen, tai laittaa varjostin vaikuttamaan heikommin tietyille alueille. Tätä voidaan käyttää nopeiden animaatioiden tekemiseen ja on todella yleistä kasvillisuudessa, esimerkiksi ruohon ja puun lehtien heilumisanimaatiossa.

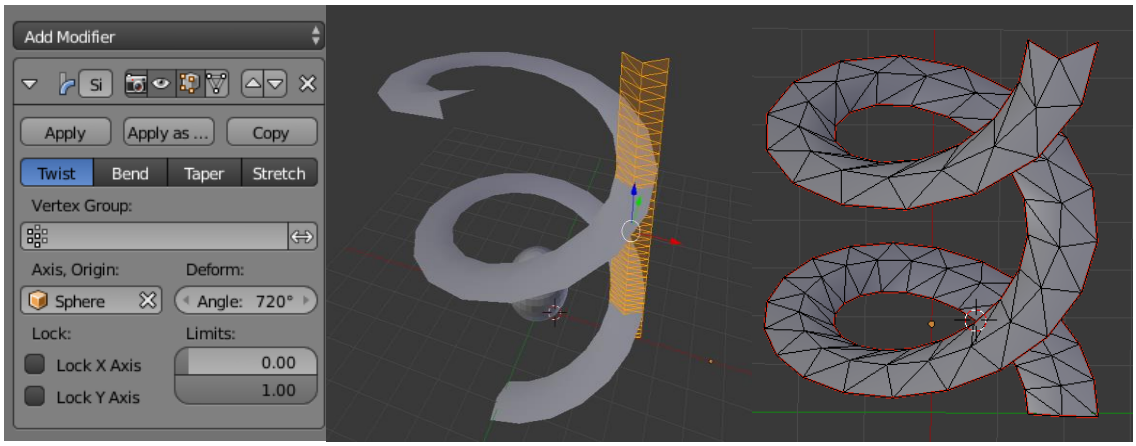
Varjostimella voidaan myös tehdä erittäin uniikkeja tehosteita. Esimerkiksi pelissä Dishonored 2 on erinomainen esimerkki itse koodatusta hyvästä varjostimesta. Pelissä on tehtävä, jossa pelaaja käyttää työkalua, joka näyttää kuvaa eri ajalta (kuva 6). Tässä tehtävässä on kaksi eri kenttää ladattuna samaan aikaan ja pelaaja voi hyppiä niiden välillä vapaasti [12]. Työkalun linsseissä on käytetty varjostinta, joka näyttää kuvaa kentästä, jossa pelaaja ei ole tällä hetkellä. Linssissä on myös normaalikartta, joka vääristää kuvaa hieman ja ohjaa heijastusten valoja.

Kuva 6. Dishonored 2-aikakristallivarjostin [12]

4.5 3D-objektit

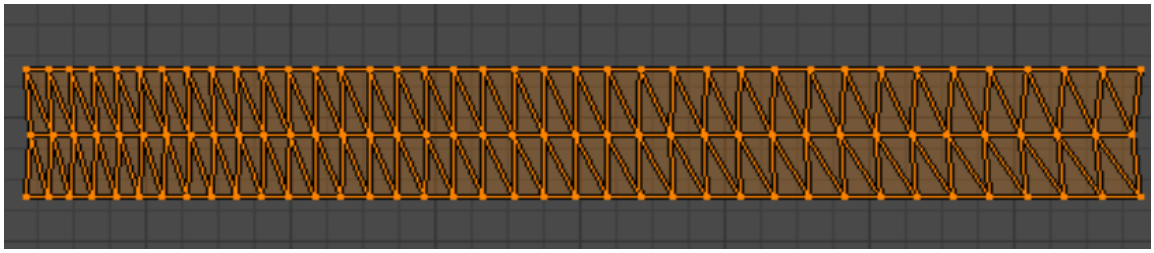
3D-objektit toimivat niin sanotusti tehosteen selkärankana. Jos tehosteen täytyy olla juuri tietyn muotoinen tai mennä tiettyä rataa, niin se on helppo saavuttaa käyttämällä 3D-objektia.

Objektia mallintaessa on tärkeää pitää mielessä käyttötarkoitus ja toimivuus. Koska itse objektia ei yleensä tulla täysin näkemään, niin on tärkeää välttää ylimääräisten polygonien lisäämistä alueisiin, joissa tehoste ei tule näkymään. Kuitenkin jos kyseessä on objekti, jonka geometriaa tullaan muuttamaan varjostimen avulla pelin sisällä, niin niihin alueisiin kannattaa laittaa enemmän polygoneja, jotta tehoste toimisi paremmin. Esimerkiksi jos haluaa, että hahmon ympärillä pyörii partikkeleja, niin pohjana voi käyttää spiraalin muotoista 3D-objektia. Tämä on erittäin nopea ja helppo tapa saada tehoste käyttäytymään juuri halutulla tavalla. 3D-objekteja tehdessä kannattaa hyödyntää mallinnusohjelman eri työkaluja, esimerkkinä kuvassa 7 oleva ”simple deform”, joka pyöritti suoran objektin pallon ympärille säästäen paljon aikaa spiraalin mallintamisen sijaan.



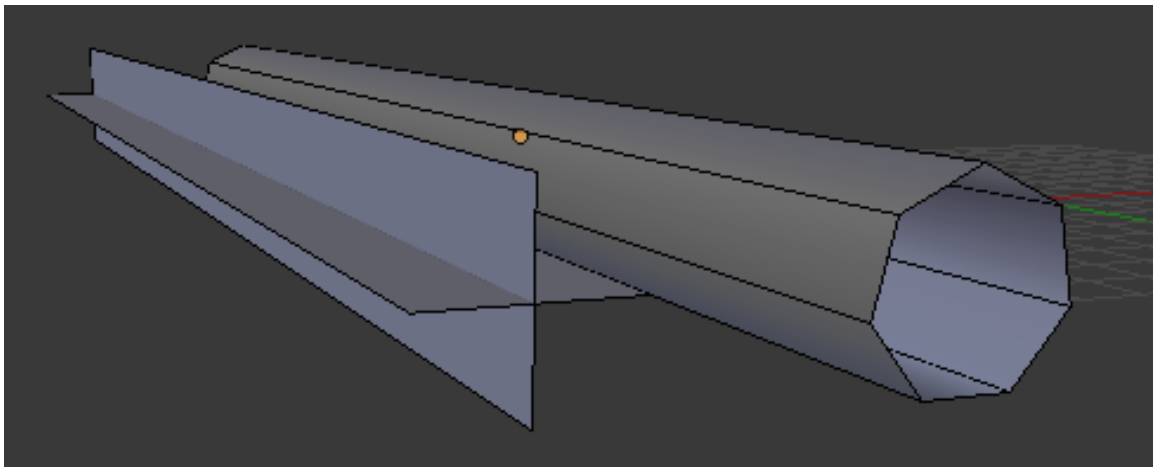
Kuva 7. Spiraalin muotoinen 3D-objekti

UV-kartta kertoo pelimoottorille, missä kohtaa objektia tekstuurien kuuluu olla. Se antaa kaikille vertekseille omat koordinaatit 2D-kuvalla, jotta varjostin osaa laittaa 2D-tekstuurin oikeaan kohtaan 3D-objektissa. Koska 3D-objekteihin lisätään tekstuureja ja tehosteita, niin UV-kartta kannattaa tehdä huolella. Tämä helpottaa objektin käyttämistä huomattavasti. Esimerkkinä käytetyssä spiraalissa UV-kartta on vain suora kaistale eikä spiraali niin kuin itse objekti (kuva 8). Tämän ansiosta tekstuurin voi laittaa liikkumaan, vaikka vasemmalta oikealle, jolloin pelin sisällä tehoste pyörii spiraalin mukana [13].



Kuva 8. UV-kartta

Ei ole vain yhtä oikeaa tapaa tehdä objekti. Esimerkiksi jos haluaa tehdä suoran säteen, niin sen voi mallintaa tekemällä lieriön, liittämällä kaksi levyä yhteen ristikkäin tai vaikka käyttämällä molempia (kuva 9). Aina kannattaa ajatella käyttötarkoitusta, haluttua tulosta ja kuinka paljon tehoa tehoste saa viedä. Kannattaa aina pitää mielessä, että pelin sisällä objekteista tehdään usein läpinäkyviä ja vain pieni osa niistä tulee näkymään kerrallaan. Eli tehon kannalta halpa ristilevyobjekti voi näyttää yhtä hyvältä tai jopa paremmalta kuin vaivalla tehty kiemurteleva lieriö.



Kuva 9. 3D-objekteja

4.6 Tekstuurit

Tekstuurit tuovat tehosteeseen halutun ulkonäön. Tekstuuri on 2D-kuva, joka voidaan liittää partikkeleihin ja 3D-objekteihin. Tekstuureita myös käytetään tehosteiden ns. näkymättömissä osissa, esimerkiksi varjostimessa maskina tai emissiossa, jos ne halutaan juuri tiettyyn kohtaan tietyn muotoisena.

Usein tekstuurit tehdään käyttämällä piirto-ohjelmaa, kuten Adobe Photoshop sovellusta. Tällä pystyy maalaamaan 2D-kuvan juuri tarpeidesi mukaisesti minimaalisilla rajoituksilla. Aivan kuin sprite-arkkien, tekstuurien täytyy kuitenkin olla neliön muotoisia ja neljällä jaettavalla koolla,

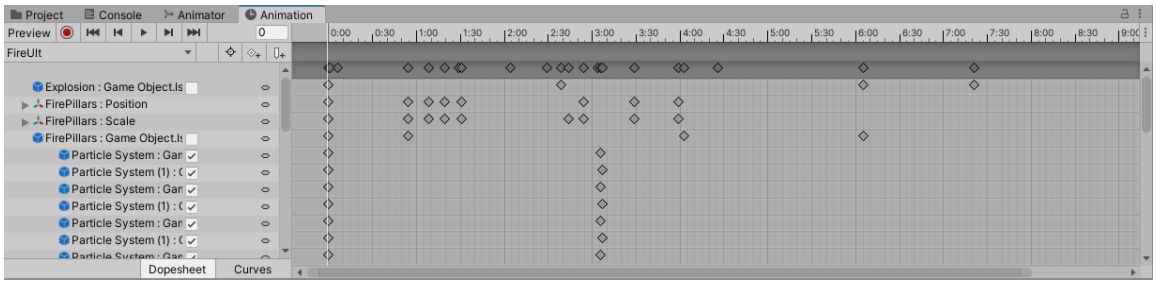
jotta pelimoottori voi ne helpommin kompressoida säästäen muistia. Tehosteissa käytettäviä tekstuureita ei kannata tehdä kovin isolla koolla. Esimerkiksi pienissä partikkeleissa käytettävä tekstuuri voi olla vaikka 32x32 pikselin kokoinen, sillä pienessä liikkuvassa partikkelissa ei helposti näe yksityiskohtia. Olisi siis täysin turhaa tuhlata tehoja niiden lisäämiseen, jos kukaan ei niitä näe.

Yksi mahdollisuus on myös tehdä tekstuurit proseduraalisesti esimerkiksi Substance Designer sovelluksella yhdistämällä eri solmuja. Tämä on nopea tapa tehdä erilaisia yksinkertaisia muotoja, jos niille on tarvetta. Niitä on myös helppo muuttaa tarpeen tullen muokkaamalla solmujen arvoja, jolloin koko kuva muuttuu [14].

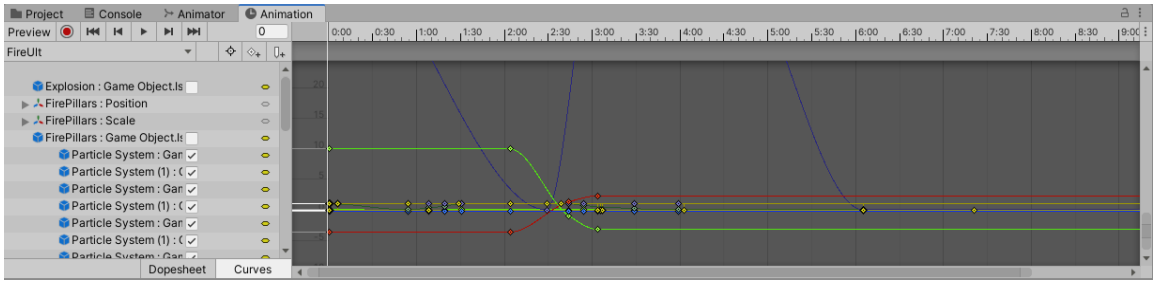
Jos tekstuurista haluaa läpinäkyvän, niin kannattaa käyttää mustaa tai läpinäkyvää taustaa. Tällöin varjostimen voi laittaa poistamaan taustan helpommin. On myös suositeltavaa muokata tekstuurista mustavalkoversio, jota voidaan käyttää emission luomisessa tai maskina.

4.7 Animaatio

Animaatiolla tuodaan kaikki tehosteen osat yhteen ja laitetaan ne toimimaan halutulla tavalla. Animaattorissa voidaan aikajanalle merkitä asioita ja säätää niiden käyttäytymistä animaation koko elinkaaren aikana. Käyttämällä animaattoria tehosteessa voidaan kaikki tehosteen osat ajoittaa toistumaan juuri halutulla ajalla. Animaattorilla voidaan myös säätää kaikkia osia tehosteesta aina objektien muodosta partikkelisysteemien nopeuteen. Yksinkertaisissa tehosteissa, kuten kipinöissä, animaattori ei ole tarpeellinen, mutta jos tehoste koostuu monesta eri osasta, niin sen käyttäminen on suositeltavaa. Tällä voidaan myös hienosäätää ajoituksia helposti, esimerkiksi synkronoimaan pelaajan hyökkäysanimaatio ja hyökkäystehoste. Hienosäätöä voidaan tehdä säätämällä arvoja, liikuttelemalla aikajanalle tehtyjä neliöitä ja niistä luotuja kurveja. Neliöiden liikuttelu on hyvä, jos haluaa, että jokin asia tapahtuu aikaisemmin tai myöhemmin animaatiossa (kuva 10). Kurvilla voidaan säätää, kuinka nopeasti tai pehmeästi muutokset tapahtuvat eri kohtien välillä (kuva 11). Arvoilla säädetään eri asioiden voimakkuutta tai laitetaan niitä pois päältä kokonaan. Esimerkiksi jos animaattoriin kytkee partikkelisysteemin, niin siitä voi luoda kaksi eri vaihetta aikajanalle, millä säädetään onko partikkelisysteemi pällä vai pois päältä. Näiden välillä voi sitten säätää kurvista, jos vaikka haluaa, että partikkelien luonti alkaa hitaasti kiihtymään, eikä vain mene nollatilasta täysille.



Kuva 10. Unity animaattorin Dopesheet-aikajana



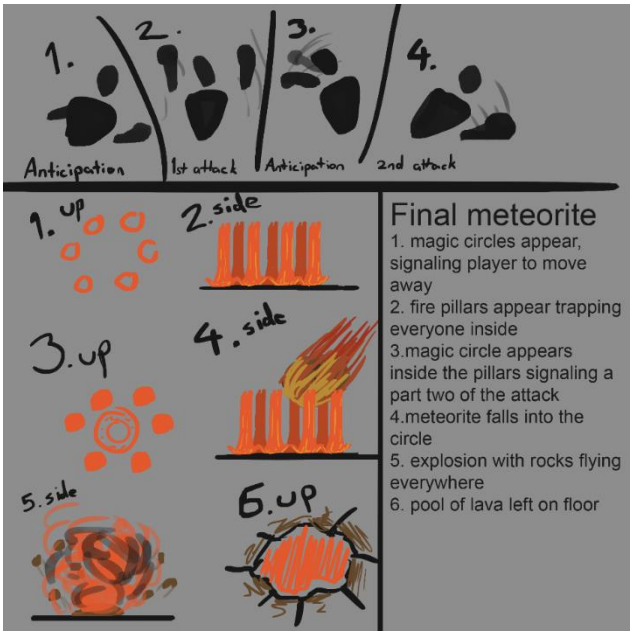
Kuva 11. Unity animaattorin Curves-aikajana

5 Meteoritaikaisku-tehoste

Opinnäytetyössä päätettiin tehdä iso viimeistelyhyökkäys, jotta voisi käyttää kaikkia eri tekniikoita ja työtapoja. Monet graafikot tekevät liekin ensimmäisenä tehosteenaan, joten päätettiin tehdä tuli elementtiin pohjautuva loitsu. Pelkkä liekki kuitenkaan ei olisi tarpeeksi testaamaan eri tekniikoita ja työtapoja, joten siihen täytyi keksiä jotain lisäksi. Lopuksi päädyttiin loitsuun, joka ensin lukitsee vihollisen tulipilareiden sisälle ja sitten tiputtaa meteorin pilareiden muodostamaan ympyrään aiheuttaen ison räjähdyksen. Tässä tehosteessa on niin monta osaa, että eri tekniikoiden kokeileminen olisi helppoa.

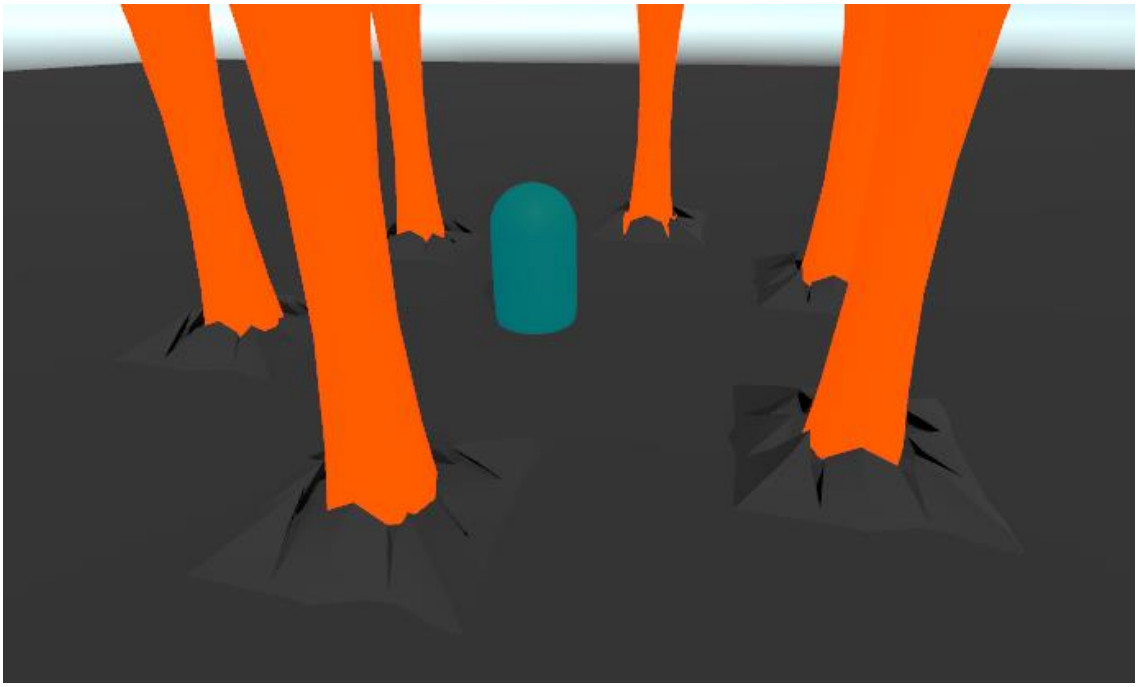
5.1 Alkuvaiheet

Idean valmistuttua lähdettiin suunnittelemaan tehosteen toteutusta. Vaikka oli jo hyvä idea tehosteesta mitä lähtisi tekemään, niin päätettiin aloittaa konseptilla. On hyvä käytäntö aina tehdä monta konseptia, jotta työympäristössä ideat välittyisivät muille ja niistä voisi valita sopivan suunnan. Vaikkei sitä tässä projektissa tarvinnut, niin Adobe Photoshop-sovelluksessa piirrettiin nopea sarja kaikista tehosteen eri vaiheista ja selityksen niille (kuva 12). Tehoste alkaa taikaympyröillä, jotka viestivät pelaajalle hyökkäyksen kattamaa aluetta, jonka jälkeen tulipilarit lukitsevat pelaajan sisälle. Lopulta meteori tippuu alueelle aiheuttaen räjähdyksen.



Kuva 12. Meteori-iskukonsepti

Tehosteesta on hyvä ensin tehdä nopea karkea versio, jotta näkee komposition, ajoitukset ja yleisen toimivuuden. Tämä on tärkeä vaihe suunnittelun kannalta, joten tämä tehdään usein ensimmäiseksi. Karkeaa versiota voi jopa käyttää konseptina, jos ei halua piirtää. Karkean version tavoitteena on vain antaa idea tehosteesta ja testata toimivuutta. Tämän vuoksi siinä ei kannata käyttää tekstuureja, partikkeleja tai mitään ylimääräistä. Unitystä löytyy jo erilaisia 3D-objekteja, joita voi hyödyntää testissä, joten tässä keskityttiin vain monimutkaisempiin 3D-objekteihin eli tulipilareihin. Blender3D-sovelluksessa tehtiin taso, josta rikottiin keskiosan saaden sen näyttämään haljenneelta. Tämän jälkeen laitettiin sisälle lieriö, josta kavennettiin keskiosaa (kuva 13).



Kuva 13. Meteori-iskutesti

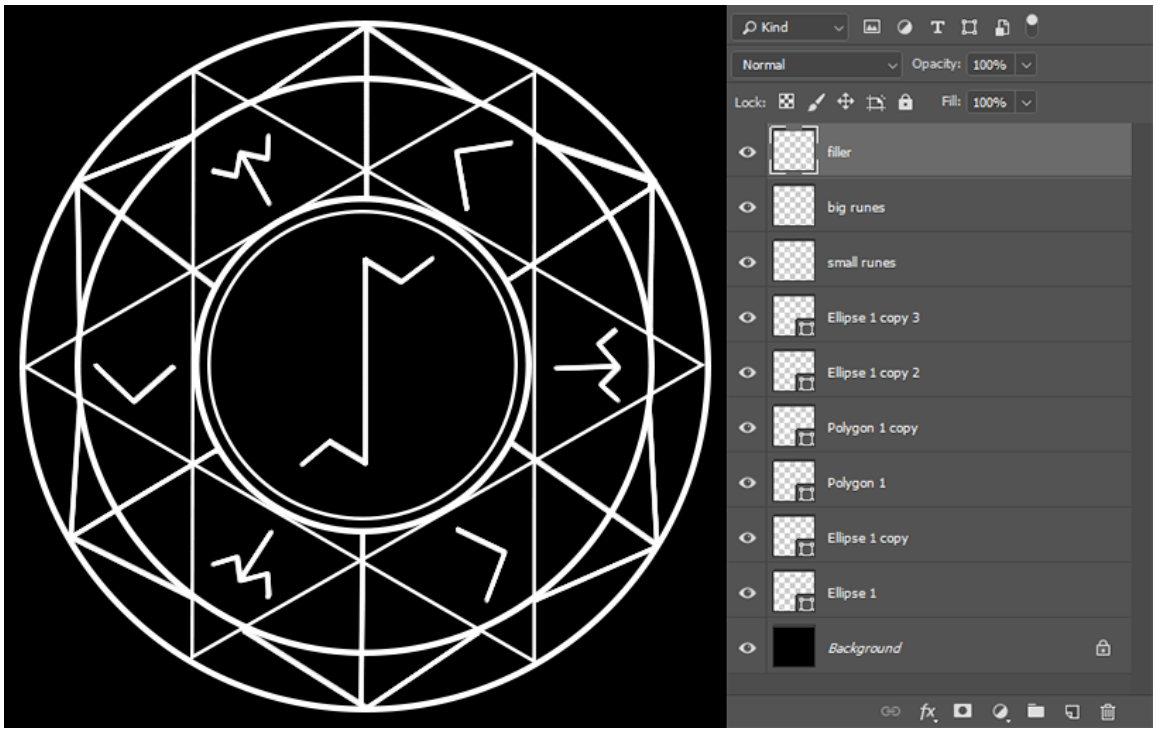
Päätettiin käyttää Unityn High Definition Render-putkea projektissa, koska se on luotu helpottamaan realististen grafiikkojen saavuttamista. HDR-putkessa on suhteellisen paljon eroja normaaliin tai Universal Render putkeen, joten aloitettiin etsimällä ohjeet sen lisäämiseen projektiin ja projektissa käyttämiseen. Unity oli tehnyt kyseisestä aiheesta blogin, joten putken käyttöönotto oli helppoa [15].

Vaikka tekee projektia yksin, niin kannattaa miettiä kaikkia eri vaiheita työympäristössä. Tämä auttaa muistamaan tehosteiden rajoituksia ja tärkeitä vaiheita. Esimerkiksi jo konseptissa on hyvä miettiä hahmojen liikkeitä ja ajoituksia, sillä ne auttavat tehosteen suunnittelussa.

5.1 Eri osien valmistelu

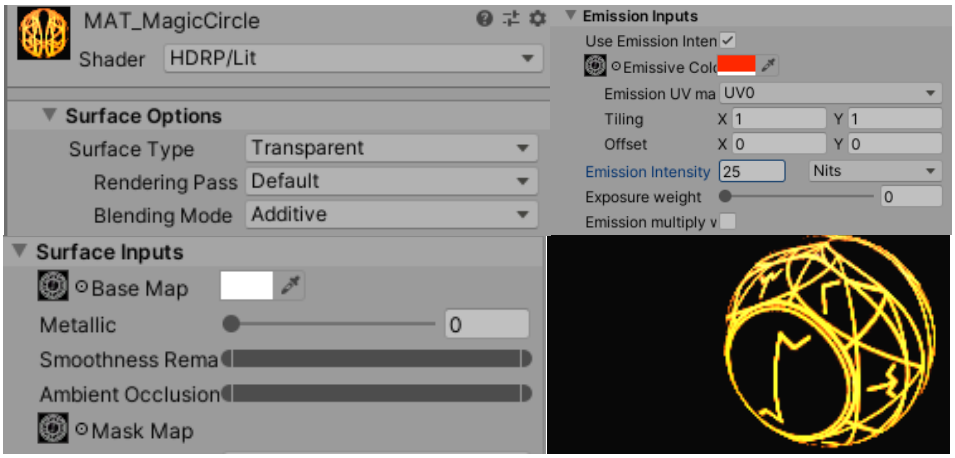
Suunnitteluosion valmistuttua lähdettiin tekemään tehosteen eri osia. Aloitettiin laittamalla Unity-projekti valmiiksi tehosteen testaamista varten, käyttämällä Unityn omaa esimerkkikenttää. Tästä poistettiin aivan kaikki paitsi lattia, koska siinä oli valmiiksi jo hyvä tekstuuri. Kentän valot asetettiin suhteellisen matalalle, jotta lattian tekstuuri näyttäisi entistä paremmalta ja lamppujen sekä taivaan värit muutettiin vaikuttamaan illalta.

Tämän jälkeen oli aika lähteä tekemään ensimmäistä osaa tehosteesta eli taikaympyrää. Referenssinä käytettiin proseduraalista ohjelmaa, joka automaattisesti luo erilaisia taikaympyröitä [16]. Näistä proseduraalisista taikaympyröistä voi joko ottaa inspiraatiota omaan taikaympyrään tai suoraan ottaa ne käyttöön. Päätettiin lisätä taikaan jokin mystinen elementti pelkkien viivojen ja ympyröiden lisäksi, joten lähdettiin tutkimaan pohjoismaisia riimuja ja niiden merkityksiä. Päädyttiin valitsemaan tulta ja maata tarkoittavat riimut, koska ne symboloivat meteoria parhaiten. Tämän jälkeen Photoshopissa piirrettiin ympyröitä ja täytettiin isot tyhjät alueet viivoilla, lopuksi lisättiin riimut tyhjiin kohtiin (kuva 14).



Kuva 14. Taikaympyrätekstuuri

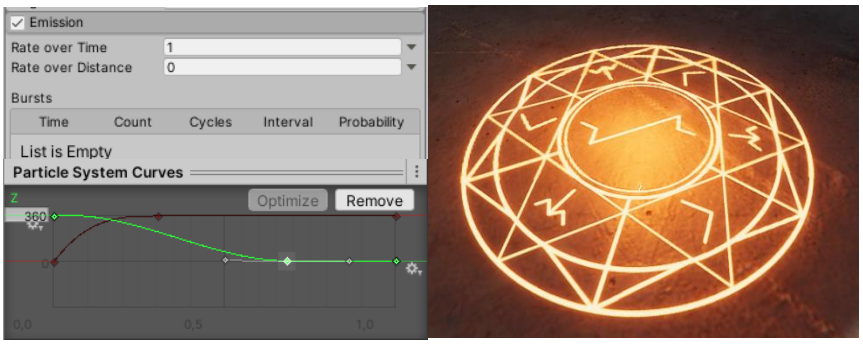
Tekstuurien valmistuttua siirryttiin Unityyn. Haluttiin saada taikaympyrä ilmestymään kentälle kasvavalla pyörivällä liikkeellä, joten päädyttiin käyttämään partikkelisysteemiä (kuva 15). Tässä kohtaa kuitenkin ilmeni ensimmäinen ongelma. HDR-putki ei tue normaaleja partikkelimateriaaleja, joten täytyi testata eri materiaaleja ja löytää juuri oikea materiaali tähän tarkoitukseen.



Kuva 15. Taikaympyrämateriaali Unityssä

Tämä kuitenkin toi pieniä rajoituksia, kuten partikkelisysteemin läpinäkyvyys ja väri lakkasivat toimimasta. Tämä kuitenkaan ei ollut ongelma, koska pystyttiin hieman muuttamaan konseptia ja vaihtamaan arvoja kesken animaation tarvittaessa.

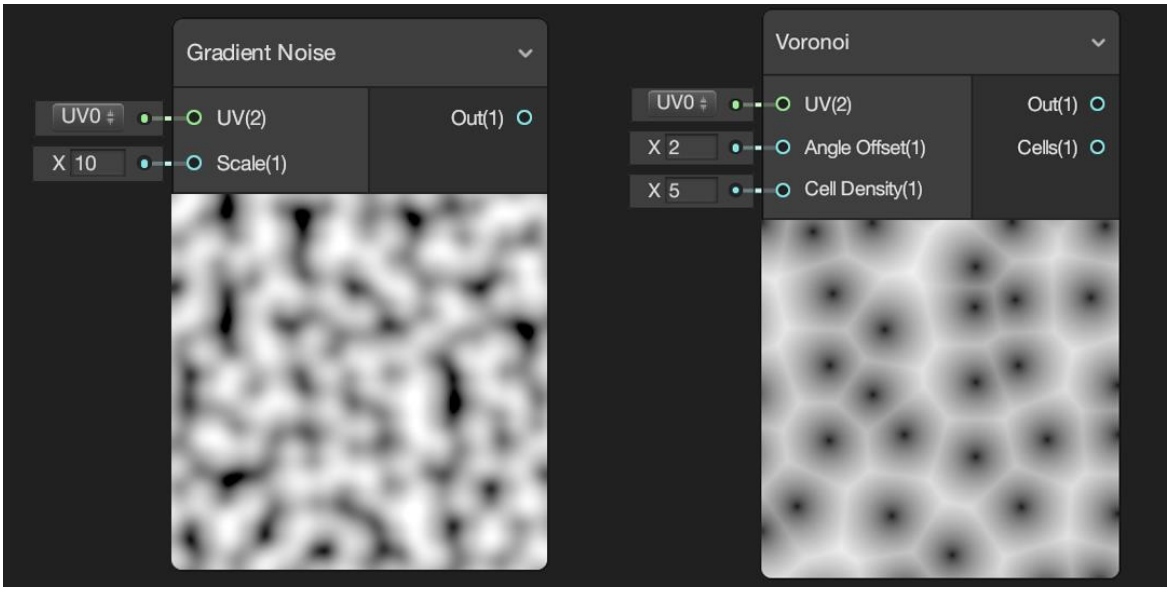
Tavoitteena on saada taikaympyrä ilmestymään visuaalisesti kiinnostavalla ja luonnollisella tavalla. Helpoiten tämän voi saavuttaa tekemällä partikkelisysteemin ja laittamalla sen luomaan yhden partikkelin, jossa on taikaympyrämateriaali. Ei kuitenkaan olisi visuaalisesti miellyttävää, jos se vain ilmestyisi ilman animaatiota. Helppo tapa korjata tämä on muuttamalla partikkelin koko muuttumaan asetetun kurvin mukaisesti, eli alkaa pienenä ja kasvaa täysikokoiseksi. Laittamalla ympyrän pyörimään kasvamisen aikana lisää visuaalista mielenkiintoa vielä enemmän. Lopuksi lisäsin taikaympyräobjektiin lampun, jotta se näyttäisi hohtavan (kuva 16).



Kuva 16. Taikaympyrä ja partikkelisysteemin asetukset

Tehosteen seuraava osa oli tulipilarit, jotka ilmestyvät maasta taikaympyröiden jälkeen. Hyvän tulianimaation tekeminen vie paljon aikaa, joten päädyttiin proseduraaliseen versioon. Proseduraalinen versio luo koodin avulla kuvia ja laittaa ne käyttämään halutulla tavalla. Niihin voidaan myös lisätä itse tekemiä kuvia pohjaksi, jolloin koodi vääristää niitä halutulla tavalla saaden kuvan näyttämään animoidulta.

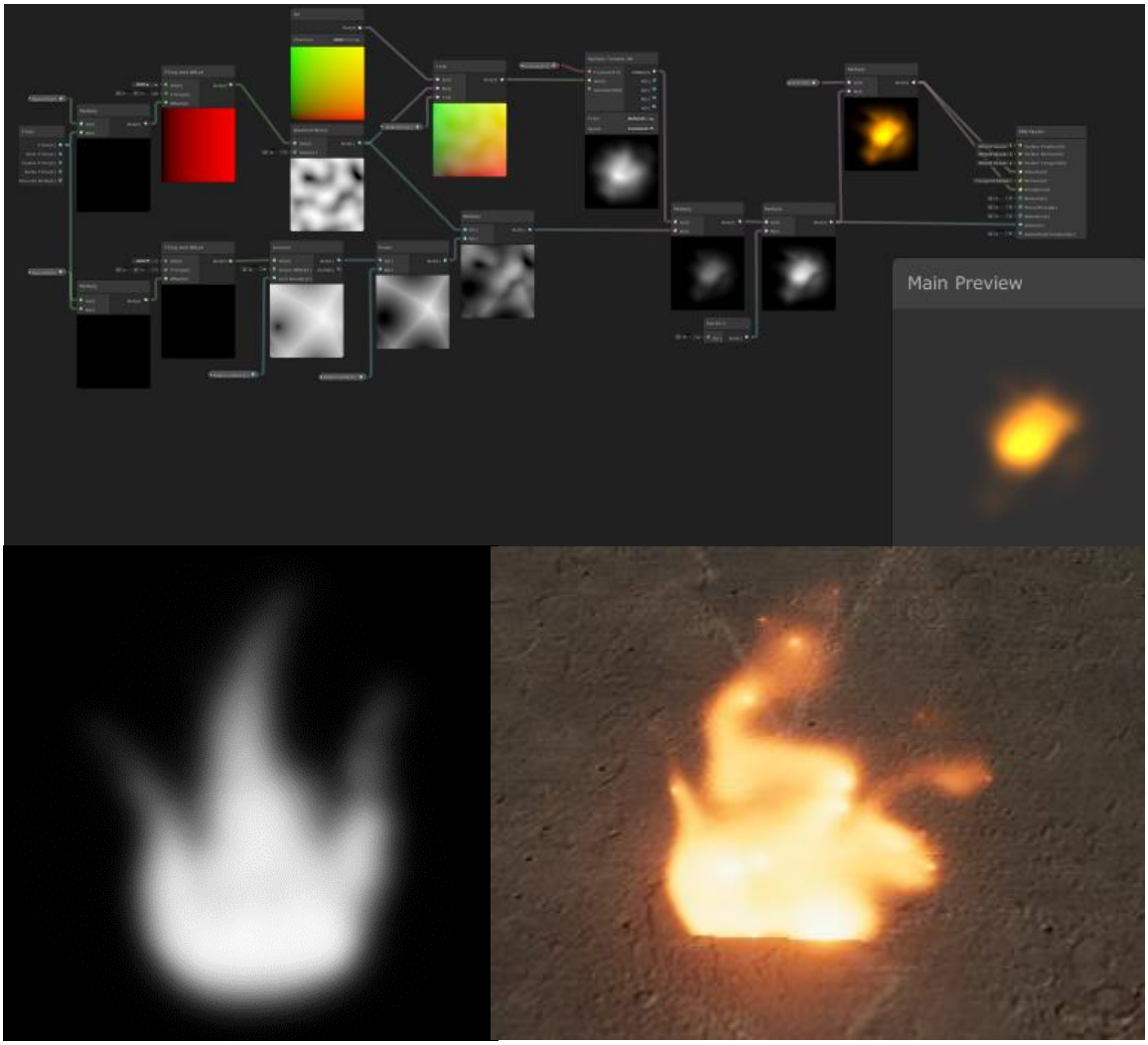
Unityn HDRP-tilassa on mukana Shader Graph-sovellus, jolla voi tehdä varjostimia ilman koodia. Tätä hyödyntämällä lähdettiin rakentamaan proseduraalista liekkiä. Unityn dokumentaatiosta löytyi osa, joka selitti, mitä jokainen solmu tekee [17]. Liekin pohjana toimivat Voronoi- ja Gradient Noise tekstuurit, jotka sai luotua proseduraalisesti (kuva 17).



Kuva 17. Gradient Noise ja Voronoi proseduraaliset tekstuurisolmut [18]

Käyttämällä Time And Offset solmua kytkettynä tekstuurien UV-karttaan, ne voi laittaa liikkumaan eri suuntiin. Laittamalla ne liikkumaan eri suuntiin ja yhdistämällä ne Multiply-solmulla saadaan aikaan yksi kuva, joka muuttaa muotoaan koko ajan. Liikettä voidaan säätää muokkaamalla tekstuurien kokoa ja Time And Offset solmujen nopeutta. Animaatio on nyt valmis, mutta varjostimeen tarvitaan vielä muoto ja väri. Photoshopissa tehdyllä mustavalkoisella kuvalla voi piilottaa halutut alueet käyttämällä sitä maskina. Tämä muuttaa varjostimen neliöstä liekin muotoiseksi. Lopuksi liekille annetaan väri Color-solmun avulla. HDR asetuksella liekki saadaan hohtamaan pelin sisällä. Tämän jälkeen liekki on valmis, mutta sille täytyy vielä asettaa materiaaliarvot, jotta se toimisi oikein pelin sisällä. Laittamalla materiaalin läpinäkyväksi ja yhdistämällä maskin läpinäkyvyyttä säätävään osioon pelimoottori tietää, että sen tulee näyttää vain liekki pelin sisällä (kuva 18).

Proseduraalinen liekki on nyt valmis ja löydettävissä Unityn materiaalit valikosta. Liekin voi lisätä kaikkiin 3D-objekteihin, jolloin ne muuttuvat objektin muotoiseksi täysin animoiduksi liekiksi.



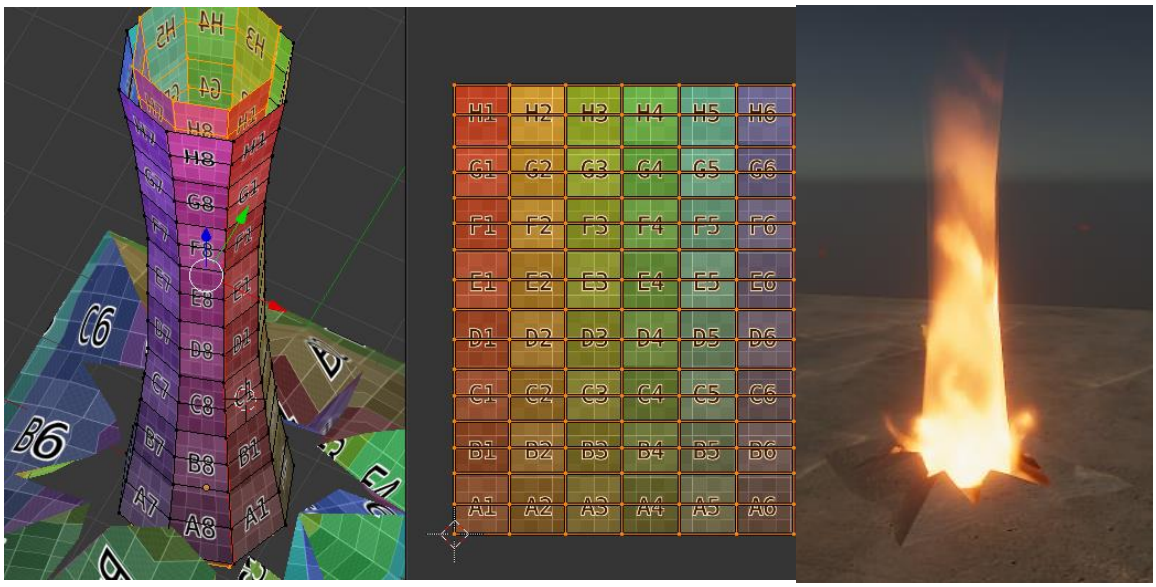
Kuva 18. Shader Graph, Maskina käytetty tekstuuri ja valmis proseduraalinen liekki

Proseduraalinen liekkivarjostin on valmis, mutta se tarvitsee 3D-objektin toimiakseen. Tässä hyödynnettiin jo aikaisemmin mallinnettu tulipilari, jota käytettiin karkeassa testissä. Tämä versio ei ole tarpeeksi hyvä, mutta toimii hyvänä pohjana objektille. Blender-sovelluksessa lähdettiin muokkaamaan pilaria käyttöön sopivaksi. Pilari oli muodoltaan hyvä, joten sille tarvitsi vain lisätä UV-kartta. UV-kartta täytyy kuitenkin tehdä huolella, koska jos siihen jää rakoja tai vääristyneitä kohtia, niin animoidut materiaalit, kuten liekkivarjostin, eivät toimi oikein. Monistamalla pilarin ja laittamalla sen pienempänä alkuperäisen sisälle tehosteesta saadaan enemmän kolmiulotteisen näköinen.

Alkuperäisessä testipilarissa oli myös mukana pala haljennutta maata. Se oli huonosti tehty, joten se siistittiin ja lisättiin enemmän kolmiulotteisuutta haljenneisiin paloihin. Koska maahan tulee

yksinkertainen tekstuuri, joka ei liiku, niin siihen pystyi tekemään nopean UV-kartan. Ylöspäin osoittavat alueet merkittiin, jotta tekstuuri toimisi saumattomasti. Muilla alueilla ei ollut paljoa väliä, koska pelaaja ei niitä tulisi näkemään, joten ne sai valmiiksi automaatiolla. Lopuksi lisättiin kaikille kolmelle objektille oman materiaalin, jotta niihin voisi lisätä oman materiaalin pelimoottorissa.

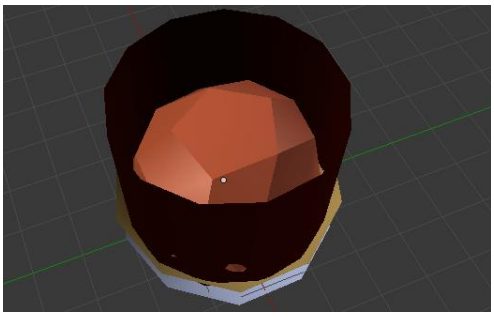
Unityssä lisättiin pilariin tulivarjostin ja toiseen pilariin sille tehty tekstuuri. Varjostimesta laitettiin molemmat pyörimään ja maskilla häivytettiin pilarin yläosa tehden siitä siistimmän näköisen. Pilari ei ollut tarpeeksi hyvän näköinen vielä, joten alas lisättiin neliö, johon oli kiinnitettynä tulivarjostin. Tämä sai haljenneen maan sisäosan näyttämään siltä kuin siellä olisi liikkuvaa laavaa. Lopuksi peitettiin kohta jossa objektit yhdistyvät luomalla partikkelisysteemin, jossa oli liekkivarjostin ja kipinöitä. Tämä toi myös lisää visuaalista mielenkiintoa saaden maan vaikuttamaan kuumemmalta (kuva 19).



Kuva 19. Blender UV-map ja Unityn sisällä oleva valmis pilari

Tehosteen toinen osa on meteori, joka iskeytyy maahan aiheuttaen räjähdyksen. Meteorin käyttäytyminen täytyy ottaa huomioon jo ennen tekemistä. Se tippuu nopealla vauhdilla maahan avaruudesta, eli todennäköisesti se on tulessa. Meteorin nopea vauhti myös tarkoittaa, ettei yksityiskohtiin kannata käyttää paljoa aikaa, koska pelaaja ei ehdi näkemään niitä.

Blender-ohjelmassa oleva Ico-Sphere-objekti on epätasainen pallo, joten siitä on nopea muokata meteori. Joistakin kohdista poistettiin yksityiskohtia ja hajotettiin ympyrämuotoa. Koska meteorin tekstuuria ei pysty kunnolla näkemään, UV-kartta ei ollut kovin tärkeä. Meteorin liekeissä hyödynnettiin taas tulivarjostinta, eli tarvittiin niille erilliset objektit kiinni meteoriin. Meteorin alaosa kopioitiin kahdesti ja niiden kokoa muutettiin suuremmaksi. Tämän avulla saa nopean objektin, mikä menee juuri meteorin muotojen mukaisesti. Nämä kuitenkin vain kattavat meteorin alaosan, joten tarvittiin sylinteri, joka menee meteorin ylitse luomaan liekit sen ympärille ja taakse (kuva 20). Koska kyseessä on erittäin yksinkertaisia muotoja ilman kulmia, UV-Mapit sai tehtyä automaattisesti.


Kuva 20. Meteori 3D-objekti Blenderissä

Unityssä hyödynnettiin meteorissa jo valmiita liekkivarjostimia pienillä muutoksilla. Alimmainen ja pienin liekki oli sininen, saaden meteorin näyttämään kiinnostavammalta ja kuumemmalta. Meteorin takana olevassa liekissä hyödynnettiin pilareiden materiaalia, koska molemmat objektit ovat sylintereitä. Koska tekstuuria ei tulla näkemään kunnolla, niin meteorin tekstuurina käytettiin tummennettua maatekstuuria. Materiaalin normaalikartta, joka säätää valon käyttäytymistä objektiin osuessa, oli myös paljon korkeampiarvoinen, jotta valon osuessa meteoriin epätasaisuudet korostuisivat. Meteorin liekit näyttivät liian tasaiselta, joten pintaa rikottiin käyttämällä partikkelisysteemiä, joka loi meteorin ympärille partikkeleja liekkivarjostimella (kuva 21).

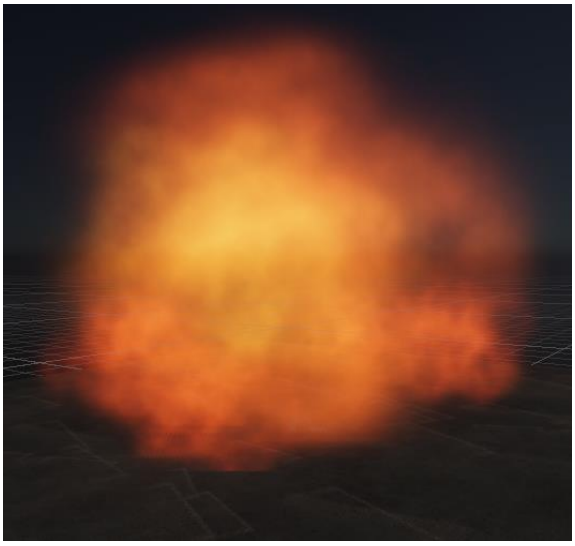

Kuva 21. Valmis meteori Unityssä

Meteori on nyt valmis, mutta sen iskeytymiseen maahan tarvitaan vielä tehoste. Koska meteori tippuu kovalla vauhdilla avaruudesta, niin päädyttiin räjähdystehosteeseen. Tämä on näyttävä ja voimakas tapa lopettaa tehoste. Räjähdys myös viestii pelaajalle ja katsojalle iskun voimakkuuden. Päädyttiin tekemään räjähdys hyödyntäen Unityn partikkelisysteemiä. Photoshop sovelluksessa nopein tapa tehdä tekstuuri räjähdykselle on käyttää ”render cloud” ominaisuutta, joka proseduraalisesti luo pilvitekstuurin. Poistamalla reunat tästä saadaan nopeasti aikaan hyvä pilvitekstuuri räjähdystä varten.

Räjähdys koostuu kahdesta isosta pilviräjähdyksestä, pilvirenkaasta ja räjähdyksen jälkeen jäävästä savusta (kuva 22).

Kuva 22. Räjähdystehosteen hierarkia Unityssä ja pilvitekstuuri

Isot pilvet ovat eriväriset ja toinen partikkelisysteemi toimii räjähdyksen ytimenä kirkkaammalla värillä. Pilvirengas tuo hieman enemmän eloa ja volyymia räjähdykseen, tehden siitä mielenkiintoisemman. Kaikki partikkelisysteemit paitsi isoin räjähdys käyttävät viittä tai alle partikkelia, kun taas isoimmassa osassa on 25 partikkelia. Kaikissa partikkelisysteemeissä pilvet luodaan erikokoisina ja eri asennossa. Heti luomisen jälkeen ne alkavat pyöriä hieman, tehden räjähdykseen paljon enemmän pientä liikettä (kuva 23).



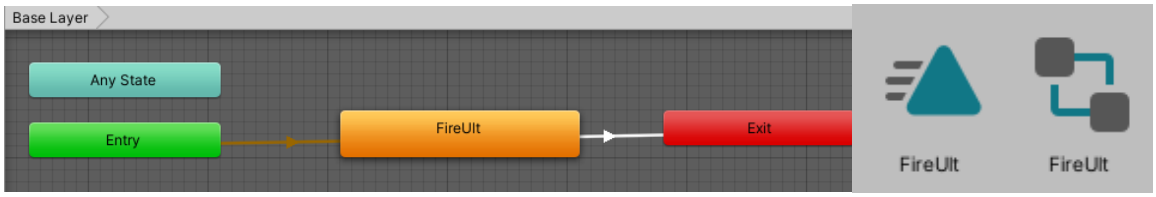
Kuva 23. Räjähdys Unityssä

5.2 Kaiken yhdistäminen

Kaikkien osien valmistuttua puuttui enää kaiken yhdistäminen. Aloitettiin kasaamalla kaikki tehosteen eri osat uuden tyhjän objektin alle Unityssä. Tämä mahdollistaa kaikkien osien liikuttamisen ja muokkaamisen tyhjän objektin kautta, tehden tehosteen käyttämisestä paljon helpompaa (kuva 24).

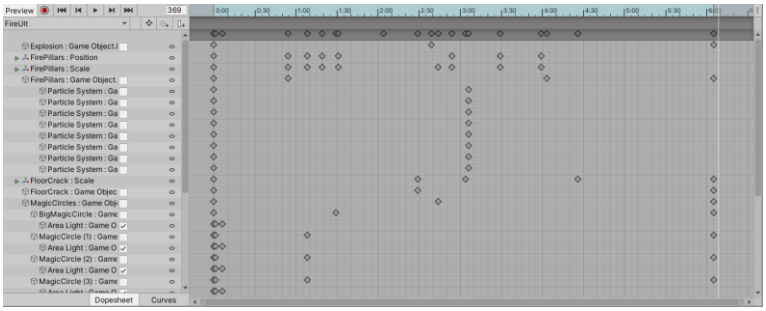
Kuva 24. Kaikki osat tyhjän objektin sisällä Unityssä

Tyhjään objektiin voi myös lisätä animaattorin, jolloin kaikkia osia ei tarvitse animoida ja synkronoida erikseen. Tämä onnistuu luomalla tyhjä animaattori ja animaatiotiedosto, jonka jälkeen ne kytketään objektiin, joka sisältää kaikki tehosteen osat (kuva 25). Koska kaikki osat ovat animaattorin alla, niin niitä pystyy käyttämään samassa animaatiossa.



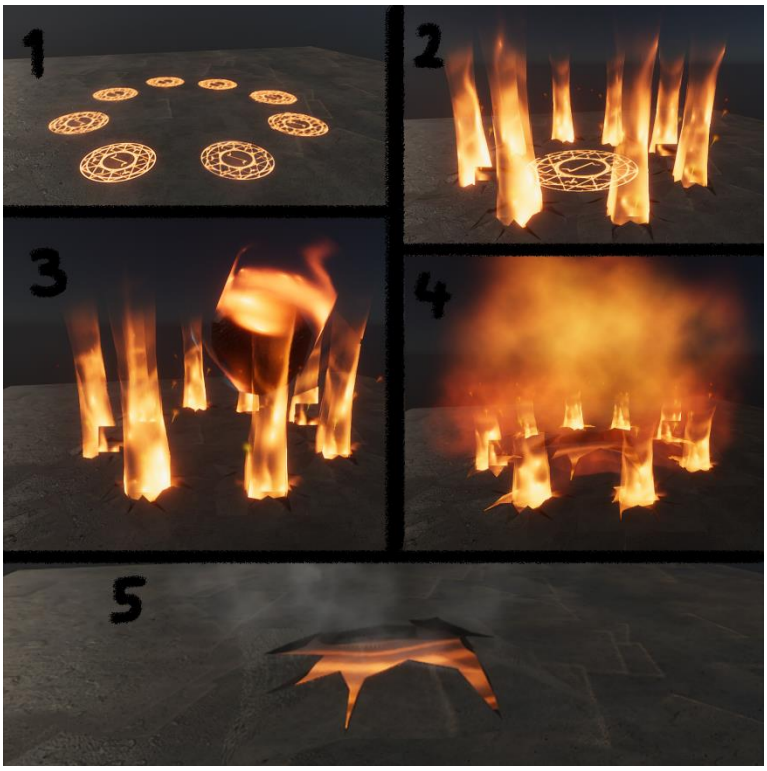
Kuva 25. Unity-animaattori ja uudet tiedostot

Avaamalla animaatioikkunan ja painamalla Record-näppäintä pystyy aloittamaan animaation luomisen. Record-tilan ollessa päällä kaikki pelin sisällä tekemät toiminnot tallentuvat animaatioon valitulle kohdalle. Tämä on helppo ja nopea tapa saada eri objektit ilmestymään tai liikkumaan halutulla hetkellä. Animaation aikana voi myös säätää objektien arvoja, esimerkiksi kuinka paljon partikkeleita partikkelisysteemit käyttävät. Dopesheet-ikkunassa säädetään aikajanalla olevia palloja, kun taas Curves-ikkunassa voidaan muuttaa, kuinka tasaisesti tai nopeasti arvot muuttuvat Dopesheet-ikkunassa olevien pallojen välillä (kuva 26).



Kuva 26. Unity-animaation aikajana.

Nopea tapa luoda animaatioita on liikuttaa y-akselin arvoa. Esimerkiksi tulipilarin ilmestymisanimaatiossa pilarit alkavat nolla-arvolla y-akselissa ja kasvavat täyteen mittaan. Koska pilarissa on kiinni myös pala haljennutta maata, niin tämän nopean animaation ansiosta näyttää kuin maa ensin halkeaisi ja sieltä lentäisi tulta. Jokaisen vaiheen tehoste asetettiin kestämään noin sekunnin tai kaksi ja menemään hieman päällekkäin. Asetettua kaikki eri osat laukeamaan oikeaan aikaan, kokonaisuus oli valmis. Ensin taikaympyrät ilmestyvät näyttäen pelaajalle, että jotain on tulossa ja kannattaa väistää. Seuraavaksi tulipilarit räjähtävät ulos maan sisältä lukiten pelaajan sisälle ja niiden sisään ilmestyy iso taikaympyrä signaloiden hyökkäyksen tulevan. Lopuksi meteori iskeytyy ympyrän sisään aiheuttaen räjähdyksen ja savuavan kraatterin, joka sisältää laavaa (kuva 27).



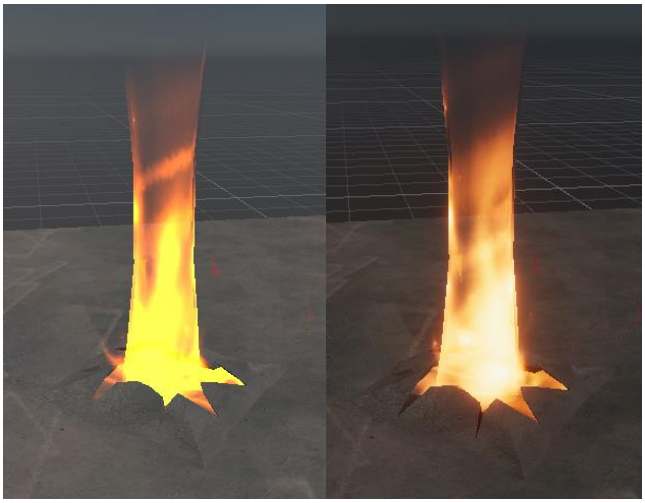
Kuva 27. Viimeistellyn tehosteen eri vaiheet

5.3 Loppuhiomiset

Vaikka tehoste on tässä kohtaa valmis, niin se ei tarkoita, että sitä ei voisi parannella. Hyvä tapa saada tehoste, sekä koko peli, näyttämään paremmalta on lisätä jälkikäsittelyä. Unityssä tämä on usein tehty Post Processing Stack-nimisellä ohjelmalla. Se on Unityn tekemä, mutta täytyy ladata erikseen uuteen projektiin. Meteori-iskutehosteen projektissa käytettiin HDR-putkiprofiilia, joka tekee tämän automaattisesti (kuva 28). Kaikista eniten tehosteiden kannalta vaikuttaa Bloom. Kytkemällä tämän päälle kaikki esineet, joilla on emissioarvo isompi kuin nolla, alkavat hohtamaan. Tämä korostuu paljon tehosteissa, etenkin hohtavissa partikkeleissa.

Kuva 28. Unityn jälkikäsittelyasetukset

Muut asetukset jälkikäsittelyssä auttavat myös tehosteiden parantamisessa, mutta valtaosa niistä vaikuttaa paljon koko pelin tuntumaan. Esimerkiksi Tonemapping-asetuksella voi muuttaa, miten värit käyttäytyvät kamerassa (kuva 29).



Kuva 29. Tulipilari ilman jälkikäsittelyä (vasemmalla) ja sen kanssa (oikealla)

Muita tapoja parantaa tehosteita on esimerkiksi lisätä kameraan tehosteita, kuten tärinää, sumennusta, reunojen tummennusta ja niin edelleen.

6 Loppusanat

On monia eri tapoja tehdä tehosteita, ja tämän opinnäytetyön tavoitteena oli tutkia niitä. Olen toiminut yli vuoden tehostegraafikkona, joten lähdin luomaan pohjaa työlle tekemäni tutkimuksen perusteelta. Tämän jälkeen aloitin tutkimukset taiteelliselta kannalta esimerkiksi tutkimalla värejä ja niiden vaikutusta pelaajassa. Itse en ole kovin hyvä värien käytössä, joten tästä oli paljon hyötyä. Opin myös värien käyttötarkoituksista eri kulttuureissa, eli että samat värit eivät välttämättä tarkoita samaa asiaa kaikille katsojille. Tästä on paljon apua pelaajan ohjaamisessa sekä symbolismissa.

Ei ole vain yhtä oikeaa tapaa tehdä tietty tehoste, ja kaikilla graafikoilla on omat suosikkityökalut ja työtavat. Tehosteiden tekeminen on kuin pulmien ratkaisemista, ja jokainen graafikon hallitsema työtapa on uusi vihje pulman ratkaisemiseksi. Tämän takia lähdin seuraavaksi tutkimaan kaikkia eri työkaluja ja työtapoja. Esimerkiksi itselleni sprite-arkit ja animaatio ovat tuottaneet ongelmia. Opinnäytetyötä tehdessä sain näistä paremman käsityksen.

Olen aina tehnyt tyyliteltyjä tehosteita, joten nyt halusin tehdä realistisen ja monimutkaisen tehosteen. Aloitin tutkimalla Unityn realismiin perustuvaa työtapaa ja opettelemalla sen käyttöä. Koska tein työtä yksin ilman ohjelmoijien apua, niin tutustuin uusiin työkaluihin, joiden avulla pystyin tekemään varjostimia. Tämä oli iso apu tehosteen tekemisessä ja tulee tulevaisuudenkin projekteissa nopeuttamaan työtä. Koska tehosteessa oli yhdistettynä monta eri osaa, niin jouduin tutkimaan paljon niiden yhdistämistä ilman, että kaikki hajoaisi. Olen lopputulokseen tyytyväinen ja tehostetta tehdessä oppimistani asioista on varmasti apua tulevaisuudessa.

Lähteet

1. People make Games (2019). Why the sound of a gun had to be nerfed in Wolfenstein: Enemy Territory. Haettu 4.2.2020, linkki: https://youtu.be/RDxiuHdR_T4

2. GDC 2017. Visual Effects Bootcamp: Artistic Principles of VFX. Haettu 10.2.2020, saatavilla: https://youtu.be/-L2JvngkkWw

3. Riot Games, So You Wanna Make Games?? | Episode 7: Game VFX, 2018. Haettu 10.2.2020, saatavilla: https://www.youtube.com/watch?v=3QKK2o5rWSQ

4. Riot Games, The Complete Guide To Creating Visual Effects Within League of Legends. Haettu 14.2.2020, saatavilla: https://nexus.leagueoflegends.com/wp-content/uploads/2017/10/VFX_Styleguide_final_public_hidpjqwx7lqyx0pjj3ss.pdf

5. Jason Keyser, Artistic principles of VFX #4: Gameplay, 2018. Haettu 14.2.2020, saatavilla: https://www.youtube.com/watch?v=BOE8osaPzOY

6. Blender Guru, Understanding Color, 2014. Haettu 15.4.2020, saatavilla: https://www.youtube.com/watch?v=Qj1FK8n7WgY

7. Anna Lundberg, Color meanings and the art of using color symbolism, 2019. Haettu 15.4.2020, saatavilla: https://99designs.com/blog/tips/color-meanings/

8. Jeremy Girard, Visual Color Symbolism Chart by Culture, 2019. Haettu 15.4.2020, saatavilla: https://www.lifewire.com/visual-color-symbolism-chart-by-culture-4062177

9. Kuva 3. Itten, Johannes: Värit taiteessa: Värien subjektiivinen kokeminen ja objektiivinen tunnistaminen johdatuksena taiteeseen, 2004. Haettu 15.4.2020

10. GDC, Art Direction VFX for Stylized Games, 2017. Haettu 23.3.2020, saatavilla: https://www.youtube.com/watch?v=x30j-PUN1G4

11. Francisco García-Obledo Ordóñez, VFX for Games Explained, 2017. Haettu 23.3.2020, saatavilla: https://80.lv/articles/vfx-for-games-explained/

12. Arkane Studios, Dishonored 2, 2016. Haettu 23.3.2020, saatavilla: https://store.steampowered.com/app/403640/Dishonored_2/

13. tharlevfx, VFX UV Unwrap Tehcniques, 2020. Haettu 2.4.2020, saatavilla: https://www.youtube.com/watch?v=6VD4CtTFBbg

14. Nikola Damjanov, Free VFX Patterns for Subtance Designer, 2018. Haettu 3.4.2020, saatavilla: https://80.lv/articles/free-vfx-patterns-for-substance-designer/

15. Unity, The High Definition Render Pipeline: Getting Started Guide for Artists, 2018. Haettu 8.4.2020, saatavilla: https://blogs.unity3d.com/2018/09/24/the-high-definition-render-pipeline-getting-started-guide-for-artists/

16. u/forkafork, Alchemy circles procedural generator, 2018. Haettu 12.4.2020, saatavilla: https://ciaccodavi.de/qbdp/acg/

17. Unity, Shader Graph Node Library. Haettu 15.4.2020, saatavilla: https://docs.unity3d.com/Packages/com.unity.shadergraph@6.9/manual/Node-Library.html

18. Kuva 17. Unity, Shader Graph Node Library Procedural Nodes. Haettu 15.4.2020, saatavilla: https://docs.unity3d.com/Packages/com.unity.shadergraph@6.9/manual/Procedural-Nodes.html

19. Niklas Taipalinen, Meteor Strike Magic, Haettu 22.5.2020, saatavilla: https://youtu.be/Dei45VeFEiE