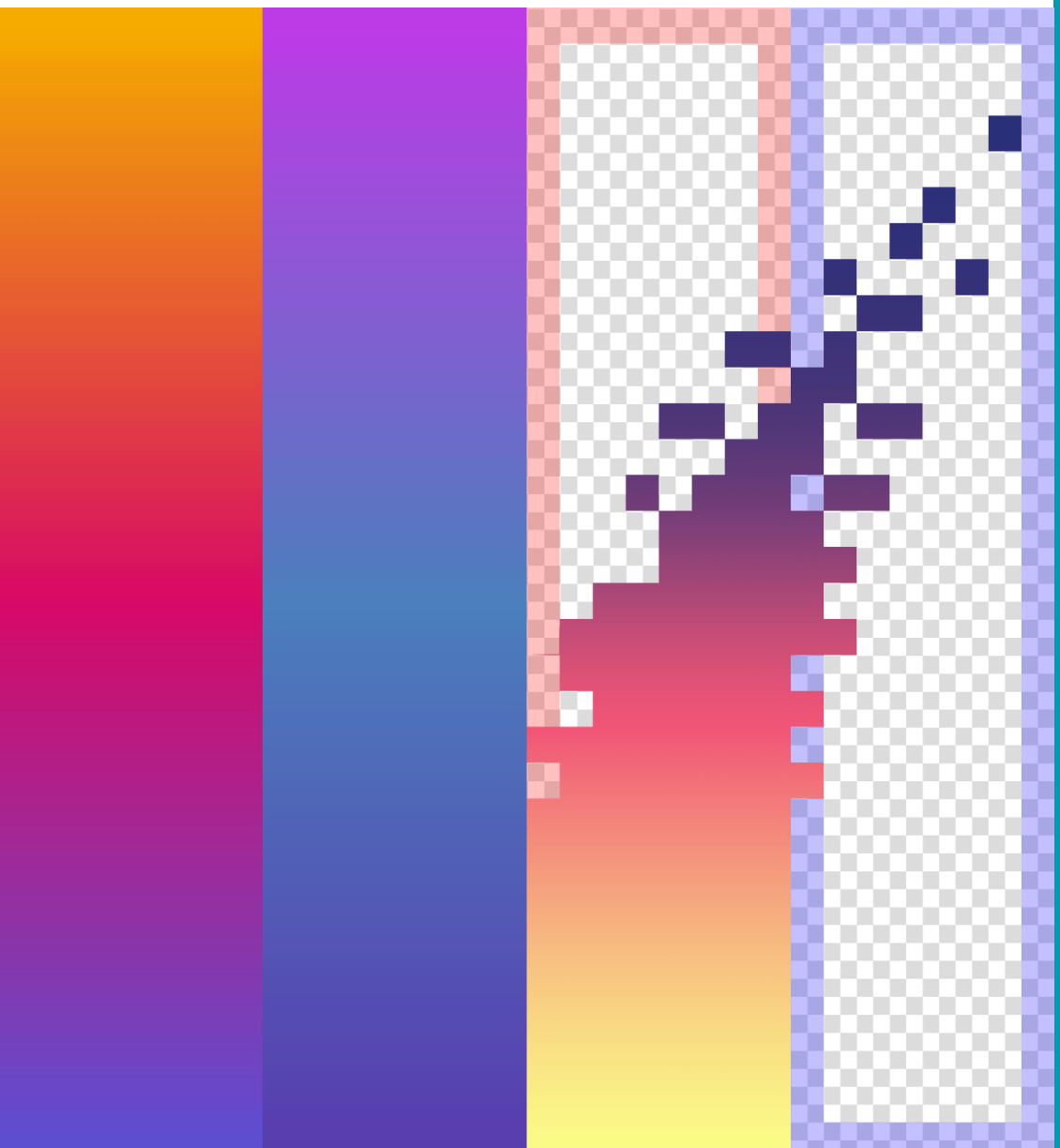
Eero Kinnunen

Liukuväriteksturointityötavan eteenpäinvienti Unity-pelimoottorissa



Tradenomi

Tietojenkäsittely

Syksy 2022

KAMK • University of Applied Sciences

Tiivistelmä

Tekijä: Kinnunen Eero

Työn nimi: Liukuväritekstusointityötavan eteenpäinvienti Unity-pelimoottorissa

Tutkintonimike: Tradenomi, tietojenkäsittely

Asiasanat: Liukuväri, teksturointi, UV-mappaus, työkalu, Unity-pelimoottori

Opinnäytetyön tavoitteena oli kehittää työkalu teksturointiin pelimoottorin sisällä. Kehittämisessä huomioitiin työtapaan liittyviä haasteita ja ongelmia. Liukuväritekstusointi on ollut pinnalla sosiaalisessa mediassa kuten Twitterissä ja 3D-taiteen näyttösivulla Sketchfabissä.

Tässä opinnäytetyössä arvioitiin liukuväritekstusointitekniikan taustaa ja siihen liittyviä huomioita. Lisäksi käytiin läpi, miten kyseistä työtapaa lähdetään toteuttamaan, mitä siihen kuuluu, miten sitä voidaan hyödyntää ja miten sitä voidaan viedä eteenpäin. Tässä opinnäytetyössä sivuttiin myös työkalun luontia Unity-pelimoottorin sisällä ja gradienttityötapaan liittyvän työkalun tekemisen vaiheita.

Opinnäytetyön tuloksena on työkalu käytettäväksi gradienttitekstusointityötavan kanssa, joka jo tässä vaiheessa nopeuttaa työskentelyä huomattavasti. Johtopäätöksenä voidaan todeta, että työtavan kanssa työskentelyä voidaan todella ehostaa Unityn sisäisen työkalun avulla.

Abstract

Author: Kinnunen Eero

Title of the Publication: Gradient Texturing Pipeline in Unity Game Engine

Degree Title: Bachelor of Business Administration, Business Information Technology

Keywords: Gradient, texturing, UV mapping, tool, Unity, Blender, lazy unwrapping, C#

The goal of this thesis was to develop a tool based on a specific texturing pipeline and extend the Unity game engine by implementing a plugin. The plugin was originally created with a school project in mind, but it extended afterwards into a free time and thesis project. The purpose of the plugin was to support the game's texture pipeline inside Unity game engine.

The research part of this thesis is focused primarily on getting familiar with the Unity game engine's features and tools, Unity's C#, extending Unity Editor and developing a Unity plugin. Finishing a simple albedo version of the tool was followed by designing the various components for the plugin and agreeing on specific requirements of the plugin. The aim of the plugin was to have simple color arrays and to render them inside the game engine. These requirements were also set as an evaluable part to determine the success of the project. After the research, the project part was started by designing the various components of the plugin. After the design phase, the work moved into the implementation phase, where the plugin was implemented in the Unity game engine. The plugin was implemented with C# programming. As a result of this work, a plugin is designed to be the main plugin for the pipeline in the Unity game engine.

Käytetty sanasto

Blender - Avoimeen lähdekoodiin perustuva 3D-mallinnusohjelma.

JSON - JavaScript Object Notation. Avoimen standardin tiedostomuoto tiedonvälitykseen.

Liukuväri - Vähintään kahden värin, usein lineaarinen, siirtymä.

Materiaali - Materiaali Unity-pelimoottorissa. Se toimii kiinnityspintana shaderin, tekstuurien ja erilaisten arvojen välillä.

PBR - (physically based rendering) fysiikkaperusteinen renderöinti, jolla pyritään saamaan aikaan realistista grafiikkaa.

Tunniste – Esiasetukseen liitetty string-muuttuja arvo, joka auttaa järjestelyssä ja löytämisessä.

Unity-pelimoottori - Unity on Unity Technologiesin kehittämä maksuton monialustainen pelimoottori, jolla voidaan kehittää kaksi- ja kolmiulotteisia videopelejä.

UV-kartoitus - Topologian tekstuuriin sitova asettelu.

UV-saari - Tietystä vertekseistä koostuva saareke UV-kartoitustilassa.

Varjostin (Shader) - 3D-objektin tapa ottaa sisään erilaiset tekstuurit ja arvot ja reagoida valaistukseen.

Sisällys

1 Johdanto 5

2 Värigradienttien esittelyä 6

3 3D-mallinnus 9

4 Sosiaalisessa mediassa 16

5 Kuinka käyttää liukuvärejä? 20

6 Työkalun suunnittelu 22

6.1 Perusta 22

6.2 PBR 24

7 Työkalun tekeminen Unity pelimoottorissa 25

7.1 Työkalun toteutus 26

7.2 Työkalun laajennus 40

8 Työkalun käyttöesimerkkejä 41

9 Työkalun eteenpäinvienti ja tulevaisuus 44

10 Yhteenveto 48

11 Lähteet 49

1 Johdanto

Pelejä tehdessä on usein erilaisia teksturointitapoja. Kuten vanhimmissa 3D-peleissä käytetty verteksien maalaus ja uudemmissa peleissä käytetty PBR eli fysiikkaperusteinen teksturointi. Tässä opinnäytetyössä keskitytään kuitenkin erääseen teksturointi tapaan, joka on viime vuosina pyörinyt sosiaalisessa mediassa erityisesti Joycen, Minion's Art - nimimerkillä kulkevan artistin, toimesta. Tämä teksti myös sivuaa peliä nimeltä Toy Thieves, jota olen ollut tekemässä koulun aikana.

Tämä teksti käsittelee myös hieman sitä, miten tehdä työkalu Unity-pelimoottorissa kyseisen työtavan nopeuttamiseksi ja pohdintaa, miten tätä työtapaa voidaan nopeuttaa entisestään. Tekstin ja työkalun eteneminen perustuu omiin havaintoihin 3D-taiteen teosta kuitenkin perustellen sitä lähteillä ja verraten sitä muihin toimintatapoihin.

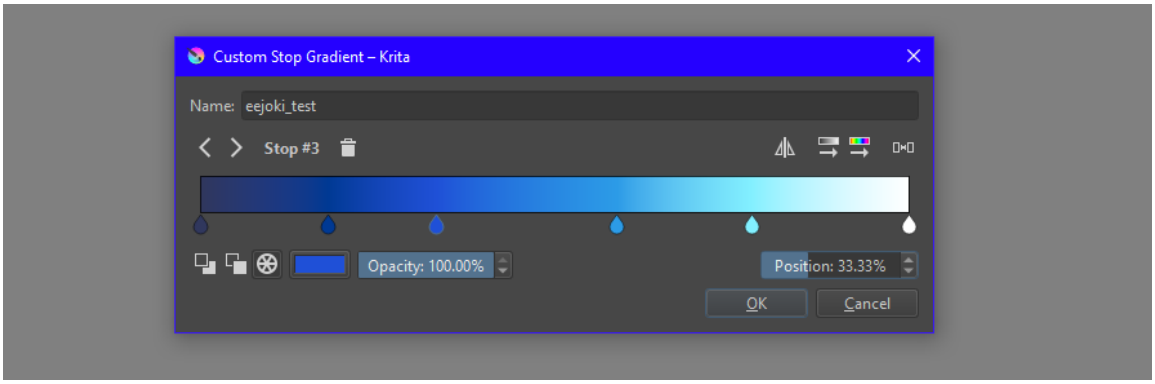
Aihe kiinnostaa itseä myös teknisen taiteen kannalta ja on ollut pitkään aikeissa yhdistää pelitaidetta ja koodia ja tässä oli siihen oiva sauma. Toivon saavani tästä myös hyvää näyttöä portfoliooni.

2 Värigradienttien esittelyä

Gradientti sana tulee englannin kielen kaltevuutta tarkoittavasta sanasta. Värigradientti kuitenkin kuvaa yleensä siirtymää kahden väriarvon välillä, usein lineaarisesti. Gradientteja on monenlaisia. Tässä opinnäytetyössä keskitytään nimenomaan niiden käyttöön pelinkehityksessä teksturoinnin osana.

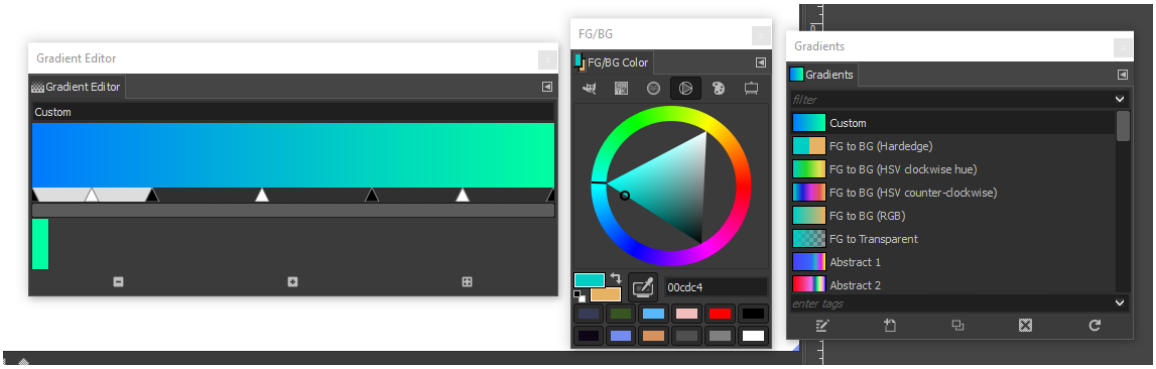
Gradient tool eli gradienttityökalu löytyy muun muassa Photoshopista. Alle on listattu kuvia gradienttien tekotyökaluista eri kuvankäsittelyohjelmissa. Jokaisessa ohjelmassa gradientti työkalu on hieman erilainen, mutta niistä yleensä löytyy yhtäläisiä ominaisuuksia, kuten kunkin pisteen väriarvot ja paikka janalla.

Krita-kuvanmuokkausohjelmassa esiasetukset, itse gradienttimuokkain ja väriarvot ovat erillisissä ikkunoissa. Kritan gradienttityökalusta löytyy myös mahdollisuus järjestää pisteet uudelleen väriarvon tai värisävyn perusteella. Kuten myös mahdollisuus kääntää koko gradientin järjestys, sekä asettaa väripisteet tasaisin välein janalle.



Kuva 1. Custom Stop Gradient -työkalu Krita 5.0.5 -kuvankäsittelyohjelmasta

Gimp-kuvanmuokkausohjelmassa gradienttieditorin esiasetusten esikatselut ovat horisontaaliset, mutta niistä saa myös neliöt valinnan kautta. Värin valinta itsessään on hieman vaikea. Gimpissä yksittäisiä väripisteitä kutsutaan nimellä ”stop.” Niiden välissä on myös liukuva piste, joka määrittelee niiden siirtymän keskikohdan.



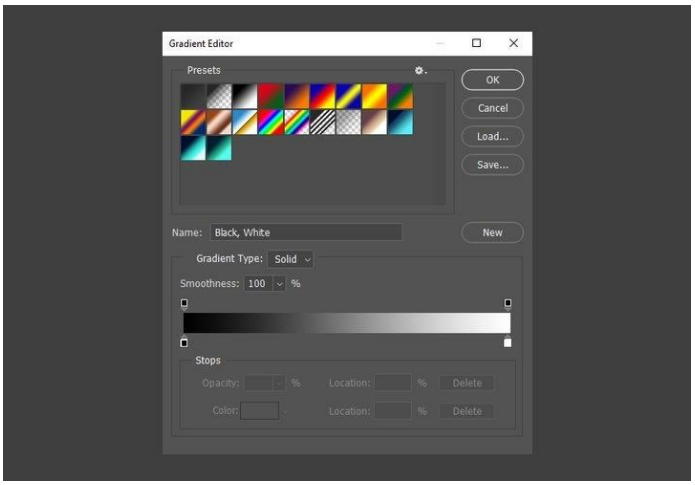
Kuva 2. Gradienttieditori Gimp-kuvanmuokkausohjelmasta

Clip Studio Paint-ohjelman Gradient Editorissa (kuva 3) on mukana esiasetukset vaakasuuntaisessa listauksessa. Hienosäätöjä varten mukana on myös käyräeditori, jota voi halutessaan käyttää valitun ja sekä sitä seuraavan pisteen välisen siirtymän säätämiseen. Yksittäisten väripisteiden nimenä käytetään nodea eli solmua, ainakin pisteen poistopainikkeen ja valintanuolien kohdalla.

Kuva 3. Clip Studio Paintin Gradient Tool -ikkuna

Photoshopin gradienttieditori-ikkunassa gradienttiesiasetukset esiityvät viistoon oikealle alas menevissä neliön muotoissa paikoissa. Ne vievät vähemmän tilaa ja valitun esiasetuksen nimi on sijoitettu itse nimenmuokkauskenttään. Photoshopissa yksittäisiä väripisteitä kutsutaan myös

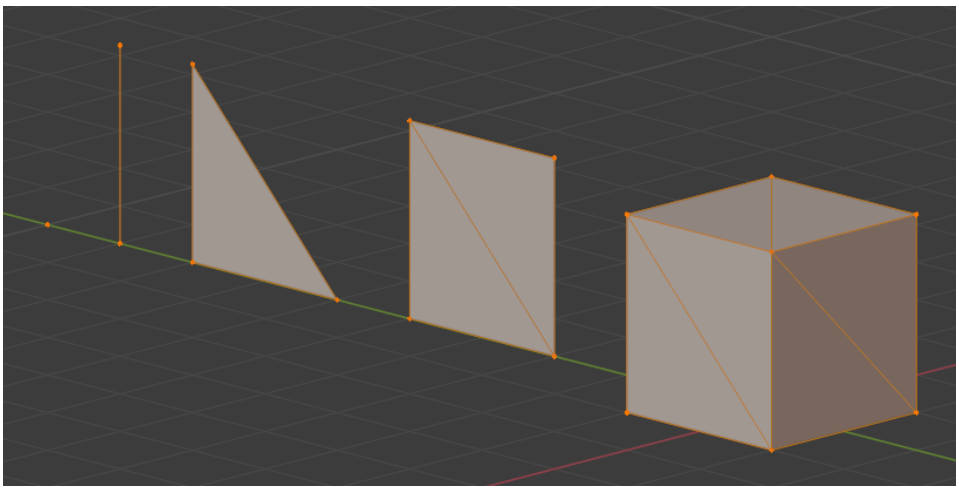
nimellä ”stop” eli pysäytin niin kuin Gimp-kuvanmuokkausohjelmassa. Gimpin välipisteen ja Clip Studio Paintin kurvin asemaa pitää ”Smoothness”- eli sulavuusarvo.



Kuva 4. Photoshopin gradienttieditori

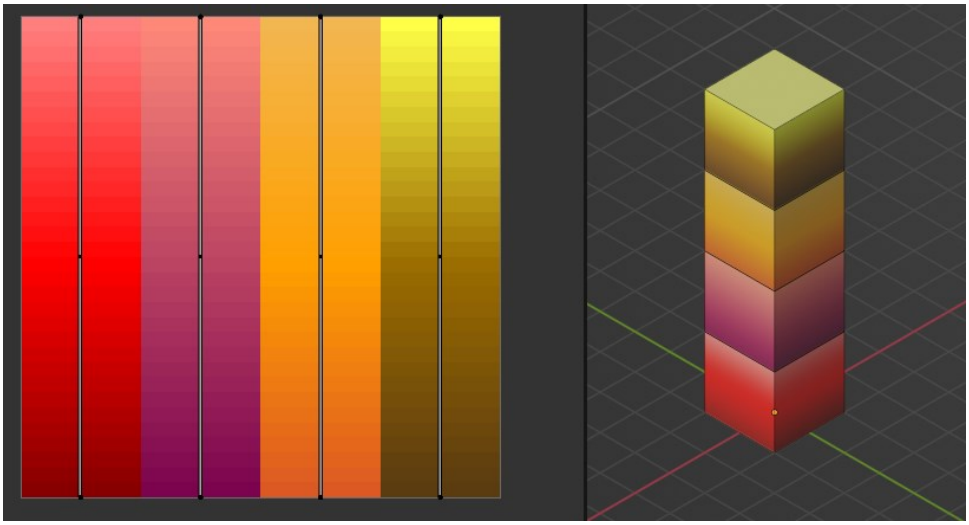
3 3D-mallinnus

Työtavan selittämiseksi käydään läpi 3D-mallinnuksen perusasioita. Kuvassa 5 nähdään kärjen, kahdesta kärjestä muodostuvan särmän, kolmesta kärjestä tai kolmesta särmästä täytetyn kolmion. Kahdesta kolmiosta saadaan muodostettua tahko ja näitä yhdistelemällä saadaan aikaan 3D-malleja, kuten tässä tapauksessa kuvassa näkyvän kuudesta tahkosta muodostuvan kuution.



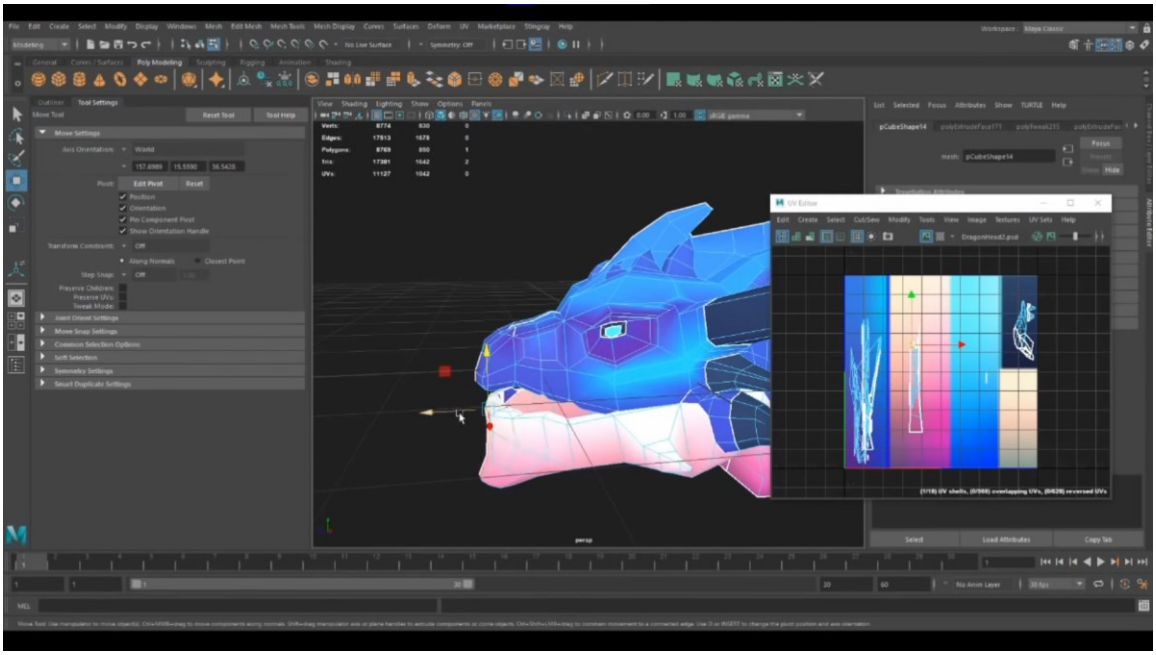
Kuva 5. Kuvassa vasemmalta oikealle kärki, särmä, kolmio, tahko ja kuutio

Kuutioista taas voidaan rakentaa kolmen kuution torni kuten nähdään kuvassa 6. Tässä tekstissä käsiteltävässä työtavassa tahkot UV-kartoitetaan gradienttiliuskoista tehdyn tekstuurin päälle. Kuvassa (6) nähdään myös tekstuuri, jossa on neljä eriväristä gradienttia.



Kuva 6. Esimerkki liukuväri-UV-kartoituksesta 3D-ohjelmassa

MinionsArtin esimerkin mukaan teksturointitapaa on hyvä lähestyä kuution kautta, unwrapata se katselukulmasta ja liu'uttaa kuution UV-saareke halutulle sävylle gradienttitekstuurin ollessa taustalla. MinionsArt käyttää 3D-työkalunaan useimmiten Autodeskin Maya LT:tä, kuten kuvassa 7, mutta perus UV-kartoituksenperiaatteet on rinnastettavissa muihin 3D-työkaluihin, kuten tämän opinnäytetyön esimerkeissä käytettävään Blenderiin.



Kuva 7. Kuvakaappaus MinionsArtin videosta [1]

Ensimmäinen askel valmiin mallin UV-mappauksessa on projektoida koko malli pystysuunnassa ensimmäiselle pohjagradientille. Tämä tapahtuu valitsemalla Joycen tapauksessa Maya LT:stä ”Create UVs” ja sitä kautta ”Planar Mapping”, joka vastaa Blenderin ”Project From View”-UV komentoa.

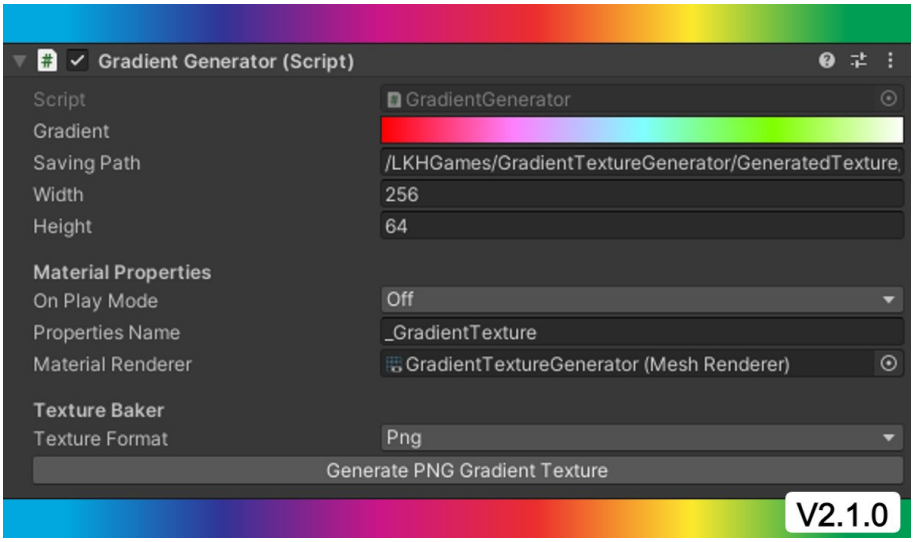
”Shadeless” tai ”Unlit”-shader toimii parhaiten 3D-mallinnunsohjelman puolella, koska sillä saa parhaiten näkyviin työtavan edut, jotka perustuvat nimenomaan itse varjostuksen vaikutuksen luontiin gradienttien avulla [2, 2:47].

Tähän mennessä Minion'sArt on käyttänyt erilaisia PBR-variaatioita pelinkehityksessään, mutta ne ovat yleensä olleet erilaisten shadereiden maskien muodossa eli sinänsä mitään vakiintunutta eli yleisesti käytössä olevaa työkalua ei työtavalle ole vielä kehitetty tai hänen käytössään. Muutama Unitylle kehitetty gradienttityökalu kuitenkin löytyy, joista seuraavaksi kerrotaan hieman.

Kuvan 8 työkalu on nimeltään Gradient Generator eli gradienttigeneraattori. Laajennusta suositellaan käytettävän Odinin eli erillisen käyttöliittymälaajennuksen kanssa. Odin on Unity Storesta saatava laajennus, jota käytetään erilaisten editori-ikkunoiden luonnissa esimerkiksi työkaluille. Odin on Sirenix-nimisen käyttäjän kehittämä. Työkalu on rajoittunut rendaamaan gradientteja vierekkäin pystypäin ja käyttää Unityn omaa gradienttiluokkaa. Työkalu on vuodelta 2018, eikä sitä ole päivitetty sen jälkeen. Gradienttien esiasetukset perustuvat tallennettaviin Unityn gradientteihin eikä siinä ole PBR-tekstuurityyppeihin liittyviä ominaisuuksia taikka erillistä kuviotekstuuri-integraatiota. Työkalun yhteydessä innostajaksi on mainittu Minion'sArt. Lisäksi työkalun sivulla on kerrottu siitä, miten gradientteja voi käyttää esimerkiksi tuuliefektin saavuttamiseksi liikuttaen 3D-mallin eri osia gradienttitekstuurin tummuuden mukaan. Esimerkkinä tästä sivustolla on puu, jossa menee gradientti alhaalta ylös. Työkalua ei löydy Unityn omasta Asset Storesta, ainakaan samalla nimellä. Työkalun on tehnyt henkilö nimeltä Daniel Snd.

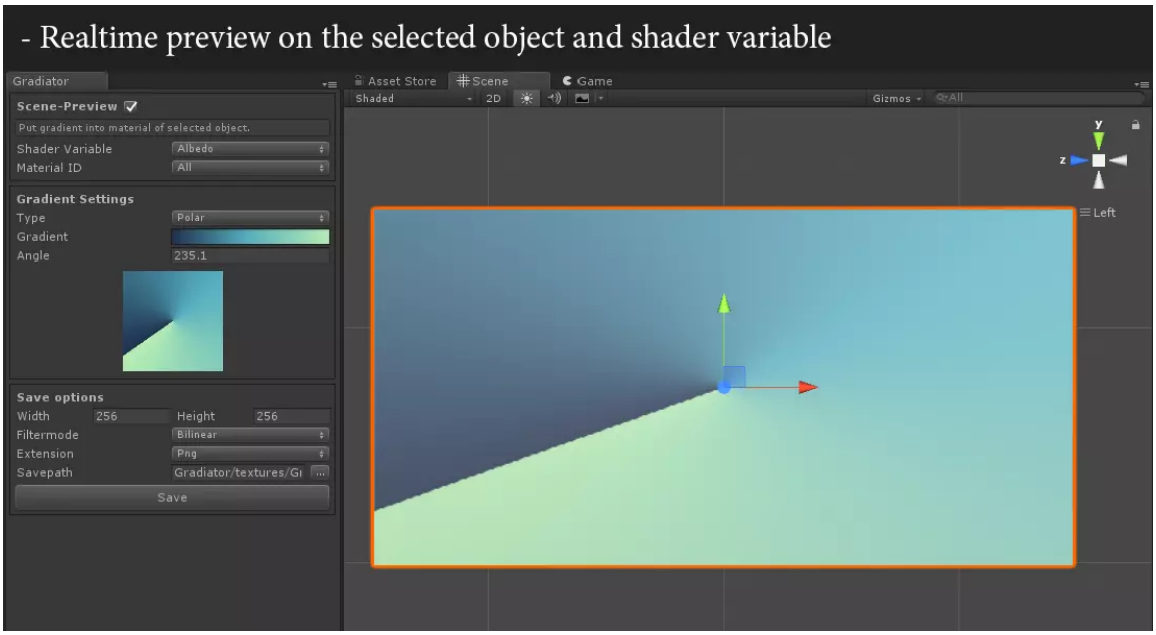
Kuva 8. Gradient Generator -työkalu [3]

Kuvan 9 työkalun nimi on Gradient Texture Generator. Sillä on melkein sama nimi kuin edellisellä työkalulla, mutta lisänä on "Texture" -osa. Työkalu löytyy ilmaisena Unity Asset Storesta. Tekijän kuvauksen mukaan kyseessä on yhden painalluksen generaattori. Se käyttää gradientteja rendatakseen Unityn omaa gradienttiluokkaa. Työkalua on päivitetty viimeksi 21. marraskuuta eli ihan vasta. 26 käyttäjää on suosikoinut kyseisen työkalun. Kolme käyttäjää on arvioinut kyseisen työkalun viidellä tähdellä.



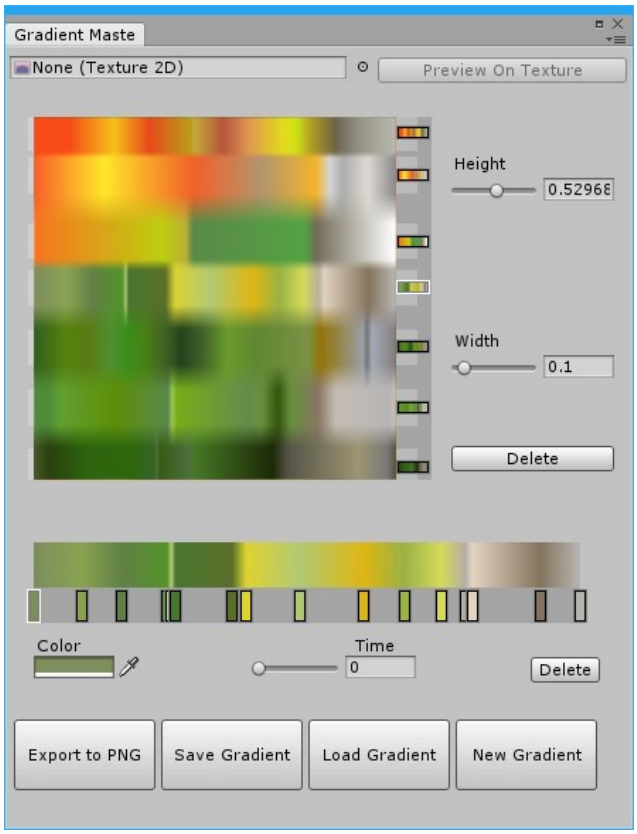
Kuva 9. Gradient Texture Generatorin UI [4]

Kuvan 10 Gradiator on parametrinen gradienttigeneraattori Unity-editorissa. Sillä voi luoda lineaarisia, neliöllisiä, säteittäisiä ja polaarisia koordinaattigradientteja. Työkalussa voit valita kohdevarjostinmuuttujan luodaksesi gradientin PBR-kanaville, kuten kiilto ja emissiivinen. Sen avulla liität luodun gradientin mukautettuihin varjostinmuuttujiin reaaliajassa. Gradientteja voidaan kiertää, kokoa muuttaa, suodattaa ja tallentaa eri muodoissa reaaliaikaisella esikatselulla, joka näyttää lopullisen kuvan. Gradiator ei kuitenkaan sisällä käyttäjän hallittavaa gradienttiasettelua. Työkalun on kehittänyt saksalainen Klonk Games, joka tunnetaan parhaiten Shift Happens -pelin kehittäjänä. Työkalua on päivitetty viimeksi 25. elokuuta 2017.



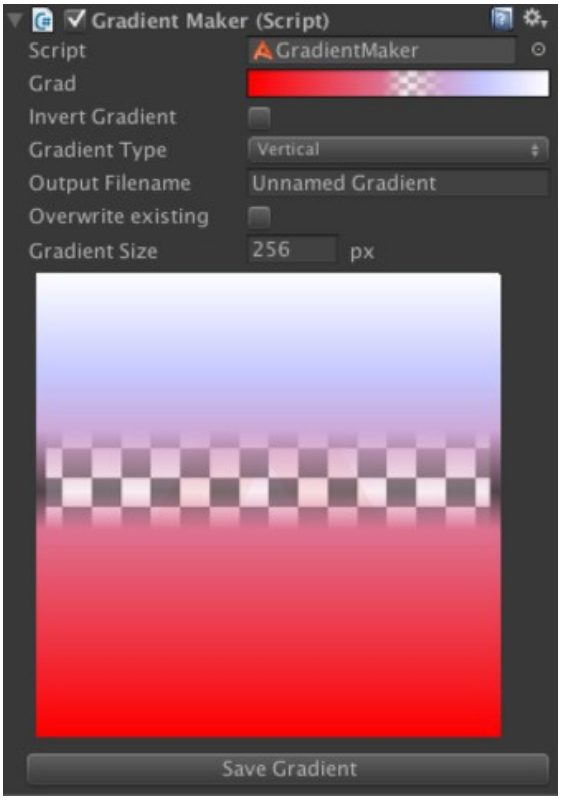
Kuva 10. Gradiator-työkalun UI [5]

Seuraavaksi on kuvan 11 Gradient Master eli gradienttimestari. Tämä työkalu tukee vakio-, metalli- ja heijastavaa renderöinti pipelineä. Kuvasta näkee, että gradientit on aseteltu pinoksi. Vuonna 2016 tehdyssä postauksessa lukee, että tulossa pian Unity Asset Storeen, mutta kyseisellä nimellä ei löydy osumia tänä päivänä. Työkalun on kehittänyt henkilö nimeltä Mark Petro. Gradient Masterin mukana tulee myös muutama shaderi, jotka on suunniteltu gradienttitekstuurointi mielessä muun muassa perus Standard, Metallic ja Specular. Lisäksi Speed Tree eli nopea puu -generointi, Terrain eli maasto, Particles eli partikkelit ja Legacy eli perinteinen shaderi.



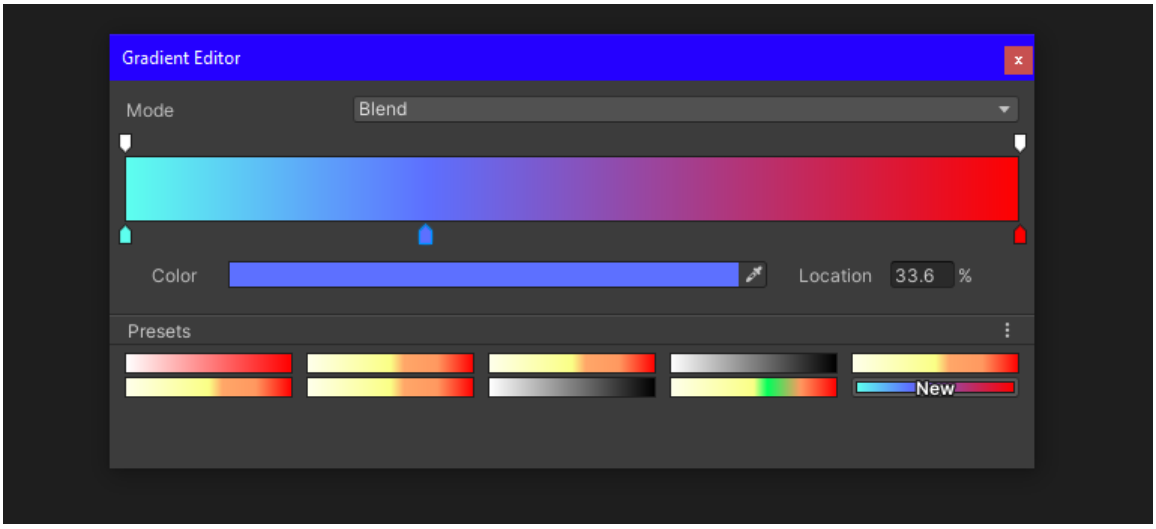
Kuva 11. Gradient Master -työkalun ikkuna [6]

Seuraavana on kuvan 12 Gradient Maker. Ominaisuuksina tästä työkalusta löytyy horisontaalisten, vertikaalisten ja radiaalisten gradienttien helppo renderöinti. Lisäksi tekijän mukaan kyseessä on tämän työkalun lite- eli kevyt versio. Jos sille on kysyntää, kehittäjä julkaisee siitä "Pro"-version, jossa on paljon kattavampi ominaisuussarja ja mukautettu editori-ikkuna. Gradient Maker Lite löytyy tällä hetkellä Unity Asset Storesta [7] ilmaisena ja Andy Green -nimisen käyttäjän julkaisemana. Sillä on tällä hetkellä 28 arviota, joiden keskiarvo on viisi tähteä viidestä tähdestä ja 164 ovat lisänneet suosikkeihinsa kyseisen työkalun. Työkalua on päivitetty viimeksi 19 kesäkuuta 2019.



Kuva 12. Gradient Maker [7]

Unityn oma gradienttieditori löytyy muun muassa Shader Graphin ja partikkelieditorin yhteydestä. Kunkin väripisteen kohdalta löytyy väriarvo sekä prosenttiluku pisteen sijainnille. Työkalussa on päädytty käyttämään erillisiä alpha-arvoja, joiden pisteet näkyvät itse gradientin yläpuolella. Siitä löytyy kolmenpisteen merkin takaa libraryt eli kirjastot, joille voi antaa nimen ja valinnan siitä, että sijaitseeko se ”preferences”- vaiko projektikansiossa. Näkyviin saa vain yhden kirjaston kerrallaan. Esiasetukset voi järjestää joko lista- tai ruudukkomuodostelmaan. Niiden järjestys määrittyy luonnin mukaan tai niitä voi halutessaan järjestää uusiksi raahaamalla ja pudottamalla. Ensimmäisestä pudotusvalikosta voidaan valita tilaksi joko ”blend” eli sekoitus tai ”fixed” eli muuttumaton. Nämä tilat vaikuttavat siihen, miten kukin väripiste sekoittuu seuraavan kanssa. Yksittäisten pisteiden välille tätä ei kuitenkaan pysty määrittämään, eikä siitä löydy Gimp- tai CSP-ohjelmista löytyviä siirtymän hallintaan liittyviä säätimiä.

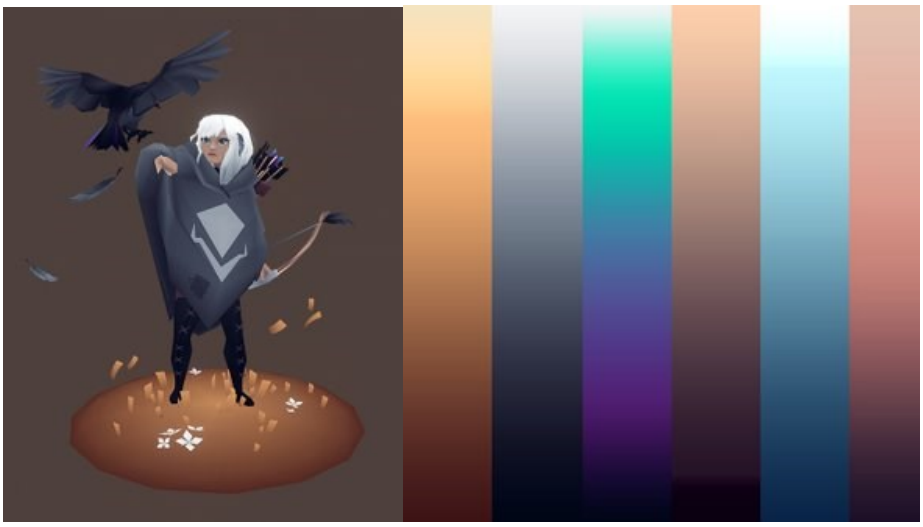


Kuva 13. Unityn gradienttieditori

4 Sosiaalisessa mediassa

Sosiaalisessa mediassa teksturointitapa on esiintynyt muutamassa eri paikassa. Twitterissä tämä kyseinen työtapa on ollut pinnalla lähinnä Joyce, MinionsArt -nimimerkillä kulkevan, pelinkehittäjän toimesta muun muassa erilaisten somepostausten ja tutoriaalien muodossa [8]. Ensimmäiset Joycen gradienttimallit ovat vuodelta 2014 Tumblrin mukaan [9].

Myös Sketchfabissä kyseinen teksturointitapa on ollut pinnalla, kun Sketchfab-sivusto piti gradienttitekstuurityötavan innoittaman haastekilpailun MinionsArtin kanssa yhteistyössä heinäkuussa 2019 #GradientChallenge -hashtagilla. Kilpailun voittanut ruotsalainen Arookh-nimimerkillä kulkeva Natascha Schirmuli kuvailee työtapaa näin: ”Tämä tekniikka on oikotie nopeudessa, mutta ei tiedossa, koska tarvitset hyvän valo- ja väritajunnan. Teet kaiken sen gradienttisi alueella.” [10] Kyseessä siis on yksinkertainen työtapa, joka kuitenkin vaatii silmää sekä valolle, että väreille.



Kuva 14. 3D-hahmo toteutettuna liukuväri työskentelytavalla [10]

Sketchfab antaa usein käyttäjilleen tunnustusta niin sanotuilla ”Staff Pickeillä” eli niin sanotuilla henkilöstön valinnoilla. Valintojen kriteereinä on, että mallit toimivat hyvin mobiililaitteilla, hyödyntävät tehokkaasti tekstuurikarttoja, geometrioita, materiaaleja ja luita rigatussa animaatiossa. Taiteellisena vaatimuksena mainitaan, että 3D-mallin on oltava mahtava, kerrottava hyvä tarina ja hyödynnettävä hyvin Sketchfabin ominaisuuksia, kuten huomautuksia, jotka on Sketchfäbiin rakennettu systeemi, joka koostuu kameralokaatioista ja selittävästä tekstistä, valaistusta ja jälkikäsittelysuodattimista.

Kuvan 15 artikkeli on kirjoitettu Sketchfab-tiimin pyynnöstä Staff Pick-tunnustuksen saaneeseen 3D-malliin liittyen. Gwendalyh Tohin kirjoittamassa artikkelissa [11] kerrotaan taidesuuntauksesta, jossa mainitaan muun muassa eräs Minion's Artin malleista, Animal Crossing ja The Legend of Zelda: Wind Waker. Referensseissä ja inspiraatiossa kerrotaan lähinnä lelureferensseistä. Kuva 15 on peräisin Sketchfabin artikkelista nimeltä ”Art Spotlight: Modular Environment Hexes”.

Kuva 15. Modulaarisia kenttäpalasia projektista nimeltä Toy Thieves [11]

Tässä opinnäytetyössä käsiteltävän työkalun alkuperä on kyseisen pelin kehityksen ajalta, mutta itse PBR-versio ja jatkokehitys on tapahtunut sen jälkeen. Kyseisessä versiossa väriarvojen tallentamiseen käytettiin Unityn omaa ”Swatces”-systeemiä. Kyseisessä systeemissä voi tallentaa yksittäisiä värejä erinimisiin kirjastoihin, mutta muuten systeemin ominaisuudet ovat rajalliset. Tätä systeemiä käytettiin projektissa muun muassa tiimivärien tallentamiseen.

Seuraavaksi luetellaan esimerkkipelejä mahdollisesta työtavan käytöstä tai minkä tyylisiin peleihin sitä voitaisiin hyödyntää. MinionsArt eli Joyce on käyttänyt gradienttitekstusointityötapaa muun muassa kuvan 16 AstroKat-projektissaan. Myös useat hänen Unity-tutoriaaleistaan keskittyy kyseisen pelin ympärille. Kuvasta voidaan nähdä gradientteja muun muassa puiden tekstuureissa.

Kuva 16. MinionsArtin oma Astro Kat Unity-peliprojekti [12]

Toy Thieves -pelin toisena pääinnoittajana toimi Sly 2: Band of Thieves -peli. Kuvan 17 Pariisi-kentässä nähdään kohtia, joissa voitaisiin käyttää gradienttiteksturointia, kuten vasemmalla näkyvän rakennuksen kupolissa, keskiosan katon ritilän suojaamassa lasissa, oikealla olevassa violetissa kupolissa ja osassa Eiffel-tornia. Lisäksi kattoihin voisi käyttää yksinkertaista toistuvaa kuviotekstuuria.

Kuva 17. Sly 2-pelin Pariisi-kenttä [13]

Animal Crossing -pelisarja ei käytä samaa teksturointitapaa, mutta gradienttiteksturointia voitaisiin käyttää saman tyylisessä grafiikassa hyödyksi. Animal Crossing käyttää hyväkseen myös yksinkertaisia kuviotekstuureita. Gradienttimaisuutta näkyy esimerkiksi kuvan 18 lehdissä ja kukissa.

Kuva 18. Kuvakaappaus Animal Crossing: New Leaf-pelistä [14]

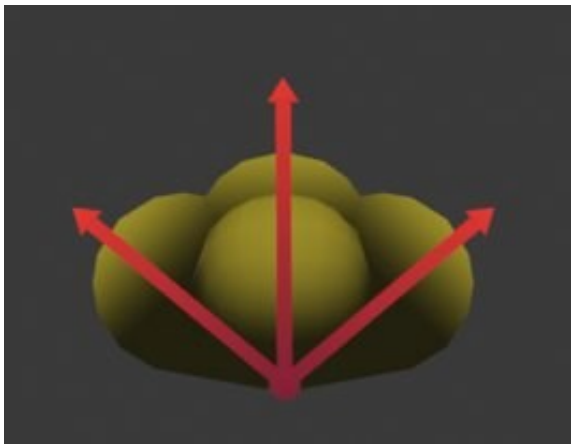
5 Kuinka käyttää liukuvärejä?

Eräs esimerkki gradientin suunnasta ja käytöstä on, kun UV-kartoittaa 3D-mallin siten, että gradientti kulkee alhaalta ylös. Tällaista varjostamista on hyödynnetty muun muassa erilaisissa MOBA eli taisteluareenamoninpeleissä kuten esimerkiksi Valven kehittämässä Dota 2:ssa (kuva 19). Kuitenkaan itse gradienttitieksturointia ei ole kyseisissä peleissä käytetty vaan se toimii lähinnä valoisuuden ohjenuorana teksturoidessa. Tämä tukee sitä, miten auringonvalokin käyttäytyy luonnossa, kun auringon valo tulee yläviistosti objektiin. Lisäksi se auttaa erottamaan pelihahmot pelialustasta. Tämä on helpoin toteuttaa silloin, kun kaikki pelin 3D-mallit seuraavat samaa sääntöä.

Kuva 19. Dota 2 Workshop – Character Art Guide [15]

Gradientilla voidaan myös luoda illuusio ambient okkluusiosta. Ambient okkluusio tarkoittaa eräänlaista valoisuuden tukkeumaa. Kuvasta 20 nähdään, miten esimerkissä on käytetty gradient-

tia varjoisasta osasta valoisaan. Tämän havainnon perusteella päädytään käyttämään kolmea väripistettä per gradientti, yksi valotetulle alueelle, yksi perussävylle ja yksi varjossa olevalle alueelle. Liukuväriä voidaan käyttää ambient okkluusion sijasta tai sen tavalla. Ambient okkluusiota käytetään usein leivotun valon sijasta tai syventämään valonkäyttäytymistä perimissä paikoissa. Myös vanhemmissa peleissä käytettiin tummennettua aluetta myös hahmojen varjostamisessa, kun sitä nykyään käytetään enemmän staattisten objektien kanssa.



Kuva 20. Esimerkkikuva liukuvärin suunnasta

6 Työkalun suunnittelu

6.1 Perusta

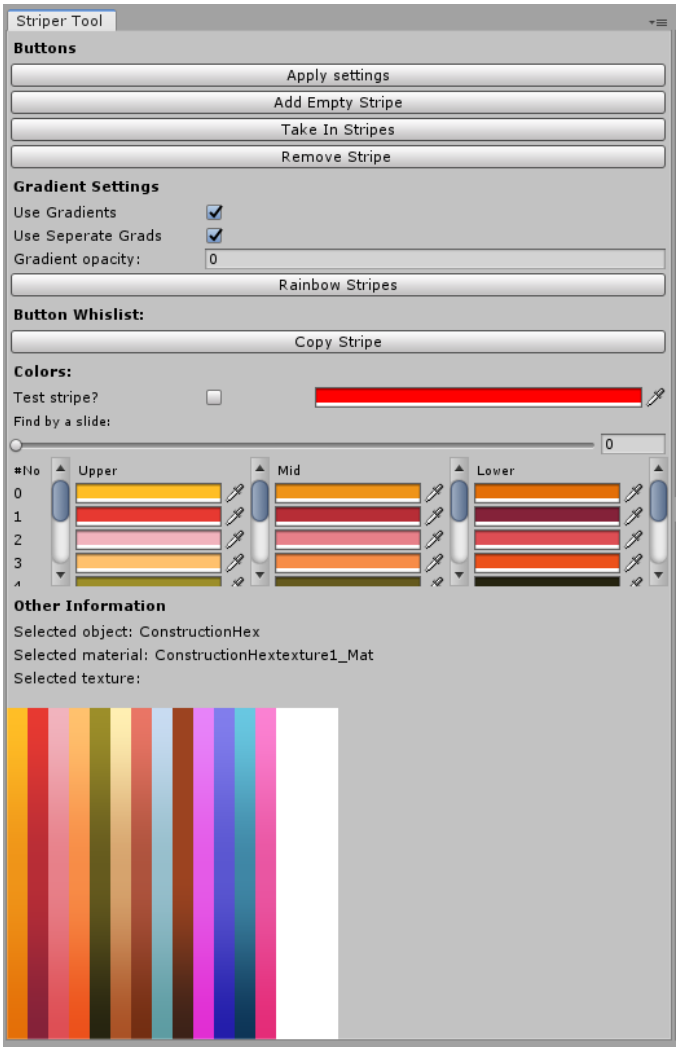
Työkalun suunnittelu lähti liikkeelle siitä, että pelille tarvittiin tunnistettava taidesuunta. Ajatuksena oli kuvan 21 The Mean Greenies -pelin näkemisen jälkeen tehdä myös pelin teksturointitavasta leikkisämpi ja erottuvampi. Perusajatuksessa oli myös mukana teksturoinnin tehokkuus ja helppous. Ajatuksena oli, että teksturointia lähdettäisiin toteuttamaan joko yksittäiselle objektille, kentän osalle tai kokonaiselle pelille. Tässä päädyttiin valitsemaan yksittäinen kentän saareke per tekstuuri, kun kyseessä oli RTS eli real time strategy -tyylinen peli ja se käytti kenttänään heksagonin muotoisia paloja. Gradienttien määräksi valikoitui 16, koska aikaisempaa kokemusta ei ollut ja haluttiin pelata varman päälle. Työkalun myöhemmässä jatkokehityksessä gradienttien määrästä tuli kuitenkin modulaarinen.

Kuva 21. Kuvakaappaus The Mean Greens -pelistä [16]

Alussa työkalun perusominaisuuksina oli antaa kolme väriarvoa kullekin kuudelletoista gradientille. Lisäominaisuuksiksi tuli värien sisäänotto ja gradienttien siirto keskenään. Gradienttien sisäänotto on erityisen tärkeä siitä syystä, että gradienttitekstuureita pystytään muokkaamaan, ilman erillistä tietokantaa väriarvoista. Alun perin gradienttien sisäänotto toimi siten, että kunkin tekstuurin kohdalla piti käydä asettamassa kansiorakenteen kautta ”readable”-totuusarvo [17]

tekstuuriasetuksista päälle kullekin erikseen, mikä taas hidasti työskentelyä huomattavasti sekä latasi sen prosessorin muistiin toistamiseen, jos sitä ei kytkenyt pois päältä.

Kuvan 22 versiossa nähdään ylhäällä peruspainikkeet, keskellä muutamia yksittäisiä asetuksia, väriarvojen listaus kolmessa vierekkäisessä rivissä, jotain tietoja valitusta peliobjektista ja alhaalla itse albedotekstuuri.



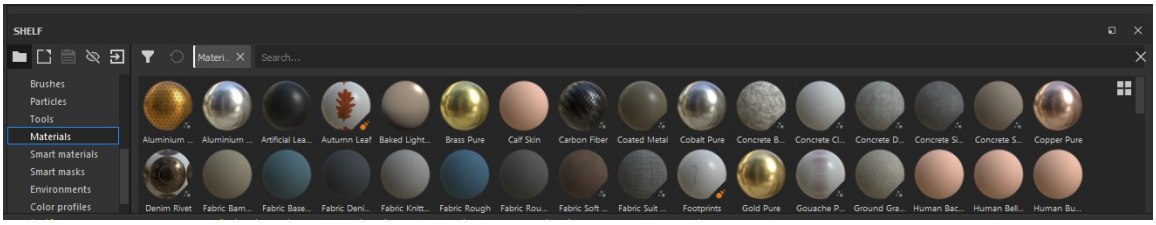
Kuva 22. Vanha Toy Thieves -pelin aikainen gradienttityökalu

6.2 PBR

Ajatus PBR:stä eli fysiikkaan perustuvasta rendauksesta tuli mieleen vasta myöhemmin, kun ajateltiin nykyaikaisia teksturointimenetelmiä enemmän. Inspiraationa toimivat lähinnä Substance Paintter-teksturointiohjelma sekä Unityn HDR-materiaali hierarkia.

Esiasetuksia oli ajatuksena tehdä kahdenlaisia sekä kokonaisia PBR-gradienttiesiasetuksia että yksittäisten pisteiden PBR-esiasetuksia. Yksittäinen piste tulisi siis sisältämään albedon lisäksi esimerkiksi metallisuuteen liittyvän arvon. Esimerkiksi kullalle voitaisiin tehdä oma esiasetus tallentamalla albedoon jokin vaaleahko ruskean sävy, keskiharmaa karkeus ja vaalea metallisuus. Kokonaisen gradientin esiasetus voisi olla, esimerkiksi liekin sävyt tyvestä yläosaan. Keltaisen, oranssin ja punaisen lisäksi tähän voitaisiin yhdistää emission värejä, jotta liekin valoneritys tulee ilmi. Esiasetusten arvoihin voi hakea suuntaa esimerkiksi Unity-dokumentaation omista materiaalikaavioista etsimällä ”material charts” [18]. Esimerkkejä löytyy sekä heijastus- että metallisesta teksturointityötavasta, joille kumpaisellekin löytyy Vakio Shader Unitystä.

Kuvan 23 Substance Painterin materiaaliesiasetuksista näkyy, että niissä on usein mukana myös tekstuuri. Tämä näkymä tulee olemaan todennäköisesti myös lähellä työkalun gradientti- ja kuviotekstuuriesiasetuksia, joissa gradientti- ja kuviotekstuuri ovat yhdessä, mutta sinne asti tämän opinnäytetyön aikana ei päästä.



Kuva 23. Self -ali-ikkuna Substance Painter -ohjelmassa

7 Työkalun tekeminen Unity pelimoottorissa

Unityn omilta sivuilta löytyvät hyvät tutoriaalit Unityyn liittyen ja heidän kotisivunsa kannustaa oppimaan heidän ”Unity Learn” -osiossa. Itse tehty Unity -työkalu on helppo aloittaa perimällä ”EditorWindow” -luokka ja sijoittamalla tulevan käyttöliittymän omaava kooditiedosto kansiorakenteessa ”Assets/Editor/” -osoitteen alaisuuteen. Tämän jälkeen voidaan määrittää ”[MenuItem("Window/My Window")]” -kohta koodia, joka antaa mahdollisuuden liittää työkalu Unityn työkalupalkkiin.

Esimerkiksi artisteilla tai suunnittelijoilla on suhteellisen helppo lähteä toteuttamaan käyttöliittymää EditorWindow-luokan alaluokilla, joita ovat GUILayout ja EditorGUILayout. GUI ja EditorGUI ovat vastaavat luokat, jos ei halua käyttää Unityn omaa asettelua. Nämä luokat kuuluvat IMGUI-systeemiin, jota tässä opinnäytetyössä käytetään.

Näiden lisäksi Unityyn on lisätty uusi UI Toolkit -niminen UI-systeemi, joka on muun muassa HTML:n, XML:n ja CSS:n innoittama. Unity kuitenkin suosittelee vielä pitäytymään vanhoissa UI-systeemeissä, sillä uudesta puuttuu vielä joitain ominaisuuksia, jotka löytyvät kahdesta muusta UI systeemistä. Tulevaisuudessa oman työkalun UI:n asettelun muuttaminen kyseiseen systeemiin on suhteellisen helppoa, koska koodi on jaettu erillisiin ”piirtoluokkiin”, joihin on kuhunkin laitettu Rect-arvo syötteeksi nimellä ”areaRect”, joka mukailee Unityn GUILayout.BeginArea-funktiota. Tämä on myös järkevää koodin toistamisen kannalta, sillä esiasetusikkunoita käytetään kahdessa paikkaa, väriarvovalikossa sekä esiasetusvalikossa. Lisäksi Unity tukee vielä vanhaa uGUI -pakettia.

7.1 Työkalun toteutus

Kun luodaan työkalua, on hyvä lähtökohta tutustua Unityn oman UnityEditor-nimiavaruuden Editor Window-luokkaan Unityn omalla ”Scripting API” eli ohjelmointirajapinta-sivustolla. Tämä luokka sisältää kaiken tarvittavan editorityökalun käyttöliittymän luomiseen ja piirtämiseen Unityn puolella. Editorityökalun käyttö tapahtuu itse Unityn pelitilan ulkopuolella, joten lisätään koodin alkuun ”#if UNITY_EDITOR”-ehto, jotta Unity tietää pyörittää sitä vain Unityn editorin puolella.

Seuraavaksi mietitään itse koodin rakennetta ja jaetaan se mahdollisen loogiseen ja uudelleen käytettävään muotoon. Työkaluun on tulossa niin sanotut ”Basic” eli perus- ja ”Advanced” eli kehittynyt tila. Niissä kummassakin on kuitenkin yhteisinä välilehtinä ”Values” eli tekstuurin tämänhetkiset väriarvot sekä ”Settings” eli asetukset, ”Presets” eli esiasetukset ja ”GradientMover” eli gradienttien liikutin osiot. Aloitetaan koodaaminen luomalla nämä välilehdet omiksi osioikseen.

Lähdetään liikkeelle siitä, että lisätään ”using”-direktiiveihin UnityEnginen lisäksi UnityEditor. Näin voidaan periä UnityEditor- nimiavaruuden kautta EditorWindow-luokka. Tämän luokan tilalla on yleensä MonoBehavior -luokka itse pelin toiminnallisuutta koodatessa, mutta tässä tapauksessa keskitytään editorin puolen koodaukseen. Itse EditorWindow-luokka periytyy Unityn omasta ScriptableObject-luokasta. ScriptableObject-luokkaa käytetään yleensä, kun tietoja talletetaan ja tallennetaan Editor-istunnon aikana.

```
using System;
using UnityEngine;
using UnityEditor;

#if UNITY_EDITOR
public class My_PBR_Gradient_Tool : EditorWindow
{

}
```

Koodiesimerkki 1. Editori-ikkunakoodin alustus

Init-luokka eli alustusluokka ajetaan nimensä mukaisesti läpi alustusvaiheessa. Itse ikkunan luontiin tarvitaan ”MenuItem()”-attribuuttia, jolla saadaan ikkuna liitettyä Unityn yläpaneelin valikkorakenteeseen ja saadaan se myös näkyviin valitsemalla se sitä kautta.

```
public class My_PBR_Gradient_Tool : EditorWindow
{
[MenuItem("Window/My_PBR_Gradient_Tool")]

        static void Init()
        {
        // Hae olemassa oleva avoin ikkuna tai jos ei ole tee uusi
        My_PBR_Gradient_Tool window = (My_PBR_Gradient_Tool)EditorWindow.GetWin-
        dow(typeof(My_PBR_Gradient_Tool));
        window.Show();
        }
}
```

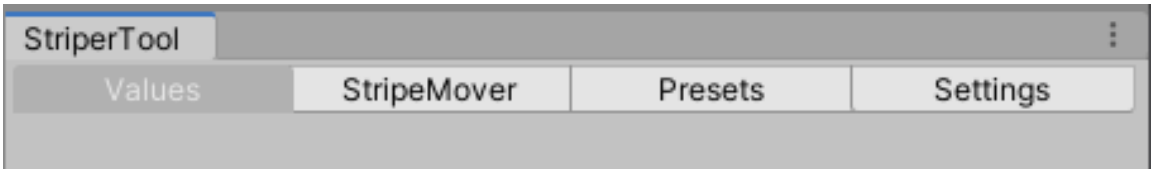
Koodiesimerkki 2. Init-vaiheen koodia

Tällä hetkellä on siis olemassa tyhjä editori-ikkuna ja seuraavaksi keskitytään luomaan itse alavälilehdet. Niiden luontiin tarvitaan OnGUI-toimintoa, jolla piirretään kaikki mahdolliset UI-elementit ruudulle sekä merkkijonolistaa, johon kirjoitetaan kunkin välilehden nimi. Listan sisällä liikkuminen toimii tässä esimerkissä switch case-rakenteen avulla. Tämän lisäksi pitää tallentaa nykyinen alavälilehdet-valinta kokonaislukuarvolla koodiesimerkki 3 alkuosaan.

```
int currentSubtab = 0;
int previousSubtab = 0;

string[] subtabs = { "Values", "Mover", "Presets", "Settings" };

public enum GUIState { mainMenu, differentMenu }
public GUIState state;
```

Koodiesimerkki 3.

Käytetään OnGUI -funktion sisällä GUILayout.BeginHorizontal() -toimintoa, jolla voidaan luoda horisontaalinen painikejono. Tallennetaan nykyinen välilehti ”currentSubtab” -kokonaislukuarvoon GUILayout.Toolbar -komennon kautta. Sen jälkeen käytetään GUILayout.EndHorizontal() -komentoa, jolla suljetaan komentoketju. Tämän jälkeen tehdään erillinen switch-case välilehdille niiden piirtämistä varten, syöttäen siihen currentSubtab-kokonaisluku.

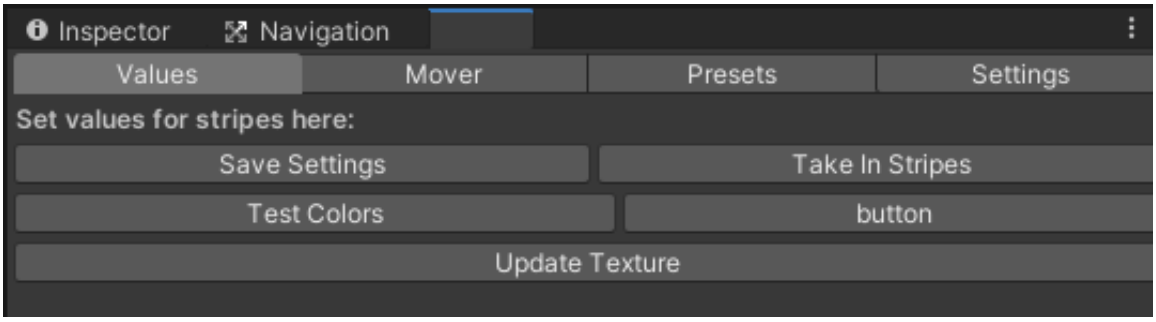


Kuva 24. GUILayoutin Toolbar -alatoiminto käytössä

```
private void OnGUI()
{
	GUILayout.BeginHorizontal(GUILayout.Width(position.width));

	currentSubtab = GUILayout.Toolbar(currentSubtab, subtabs, GUILayout.Width(posi-
	tion.width));

	GUILayout.EndHorizontal();

	Rect areaRect = new Rect ();

	switch (currentSubtab)
	{
		case 0:
		 DrawValuesMenu(areaRect);
		break;

		case 1:
		DrawMoverMenu(areaRect);
		 break;

		case 2:
		 DrawPresetsMenu(areaRect);
		break;

		case 3:
		 DrawSettingsMenu(areaRect);
		break;
	}
}
```

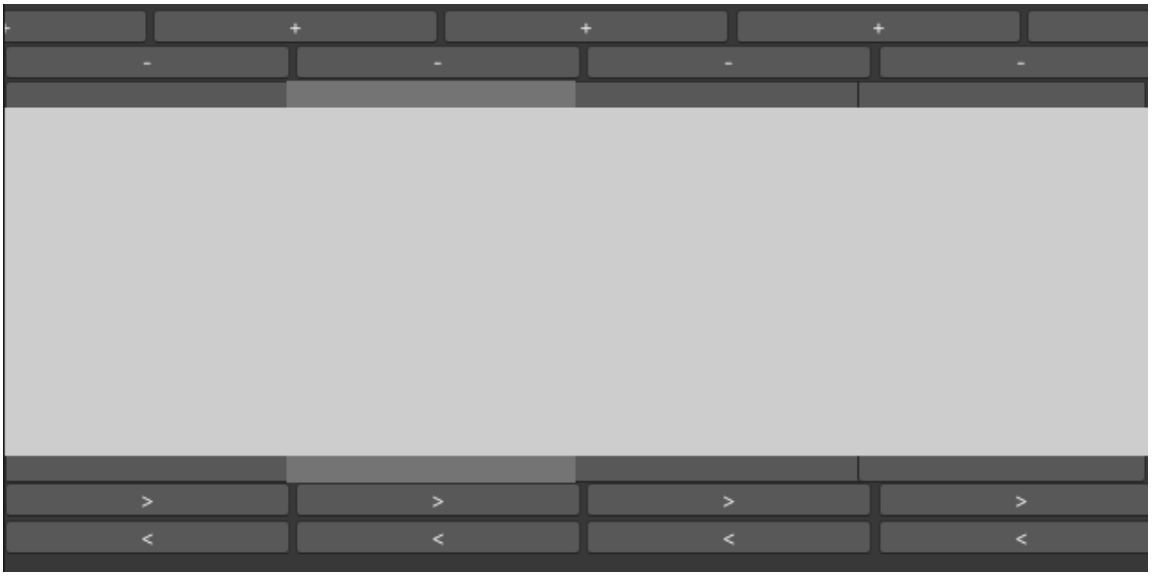
Koodiesimerkki 4. Esimerkki välilehtien piirrosta

Asetetaan seuraavaksi UI-elementit niille mietityille paikoille. Tämä tapahtuu käyttäen GUI-Layout- ja EditorGUI-toimintoja. Painikkeille toiminnallisuutta saadaan lisäämällä koodia if-lausekkeen kaarisulkeiden sisään.



Kuva 25. ”Values” eli väriarvovälilehden yläosa

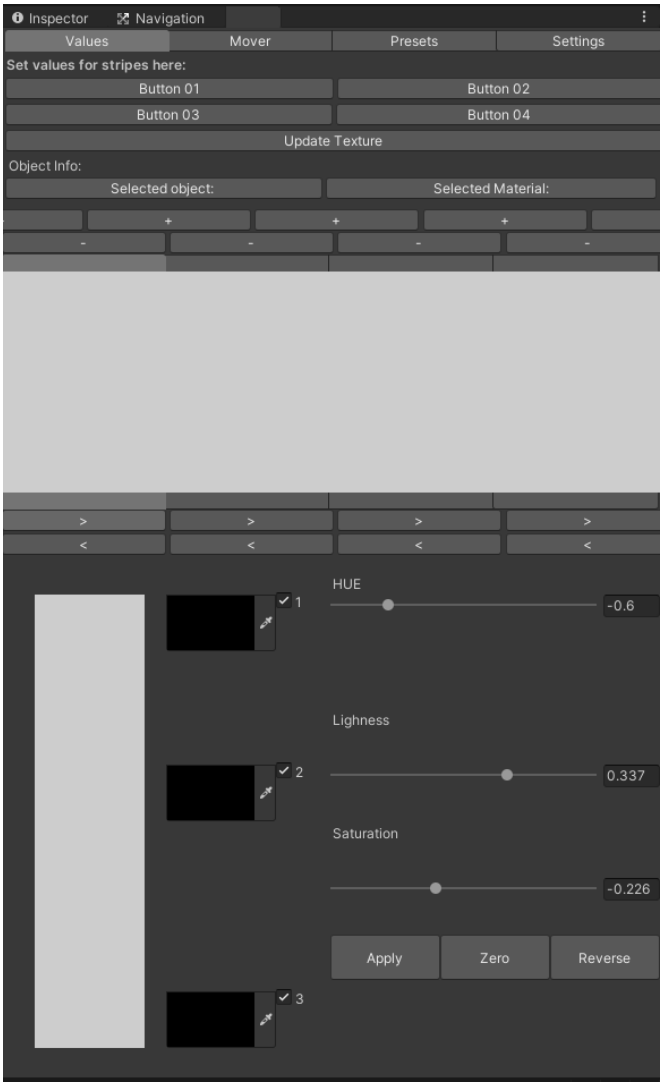
Seuraavaksi tehdään Values-välilehden seuraava osio, jossa on esikatselu valitun peliobjektin tekstuurista. Tässä tilanteessa voitaisiin tehdä kunkin gradientin alle oma kytkinelementti, mutta piirretään näppäinjono GUI.Button-komennoilla kuvan taakse, jotta valinnasta tulisi miellyttävämpää ja käytetään niin sanottu hukkatila hyödyksi. Kuvasta näkyy myös gradienttien lisäykseen tarvittavat napit, jotka on toteutettu siten, että on laitettu GUILayout.BeginHorizontal-toiminnon sisälle näppäimiä toistorakenteen avulla gradienttien määrän lisäksi yksi ja on annettu sille puolikkaan näppäimen verran siirrosta. Sitä seuraavana nähdään miinusmerkkinen poistorivi, joka on myös toteutettu samalla tavalla paitsi ilman siirrosta. Kuvan 26 esikatselun alla olevat gradienttiensiirtonapit on myös toteutettu samalla tavalla. Missään näistä napeista ei vielä tosin ole funktionaalisuutta. Funktionaalisuus lisätään myöhemmin.



Kuva 26. ”Values”-välilehden keskiosa

Seuraavaksi tehdään Values-välilehden kolmas osa. Siihen tulee valitun gradientin väriarvot sekä mahdolliset väriarvojen säädöt. Tämä osuus vaatii myös erillisen yksilöllisen gradienttitekstuurin, joka piirretään äsken luodun yllä olevan gradientinvalintapainikeryhmän valinnan perusteella kaksiulotteisesta väritaulukosta. Tässä tapauksessa gradientObject.albedo_col[,]-taulukosta. Basic-tilassa tähän tarvitaan vain tietty määrä väripisteitä.

Kuvassa 27 nähdään työkalun käyttöliittymä, joka on jaettuna kolmeen osaan, joita voidaan myöhemmin helposti liikutella tarvittaessa. Ylhäälle tulee erilaiset kokoryhmään liittyvät komennot painikkeiden muodossa sekä sen hetkisen valitun objektin tiedot. Keskiosa on omistettu gradienttiryhmän esikatselulle, gradienttien lisäämiselle ja poistolle sekä niiden liikuttelulle. Alaosassa on kulloinkin valitun gradientin tietoja sekä niiden muokkaamisen tarvittavia työkaluja.



Kuva 27. ”Values”-valikon näkymä

Seuraavaksi perehdytään kuvan renderöintiin annetuin väriarvoin. Perusmoodiin meillä on tarkoitus luoda rajattu ryhmä väriarvoja, joiden avulla luodaan gradientteja. Aloitetaan luomalla muutamia muuttujia, joita tarvitaan tämän toteutuksessa, kuten värilista ja gradienttien lukumäärä kokonaislukuarvona. Nimetään lista ”albedoksi” tulevaa PBR-kehitystä silmällä pitäen. Samalla voidaan ajatella käytettävän kaksiulotteista taulukkoa, koska tällä hetkellä on olemassa lista gradientteja, joilla kuillakin on omat väriarvonsa. Seuraavaksi siirrytään värien renderöintiin.

```
int gradientCount = 4;
int pointCount = 3;

Color[,] albedo_col = new Color[4, 3];
int resolution = 64;
Texture2D albedo_tex = new Texture2D(64, 64);
bool autoUpdate = true;

Color[] palette_rainbow = new Color[] { new Color(225, 0, 0), new Color(225, 127, 0), new
Color(225, 255, 0), new Color(0, 255, 0), new Color(0, 0, 255), new Color(75, 0, 130), new
Color(143, 0, 255) };
```

Koodiesimerkki 5.

Luodaan aliluokka, jonka avulla kirjoitetaan annetulle tekstuurille värit. Lähdetään liikkeelle satunnaisesti annetuista väriarvoista itse toimivuuden testaamiseksi ja jotta voidaan käyttää tätä aina työkalua avatessa ennen kuin sisään on annettu vielä mitään arvoja. Lisäksi tämä helpottaa myöhemmin lisäämään useamman pisteen esiasetuksen arvot tietyn väriselle alueelle.

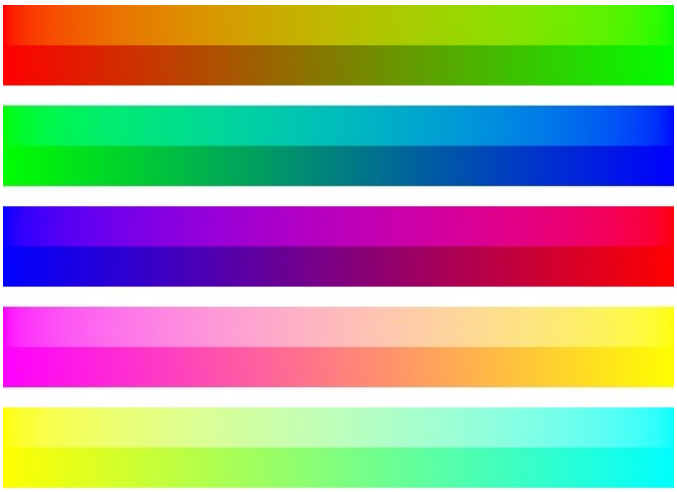
Lähdetään liikkeelle neljällä gradientilla ja kolmella pisteellä. Kolmen väripisteen lukumäärä määrittyy aiemmin keskusteltuun valosta johtuvaan lukumäärään, yksi valaistulle alueelle, yksi pohjavärille ja yksi pimennossa olevalle alueelle. Laitetaan vakiotekstuurin kooksi 64 kertaa 64 pikseliä, koska Joyce on todennut sen hyväksi [2, 0:22]. Tämän testaamiseen käytetään ”Rainbow”-nappia, jonka avulla lisätään albedo_Tex-listaan ”palette_rainbow” -listan väriarvot. Syötetään arvot tehtyyn FillWithFade()-luokkaan, joka taas käyttää MixFade -luokkaa laskutoimitusten suorittamisessa. Lisätään myös FadeColors arvot ”Settings”-välilehdelle. Näiden arvojen avulla lasketaan uudet arvot sekoittaen niitä valoisaan ja pimeään arvoon.

Käytössä olevan teksturointitavan takia ei käytetä MipMapsejä. MipMapsejä käytetään yleensä peleissä, kun kamera liikkuu kauemmaksi itse peliobjektista. Tällöin MipMapit vaihtuu pienempiresoluutioiseen versioon. Teksturointitavan takia käytetään pieniresoluutioisia tekstuureja ja tämä lisäisi tekstuurivuotoja, kun tekstuurit näyttäytyisivät vielä pienempinä. MipMapit generoidaan yleensä automaattisesti Play Mode-tilan ulkopuolella. Mipmapsien käyttö on rinnastettavissa LOD eli Level Of Detail eli 3D-mallien tarkkuustasoon.

Mipmapsit saadaan pois käytöstä manuaalisesti valitsemalla tekstuuritiedosto projektikansiosta ja klikkaamalla ”Generate Mip Maps”-totuusarvoa Inspector-välilehdeltä. Tai sitten työkalun sisällä kun luodaan tekstuuria kuten ”new Texture2D(int width, int height, TextureFormat textureFormat, bool mipChain)” säädetään mipChain totuusarvo epätodeksi.

Kirjoitetaan tekstuuriin väriarvot käyttämällä sisäkkäisiä for looppeja ja Texture2D.SetPixel() -komentoa. Tekstuuriin väriarvojen kirjoituksen jälkeen on tärkeää asettaa ne tekstuuriin Texture2D.Apply() -komennolla. Sen jälkeen se muunnetaan bittimuotoon Texture2D.EncodeToPNG() -komennolla. Tallennetaan tekstuuritiedosto kansiorakenteeseen System.File.WriteAllBytes-toiminnon avulla syöttäen siihen juuri talteen otetut bitit. Tärkeää tässä on muistaa käyttää tallennussijaintina koko käyttöjärjestelmän tiedostopolkua, sillä kyse on System direktiivin alaisesta toiminnosta, eikä Unityn omasta, jolloin se vaatisi vain Unity-projektin sisäisen tiedostopolun. Lopuksi pitää muistaa päivittää Unityn assettitietokanta käyttäen AssetData-base.Refresh-komentoa.

Kuvassa 28 nähdään, miten gradienttiparien alemmat siirtymät eivät tapahdu luonnollisesti vaan suttaantuvat. Tämä johtuu siitä, että lineaarisessa siirtymässä ei oteta huomioon sitä, miten valo käyttäytyy oikeassa elämässä. Tosielämän tilanteessa näillä kuillakin väreillä on omat aallonpituutensa, jotka vaikuttavat kunkin värin sekoittumiseen [19].



Kuva 28. Gradienttipareina gammakorjatut liukuvärit sekä alempana lineaariset liukuvärit [20]

Tämä voidaan ratkaista käyttämällä joko olemassa olevia kaavoja erottaen kunkin värikanavan omakseen tai käyttää C#:n valmista GammaCorrection-funktiota. Lisäksi tarvitaan lisätty totuusarvo, jotta voidaan antaa käyttäjälle valinta käyttääkö kyseistä korjausta vai ei.

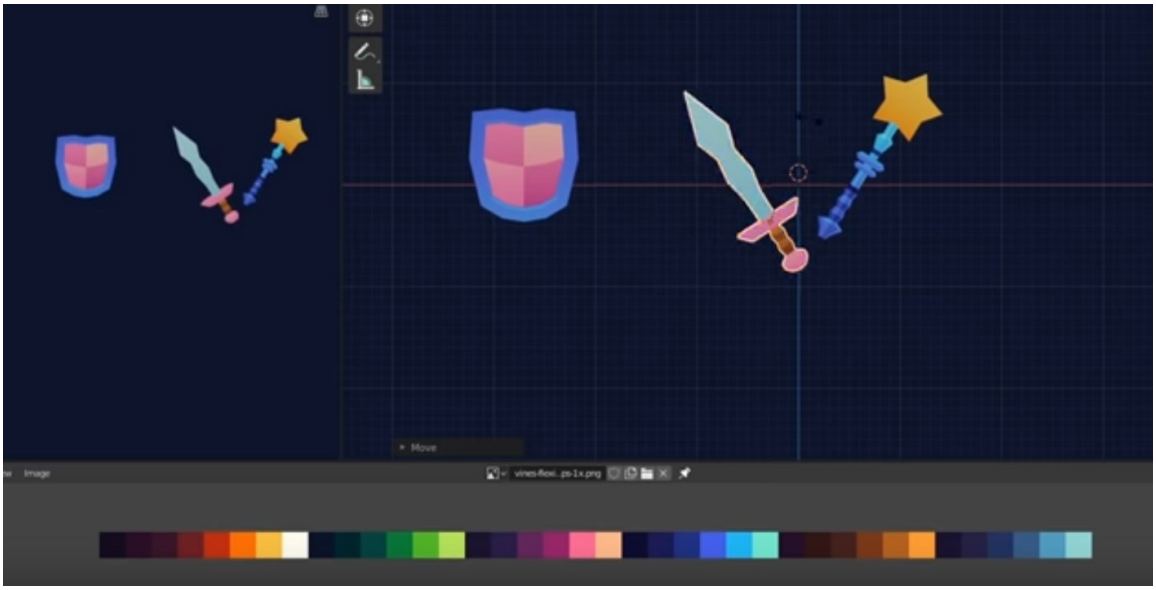
”Basic”-moodissa gradienttien sisäänotto toteutetaan siten, että luodaan väliaikainen tekstuuri luettavasta tekstuurista Unityn oman Renderer-luokan kautta. Tällöin vältetään tekstuurin ”readable”-totuusarvonmuuttujan manuaalisen muuttamisen Unity-editorissa joka tekstuurin kohdalla. Lisäksi luettavassa muodossa olevat tekstuurit vievät enemmän muistia pelin ollessa käynnissä [25]. Heijastaminen renderöijän kautta on toteutettu seuraavalla tavalla: Sen sijaan että yritettäisiin lukea väriarvoja suoraan tekstuuritiedostosta, voidaan käyttää hyödyksi RenderTexture-muuttuja tyyppiä. Resoluutio on hyvä olla 64 kertaa 64 pikseliä, Joycen [2, 0:22] mukaan. Jos resoluution menee alle 32 kertaa 32 pikselin resoluutiolla, silloin voi tulla värivuotoja.

Tekstuurien asetuksista ”generate Mip Maps”-totuusarvo pois. Yleensä Mip Mapsien käyttötarkoitus on suurempien resoluution peleissä. Vaikka itse kiintolevylle luotujen tekstuurien määrä ja niiden viemä tila olisi suurempi, se vie vähemmän renderöintikapasiteettia per tekstuuri pelin suoritusaikana. Tämä aiheuttaa gradienttityötavan kanssa alhaisen resoluution tavoin värivuotoa ja kun kyseessä on erittäin alhainen tekstuuriresoluutio, niin värit sekoittuvat toisiinsa. MipMaps on LOD:in eli level of detailin tapainen systeemi, jonka kautta pelieditori luo tekstuureista eri resoluutioisia versioita ja käyttää niitä sitä mukaan, kuinka kaukana kamera on tekstuurin omaavasta 3D-mallista. MipMapsien säädön lisäksi on tärkeää kiinnittää huomiota tekstuurin suodatustilaan ja se on hyvä säätää pisteentarkaksi eli ei filtteröintiä ollenkaan.

Gradienttien siirtelyä varten on tehty rivit ”>”- ja ”<”-nappeja kunkin gradientin kohdalle. Kuvan 29 nappeja painaessa gradienttilistojen järjestys muuttuu haluttuun suuntaan. Tämän jälkeen tekstuuri renderöidään uudestaan.

Kuva 29. Työkalun liikuttelurivi

Kuvan 30 videon katsottua hoksataan lisätä työkaluun vielä mahdollisuus renderöidä gradientteja peräkkäin. Sen sijaan että työkalulla rendattaisiin ainoastaan 1:1 kuvasuhteen tekstuureja, jolloin aiheutuu hukkatilaa, voidaan gradientit rendata peräkkäin. Mahdollista värivuotoa tapahtuisi täten vain yhdellä akselilla.



Kuva 30. Kuvakaappaus YouTube videosta käyttäjältä MortMort [23]

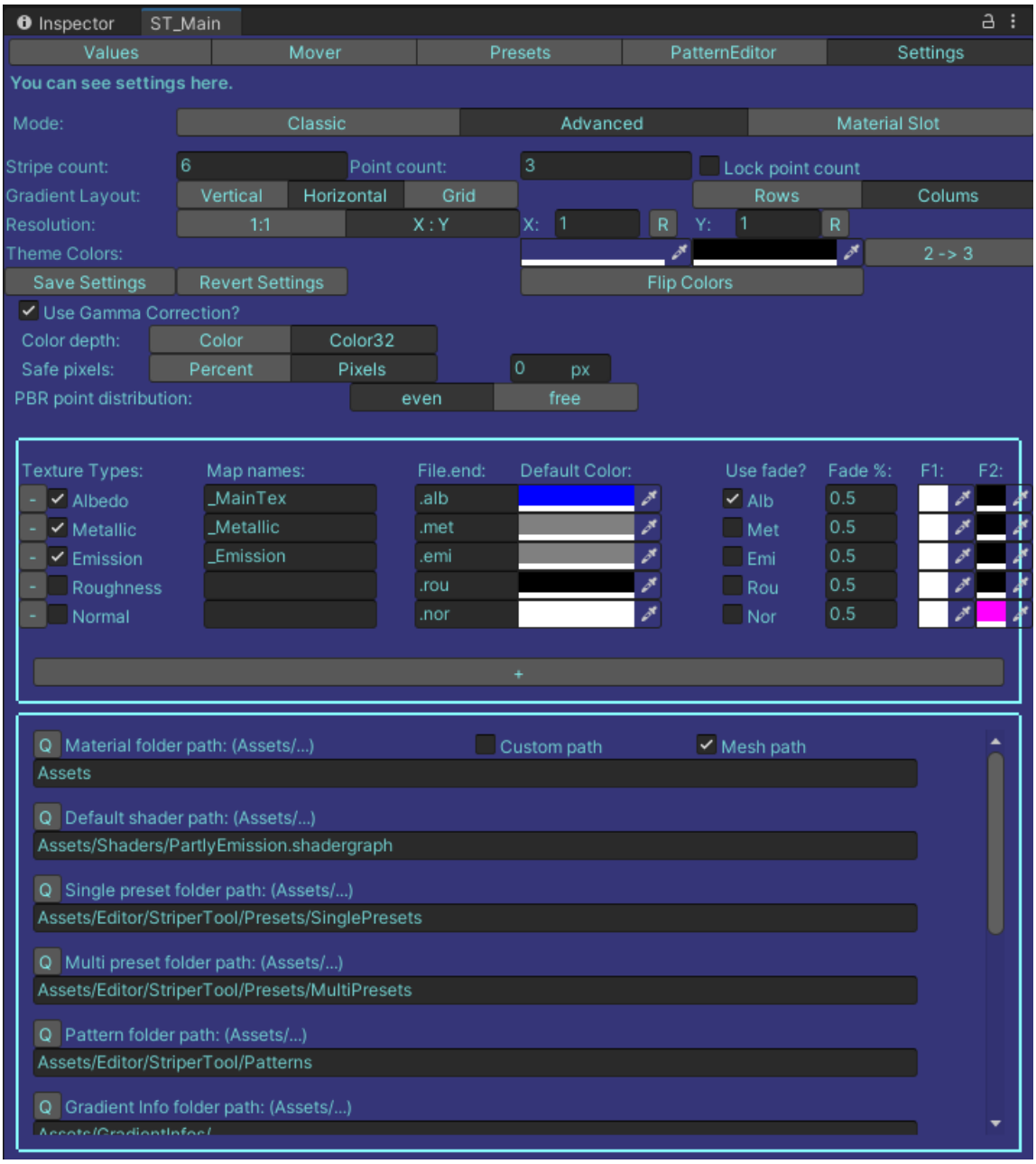
Työkalun asetukset tallennetaan erillisen omakirjoittaman SaveSystem.cs-komentosarjan kautta. Kooditiedosto sisältää LoadSettings() ja SaveSettings()-funktiot. Tallennettaessa SaveSettings()-funktioon syötetään ”GradientData”-niminen serialisoitu luokka, joka pitää sisällään tallettavat muuttujat JSON-muotoon käännettävässä muodossa. Asetuksiin kuuluvat työkalun teemavärit, resoluutio ja gradienttien määrä. Tieto tallentui aiemmin binäärimuodossa, mutta myöhemmin se muuttui avoimempaan JSON-muotoon, kun sitä käytettiin myös esiasetusten tallentamisessa hyödyksi. ”Auto detect”-vaihtoehdot kuuluvat Classic-tilaan ja niistä voidaan valita, missä muodossa gradienttien lukumäärä tallennetaan materiaalin nimeen. Safe pixels -muuttuja pidättää pikseleitä gradientin kumpaankin päähän, päädyissä olevilla väriarvoilla, täten välttäen värivuotoa.

Kuvassa 31 näkyy myös ”Used Trexture Types”-osio, joka on saanut vaikutteita Substance Painterin Output Templates – osiosta, joka näkyy kuvasta 32. Osion tiedot voidaan jatkossa tallettaa suhteellisen vaivatta erilliseen JSON-tiedostoon, jotta jokaiseen projektiin tai jokaiselle käyttäjälle saataisiin samanlaiset tai erilaiset asetukset riippuen tilanteesta.

Kuvassa näkyy myös ”path area” eli tiedostopolkualue. Se sisältää muun muassa materiaalikansiokohdan, jossa on valittavissa joko niin, että uuden materiaalin sijainti määrittyy valitun 3D-mallin mukaan tai että se käyttää kohtaan syötettyä mukautettua tiedostopolkua, johon kaikki materiaalit menevät, jos sellainen projektinhallinta sattuu olemaan käytössä.

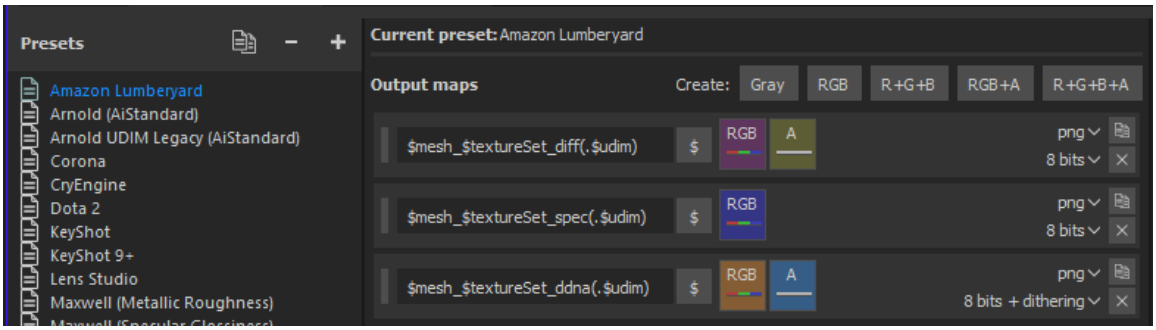
Seuraavassa kohdassa on perus-shaderin kohta. Omassa tapauksessa siinä on Shader Graphissä eli Unityn omassa node-pohjaisessa shaderieditorissa tehty shaderi, joka omaa muun muassa ”Used Texture Types”-osiossa määritellyt tekstuureiden nimet, kuten ”_MainTex”. Tulevaisuudessa kyseiseen shaderiin voitaisiin laittaa myös kuviotekstuureihin liittyviä muuttujia.

Lisäksi se sisältää yksittäis- ja useamman pisteen esiasetusten tiedostopolut, kuviotekstuurien kansiopolun ja gradientti-infojen kansiopolun. Lisäksi siellä on tulevaisuutta varten gradienttien ja kuviotekstuurien sekoittamiseen liittyviä tiedostopolkukenttiä.



Kuva 31. Kuva nykyisestä ”Settings”-välilehdestä

On myös jatkon kannalta harkinnassa luoda itse kosmeettisille ominaisuuksille esimerkiksi teemaväreille oma tallennusväylänsä, jotta ne eivät sekoittuisi itse projektissa yleisesti käytettyjen asetusten kanssa ja joka tallennettaisiin esimerkiksi Unityn ”Preferences”-kansioon, kuten Unityn oman Gradient Editorin tapauksessa. Lisäksi olisi tärkeää luoda kullekin peliprojektille omat asetusesiasetukset, jotka tallennettaisiin itse projektikansioon.



Kuva 32. Substance Painterin Output Templates -välilehti

Valitun gradientin värien väriarvoa voidaan muuttaa yhtä aikaa erillisestä säätöpaneelista, jossa on HUE, värikylläisyys ja valoisuus. Tämä ominaisuus on vielä työn alla.

Itse tallennettavien esiasetusten idea tuli hieman myöhemmin. Aluksi oli vain tarkoituksena tehdä lista kovakoodattuja perusmateriaaleja, jotka toimisivat referensseinä, esimerkiksi puu, kulta ja kivi, mutta esiasetusten tallentaminen itse työkalun kautta osoittautui helpommaksi. Alkuperäisen idean mukaan liikkeelle lähdettiin yhden pisteen esiasetuksista ja niitä lähdettiin tallentamaan JSON-muodossa, työkalun mahdollista Blender-laajennusta ajatellen. Useamman väripisteen tallentaminen oli aluksi hieman mysteeri, mutta tässä tapauksessa päädyttiin kolmeen pisteeseen, kun se oli ollut jo pitkään käytössä ja se toimi hyvin valoisan, keskiarvon ja pimeän osan määrittäjänä. Jatkokehityksessä opinnäytetyön jälkeen tulee esille, miten eri väripistemäärän omaavat esiasetukset toimivat yhteen.

Yksittäisen gradientin ja esiasetusvalinnan välissä on niihin liittyviä painikkeita, kuten väripisteiden valintaan liittyvät napit. ”Flip Order”-nappi kääntää valitun gradientin väriarvojen järjestyksen. ”Load” ja ”Emissive” -napeissa on prosenttiluvut, jotka kertovat sekoitussuhteen. Load- nappia käyttäessä prosenttiluku vaikuttaa siihen, miten nykyinen gradientti ja ladattu gradientti sekoittuvat keskenään. ”Emissive”-napin prosentti taas kertoo sen, kuinka monta prosenttia kopioitavista albedo väreistä sekoittuu mustaan muodostaen gradientin emissiiviset arvot. ”Overwrite”-nappi päällekirjoittaa valitun esiasetuksen väriarvot, jättäen nimen ja tunnisteet ennalleen. ”Calc middle point”-painike toimii, ainakin kolmen väripisteen kohdalla, siten että se laskee ensimmäisen ja kolmannen pisteen avulla keskiarvon ja laittaa sen keskimmäisen pisteen arvoksi.

”Fade treatment”-nappi toimii siten, että se laskee keskimmäisen väriarvon perusteella gradientin ääripisteiden väriarvot, sekoittaen niitä ”texture types”-asetusten arvojen perusteella, kuten yksittäisten asetusten randomisointia käytettäessä.

Lisätty tekstuurityyppivalinta-alue, kuvan 33 vasemmassa yläkulmassa muuttaa ylhäällä gradientinvalinta-alueella näytettävää tekstuuria, valitun PBR-tekstuurityypin mukaan. Lisäksi se sisältää ”Edit only current type” -kytkinelementin, jonka kautta esiasetuksista voidaan ladata vain sillä hetkellä valitun tekstuurityypin arvoja tai vaihtaa keskenään vain kyseisen PBR-tekstuuri tyypin arvoja gradientinvalinta-alueella. Lisäksi se sisältää lukkojen hallintanäppäimet, joilla voidaan esimerkiksi lukita gradientteja, kun testataan uusia värejä Randomize-napin kautta ei lukituille gradienteille.

Kuva 33. Kuvassa vasemmalla on yksittäinen valittu gradientti. Keskellä komentorivi ja oikealla on erilaisia menuja muun muassa esiasetuksille

Tunnistesysteemiä lähdettiin työstämään esiasetuksille niiden määrän kasvaessa. Ajatuksena oli, että tämä systeemi voisi toimia katekorisointia paremmin, kun esiasetuksia ei rajoitettaisi kuuluumaan vain yhteen lokeroon. Lisäksi tunnisteallasidea tuli samaan aikaan, minkä avulla voitaisiin poimia lisää tunnisteita haluttaessa sillä hetkellä valitulle esiasetukselle. Tämän jälkeen myös tunnisteiden käyttämisestä työkalun aikaisemman version Rainbow()-napin tapaan tuli innostus

ja päätös jakaa tunnisteet kahteen eri ryhmään, jotta niitä voitaisiin käyttää omilla tavoillaan. Single-mode toimii siten, että työkalu ottaa tunnisteen perusteella yhden väripisteen esiasetuksista ryppään ja sekoittaa niitä asetusvälilehden ”fade” eli häivytysasetusten mukaan, kun taas multi-modessa värit tulevat suoraan useamman väripisteen esiasetuksista. Kuvassa 34 nähdään yksinkertaistettu versio satunnaistamiskäyttöliittymästä.

Kuva 34. Nykyinen gradienttien satunnaistamisrivi.

Tunnistesysteemi toimii tällä hetkellä siten, että kun esiasetukset ladataan työkaluun, kukin esiasetus jättää tunnisteeseen oman ID:eensä string-arvona. Ja sitten kun työkalu suodattaa esiasetuksia jollain tunnisteella, valitun tunnisteen references-arvoja verrataan yksitellen ladatun multiPresets_Loaded -esiasetuslistan ID-arvoihin ja lisätään osuman kohdanneet multiPresets_Filtered -listaan. Jatkokehityksessä tähänkin voi löytyä jokin parempi tapa, kuten C# referenssin käyttö suoraan ilman erillistä läpikäyntiä.

Tunnistealtaan vasemmasta yläreunasta löytyy tunnisteiden lukumäärä. Sitä seuraa valinta siitä, mitä esiasetuksia kulloinkin näytetään. Tällä hetkellä tunnisteet voi asettaa joko aakkos- tai tunnisteen referenssilukumäärän mukaiseen järjestykseen. Asetteluksi voi säätää joko kuvan 35 mukaisen allasnäkymän tai listanäkymän. Kussakin tunnisteessa on lista referenssiesiasetuksista kerättyjä yksittäisiä värejä, jotka näkyvät kunkin tunnisteen vasemmassa päädyssä. Keskellä on tunnisteen nimi ja viimeisenä suluissa referenssien määrä.

Kuva 35. Työkalun tunnisteallas

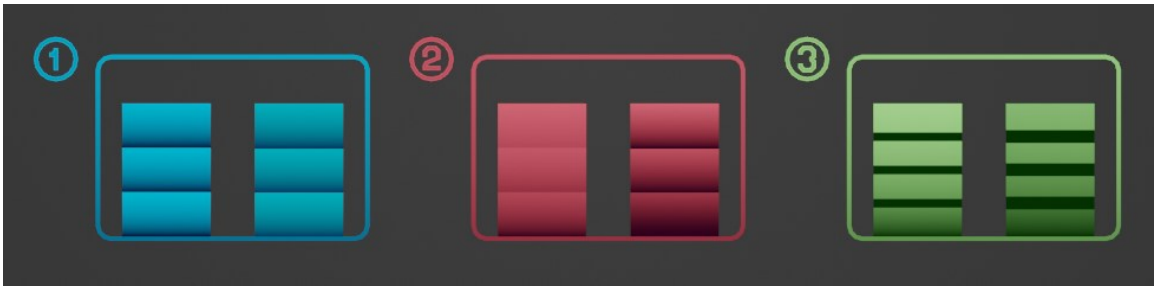
Nopein tapa hyödyntää työkalua Blenderin puolella on paikantaa työkalun luoma albedo tekstuuri Unity-projektin kansiorakenteesta. Ja aina kun tekstuuria päivitetään Unityn puolella, voidaan Blenderin puolella painaa Alt + R-näppäinyhdistelmää Image- tai UV-ikkuna aktiivisena, jolloin tekstuuri ladataan uudelleen Blenderin muistiin. Tulevaisuudessa, Blenderin Pythoniin laajennettaessa, olisi suhteellisen helppo kehittää nappi, josta eri tekstuurityypit saisi päivitettyä yhtaikaa esimerkiksi valitun objektin shaderin kautta.

7.2 Työkalun laajennus

Jatkossa voitaisiin työkaluun lisätä esimerkiksi muunninpino, jonka työkalu kävisi läpi gradientteja renderöidessä. Muunninpinon etuus on siinnä, että siihen päästään käsiksi itse työkaluikkunasta, eikä artistilla tarvitse mennä koskemaan koodiin muuttaakseen asioiden renderöimisjärjestystä. Kuten kuvasta 36 näkyy, että muuntimia on olemassa monta tyyppiä.

Kuva 36. Esimerkki Blenderin muunninpinosta (Blender 2.8)

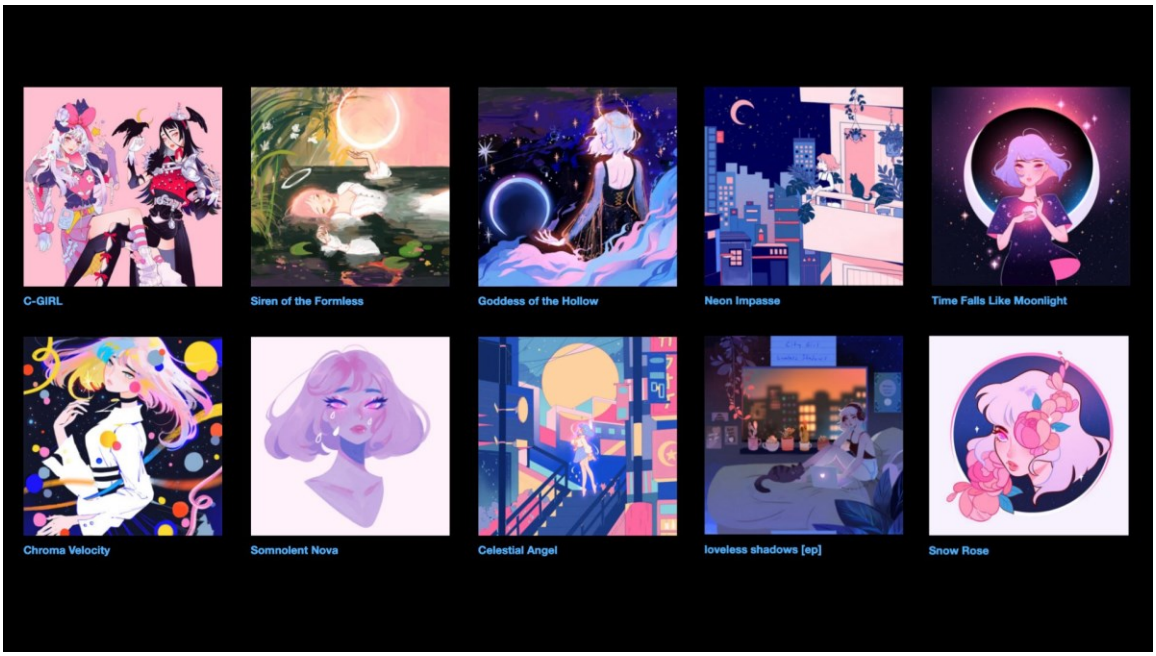
Kuvan 37 1-kohdassa näkyy miten yksinkertainen muunnin, joka rendaisi gradientteja tasaisin välein perusgradientin päälle, voisi ratkaista ongelman ylimääräisen topologian luonnissa. 2-kohdan muunnin edustaa toistuvaa gradienttia joka häilyy yläosasta. 3-kohdan muunnin on esimerkki muuntimesta, joka voisi rendata gradientin päälle raitoja tietyin välein. Esimerkit on toteutettu Blenderissä erilaisilla shadereillä.



Kuva 37. Gradienttimuunnin havainnistojia (Blender 2.8)

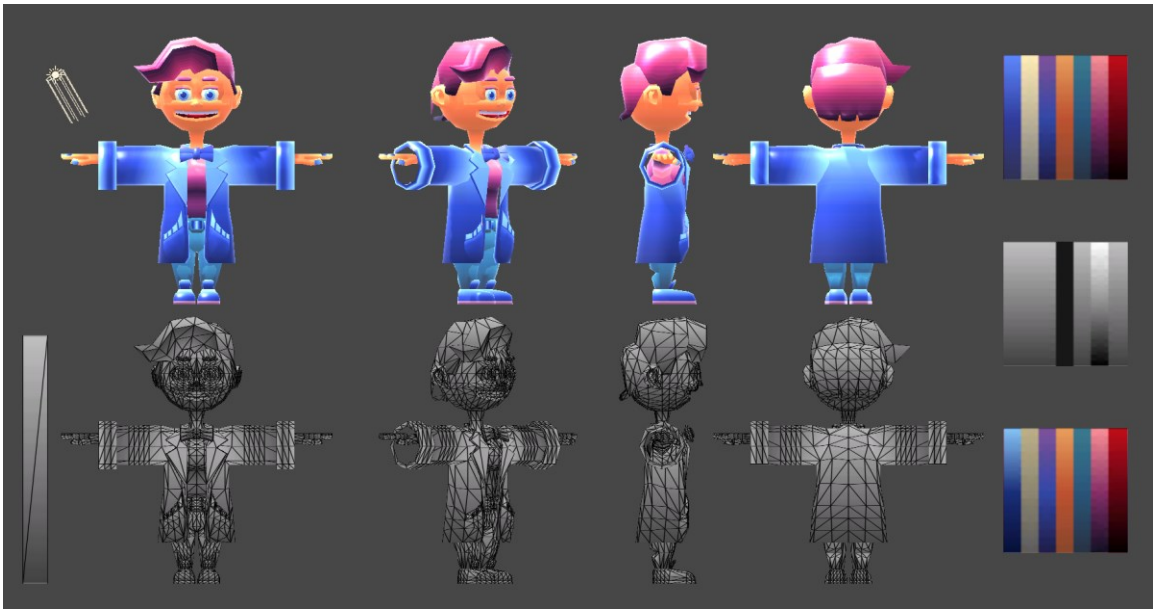
8 Työkalun käyttöesimerkkejä

Kuvissa 39–41 on gradienttiteksturoituja 3D-malleja Unity-pelimoottorissa. Kuhunkin materiaaliin on tallennettu kolme eri tekstuuria, jotka ovat albedo, metallinen ja emissioinen. Gradienttitekstuurit on luotu tämän opinnäytetyön aikana kehitetyllä työkalulla käyttäen siihen luotuja kolmen pisteen esiasetuksia. Kuvan 39 noidan väripaletti on kuvan 38 CityGirl musiikkiartistin innoittama.



Kuva 38. Kokoelma kaikista CityGirl-artistin albumitaiteista [26]

Alla nähdään omia esimerkkejä työkalun käytöstä. Kuvan 39 noitahahmossa nähdään gradienttisiirtymiä muuan muassa jaloissa alhaalta ylöspäin, samoin takin keskiosassa ja osassa hiuksia. Hiuksien etuosassa gradientti menee ylösalaisin korostaakseen aaltomuotoa. Ylemmän rivin vasemmassa laidassa nähdään Unityn Directional Lightin suunta. Alemmasta rivistä nähdään paremmin 3D-mallin topologia ja harmaasävyinen UV-avaruuden Y-akseli. Triangle count eli kolmioiden lukumäärä tässä hahmossa on yhteensä 6 881.



Kuva 39. Noitahahmo ja siihen liittyvät gradienttitekstuurit

Vihollishahmon jalan lihakset on teksturoitu korostaen niiden kolmiulotteista muotoa. Lisäksi yksittäisissä karvatupsuissa on käytetty gradientteja hyödyksi. Kolmioiden määrä tässä hahmossa on yhteensä 5032.

Kuva 40. Vihollishahmo ja siihen liittyvät gradienttitekstuurit

Kitaraan on luotu ilmeikkyyttä gradientein. Kitaran runko-osan alaosassa on haettu luutamaista varjostusta ja kitaran headstockissa eli lavassa on haettu puunrunkomaista muotoa. Gradientit saavat mikkien kiinnitysosat näyttämään läpikuultavalta muovilta. Kolmioiden määrä tässä kitarassa on yhteensä 3993.

Kuva 41. Kitaramalli ja siihen liittyvät gradienttitekstuurit

9 Työkalun eteenpäinvienti ja tulevaisuus

Kyseinen työtapa voitaisiin suorittaa myös shaderin avulla ja syöttää väriarvot itse materiaaliin ja voitaisiin renderöidä ja tuoda tekstuurit ulos vain silloin, kun niitä tarvitaan. Blenderin puoleinen työkalu helpottaisi työskentelyä ja toimisi ja välittäisi tietoa Unityn puolen työkalun kanssa. Voitaisiin esimerkiksi Blenderin puolella määrittää arvo, joka kertoo kuinka monta gradienttia kyseinen objekti käyttää, joka voisi sitten Unityyn 3D-mallin viemisen jälkeen kertoa valittaessa Unityn puolen työkalulle, kuinka monta gradienttia kyseinen objekti käyttää eikä meillä tarvitsisi manuaalisesti koskea kyseiseen kokonaislukuun tai lisätä uusia gradientteja. Blenderin puolella kyseinen työkalu kävisi ehkä enemmän järkeen. Kuitenkin oman koodausosaamisen rajoittuneisuuden takia päädyin tekemään työkalun Unityn puolella. Mutta taas esimerkiksi erilaisten tekstuurivariaatioiden luominen, kuten pelaajien tiimivärien tapauksessa, käy järkeen enemmän Unityn puolella.

Työkaluun on tulossa näillä näkymin kolme erilaista työtilaa. “Basic Mode/Classic Mode” eli perusmoodi, jossa työkalu toimii perinteisellä tavalla eli ottaen väritiedot sisään lukien ne gradienttitekstuurista perustuen tiedossa olevaan gradientti määrään, joka on tällä hetkellä tallennettuna materiaalin nimeen esimerkiksi “material[4]mat”-muodossa. Lisäksi tällä hetkellä väripisteiden kohta on lukittuna itse pisteiden määrään. Basic modessa kolme väripistettä on hyväksi todettu lukumäärä. Yksi valoisalle, yksi keskivärille ja yksi varjoisalle puolelle.

Työkalun “Advanced” eli edistyneessä työtilassa materiaalitiedoston lisäksi kansiorakenteeseen tallentuu JSON-tiedosto, joka voi gradienttimäärän lisäksi sisältää itse väriarvot kullekin tekstuurityypille, jolloin vältetään itse tekstuurin lukeminen väriarvojen sisäänotossa. Lisäksi kyseiseen tiedostoon voidaan tallentaa erilaisia muokkaimia, jotka vaikuttaisivat siihen, millä tavalla kyseinen gradientti rendataan ulos. Näitä muokkaimia voisi olla esimerkiksi jonkinlainen siirtymävaiheiden määrän määrittäjä, jossa itse siirtymän askeleita saataisiin näkyviin enemmän, joka voisi tuoda lisää ilmettä haluttuun taidetyyliin. Lisäksi voitaisiin määrittää kullekin väripisteelle oman kohdan esimerkiksi float-arvolla, jolloin päästään lähemmäksi Unityn omaa Gradient Toolia. Unityn omasta Gradients toolista tämä eroaa siten, että käyttäjillä on mahdollisuus liikutella eri tekstuurityyppien väriarvoja samanaikaisesti. Myöhemmin tallennettava JSON-tiedosto voisi sisältää myös referenssin kunkin gradientin kohdalla käytettävästä kuviotekstuurista.

Kolmas työtila olisi näillä näkymin 3D-ohjelman puolella määritettyihin ”material slotteihin” eli materiaalipaikkoihin perustuva. Tässä työtilassa tulisi todennäköisesti enemmän piirtokutsuja, joita voitaisiin toisaalta taas vähentää yhtenäistämällä shadereiden rendausta.

Kuva 42. GUI.Toolbar -luokalla tuotettu työskentelytilanvalitsin

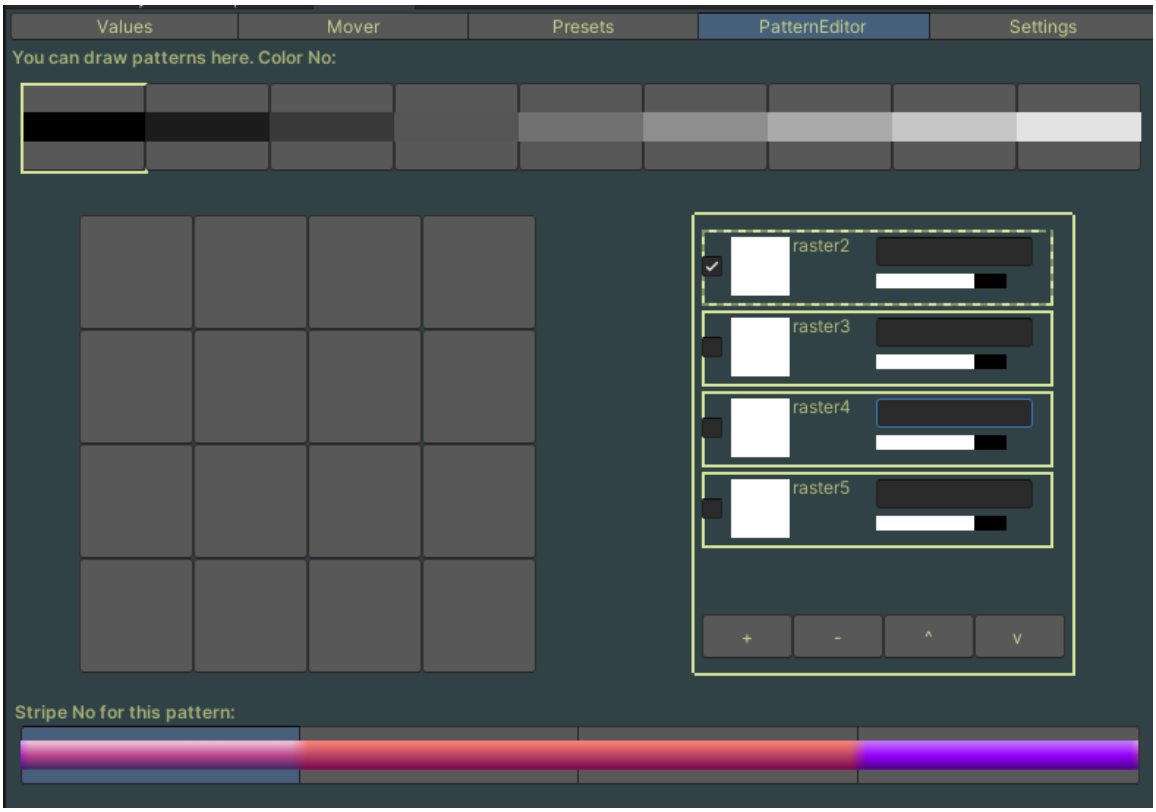
Lisäämällä kuviotekstuureja tuomme 3D-mallien teksturointiin lisää yksityiskohtia. Ennen kuviotyökalua kuviotekstuurit luotiin yhteen PSD-tiedostoon vektoritasoina, jotta voitiin myöhemmin skaalata kaikkia kuviotekstuureja saman tiedoston resoluutiota muuttaen.

Inspiraatiota kuvioeditoriin voidaan vetää esimerkiksi Animal Crossingista. Kuvan 43 Animal Crossing: New Horizonissa käytetty editori on leikkisä ja simppeli ja sopii täten hyvin gradienttitekstuuroinnin toimenkuvaan. Erona tällä hetkellä on se, että PBR Gradient toolissa käytetään pelkkää harmaansävyä. Kuviot todennäköisesti tullaan tallentamaan työkalukansioon JSON-tiedostoina, koska niiden tallentamisessa ei näillä näkymin tarvita väridataa vaan pelkästään joukko harmaansävyarvoja.

Kuva 43. Animal Crossing: New Horizons -pelin kuvio-/tekstuurieditori

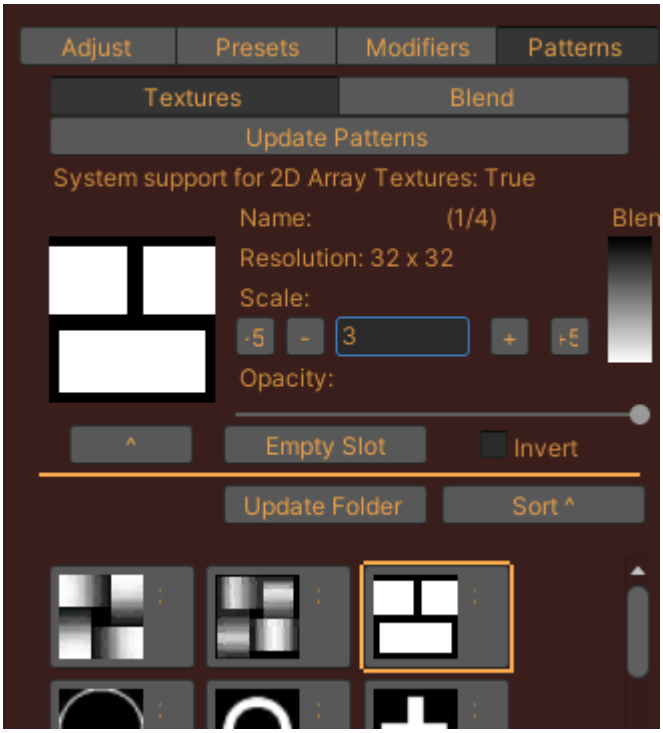
Kuvassa 44 näkyy kuvioeditorin tämänhetkinen tila. Siinä on ylhäällä arvovalitsinrivi, josta voidaan valita haluttu harmaansävy helposti. Oikealla nähdään tasovalitsin, jossa on rasteritasoja. Lisäksi suunnitteilla on täyttötaso, jolla olisi myös maski- sekä vektoritaso, joka on tällä hetkellä tarkoitettu yksinkertaisille muodoille. Työkaluosio tästä vielä puuttuu, koska työkaluikoneita ei

ole vielä tehty eikä toiminnallisuuttakaan, paitsi yksinkertaiselle pikselin värjäykselle. Aiemmin kuviotekstuurit tehtiin Kritassa vektoritasoille, jotta resoluution muuttaminen olisi helpompaa.



Kuva 44. Työkalun tämänhetkinen kuvioeditori

Kuviotekstuurien lisäksi suunnitteilla on blend- eli sekoitussysteemi, joka näkyy kuvassa 45. Siihen kuuluu materiaaliin tallennettava Texture2DArray, joka sisältää erilaisia kuviotekstuureita, ja pieni yksi pikseli per gradientti infotekstuuri, johon on tällä hetkellä merkattuina kunkin kuviotekstuurin skaala Color-muuttujan r-arvoon ja kuviotekstuurin läpinäkyvyys b-arvoon. Systeemin takaisku tällä hetkellä on Texture2DArrayhyn määritettävä tekstuurien yhteinen resoluutio, jolloin 3D-mallissa, jossa on pieni osa kuviotekstuuria gradienttia kohden, voi toinen osa vaatia isoresoluutioisemman tekstuurin. Lisäksi Texture2DArray pitää asettaa koodin kautta, eivätkä ne tallennu editorin muistiin, ainakaan vakiona. Blend-gradientteja on myös mahdollista tallentaa esiasetuksiksi. Peliobjektin Blend-gradientit tallentuvat tällä hetkellä samaa JSON-tiedostoa kuin materiaalikohtainen gradientti-info. Blend-napin takaa ja ikkunan oikealta reunalta löytyy muokattava harmaasävyinen gradientti, jonka avulla shaderi sekoittaa värigradienttia ja kuviotekstuuria keskenään. Tällä hetkellä itse gradienttitekstuuri käyttää 3D-mallin ensimmäistä UV-karttaa ja kuviotekstuuri käyttää 3D-mallin toista UV-karttaa.



Kuva 45. Työkalun tämänhetkinen kuviotekstuuriosio

Työkalu on tarkoitus julkaista Unity Asset Storessa tai Gumroad-sivustolla, joka on digitaalinen markkinapaikka digitaalisten palvelujen myyntiin, kunhan työkalun kehitys on tarpeeksi pitkällä tai sitten julkaista, jonkin avoimemman lisenssin alla. Jälkimmäinen vaihtoehto siksi, että taidot ovat edelleen sen verran puutteelliset, että työkalun täysin käyttövalmiiksi saattaminen veisi paljon aikaa ja myös sen takia että teksturointitavasta kiinnostuneet henkilöt voisivat tuoda siihen oman työpanoksensa ja omat ideansa. Työkalun lopullinen nimi on vielä tuntematon. Tähän mennessä se on ollut kehityksessä ”StriperTool”-nimellä, mutta se ei todennäköisesti tule jäämään käyttöön Stipe-rahoituspalvelun nimen takia tai koska nimestä puuttuu ”gradient”-osa, joka on hyvin olennainen osa nimeä. Myös nimi ”PBR Gradient Tool” on jo Unityn sisällä käytössä.

10 Yhteenveto

Opinnäytetyön keskeisenä tuloksena voidaan pitää itse työkalua. Työkalun tekoon päästiin tutustumalla itse työtapaan, siihen liittyviin pieniin yksityiskohtiin ja hidastaviin kohtiin. Lopputulos vaati suunnittelun lisäksi myös koodiosaamista, jota oli Unityn ja C#:n osalta jo aiemmin, mutta sen syventäminen ja varsinkin pääosassa oleva EditorWindow -luokka ja sen funktiot tulivat työn aikana tutuiksi.

Työ onnistui melko hyvin laajuudestaan huolimatta. Työkalun lopullinen tila tullaan näkemään tulevaisuudessa tämän opinnäytetyön ulkopuolella.

Työn rajaamiseen olisi kannattanut kiinnittää alussa erityistä huomiota. Toinen aihe olisi ollut oman jaksamisen ja koulun etenemisen kannalta järkevä valinta, mutta olin jo ajatellut tätä pidemmän aikaa ja olin joka tapauksessa jatkotyöstämässä kyseistä työkalua. Opin projektin aikana lisää C#:sta, Unitystä sekä koodin ja grafiikan välimaastosta.

Opinnäytetyö osoittaa, että kyseistä työtapaa voidaan todellakin viedä eteenpäin työstämällä työtapaan suunniteltua työkalua. Opinnäytetyön aihetta voidaan kehittää eteenpäin kehittämällä työkalua eteenpäin ja lisäämällä siihen ominaisuuksia.

11 Lähteet

1. Dragon Head Speed Model [Video]. YouTube.com 2017 [viitattu: 25.05.2022]. Saatavilla: https://www.youtube.com/watch?v=igjsITPntvU

2. Character Model - Texturing with Lazy Unwrapping [Video]. YouTube.com 2017 [viitattu: 20.04.2022]. Saatavilla: https://www.youtube.com/watch?v=DD84GSJiTVc

3. Bitten Toast Games Devlog — Gradient Generator and Vertex Gradient Editor [Kuva]. Tumblr.com 2018 [viitattu: 26.09.2022]. Saatavilla: https://snddev.tumblr.com/post/173483488414/gradient-generator-and-vertex-gradient-editor

4. Gradient Texture Generator | Utilities Tools | Unity Asset Store [Kuva]. assetstore.unity.com 2022 [viitattu 25.11.2022]. Saatavilla: https://assetstore.unity.com/packages/tools/utilities/gradient-texture-generator-216180

5. Gradiator | Utilities Tools | Unity Asset Store [Kuva]. assetstore.unity.com 2017 [viitattu: 25.11.2022]. Saatavilla: https://assetstore.unity.com/packages/tools/utilities/gradiator-97008

6. Gradient Master – Art, Technically… [online]. markpetroart.com 2016 [viitattu 07.06.2022]. Saatavilla: http://www.markpetroart.com/unity-tools/gradient-master/

7. Gradient Maker | Tools | Unity Asset Store [Kuva]. assetstore.unity.com 2015 [viitattu 29.11.2022]. Saatavilla: https://assetstore.unity.com/packages/tools/gradient-maker-29252

8. Minion’s Art on Twitter [online]. Twitter.com 2022 [viitattu 26.09.2022]. Saatavilla: https://twitter.com/minionsart

9. Minions Art [online]. Tumbrl.com 2022 [viitattu 09.06.202]. Saatavilla: https://minionsart.tumblr.com/

10. Art spotlight: Kaya #gradientchallenge. Sketchfab Community Blog Web site. [online]. Sketchfab.com 2019 [viitattu 09.12.2019]. Saatavilla: https://sketchfab.com/blogs/community/art-spotlight-kaya-gradientchallenge/

11. Sketchfab Community Blog - » Art Spotlight: Modular Environment Hexes. [online]. Skecthfab.com 2019 [viitattu 09.12.2019]. Saatavilla: https://sketchfab.com/blogs/community/art-spotlight-modular-environment-hexes/

12. Astro Kat Demo V1.1 by TibbleTop [Kuva]. tibbletop.itch.io [viitattu 25.11.2022]. Saatavilla: https://tibbletop.itch.io/astrokat

13. A Sucker For Sly | Eurogamer.net [Kuva]. Eurogamer.com 2004 [viitattu 25.11.2022]. Saatavilla: https://www.eurogamer.net/fi-sly2bandofthieves-ps2

14. Animal Crossing New Leaf solo ofrecerá DLC gratuito | Hobbyconsolas [Kuva]. hobbyconsolas.com 2012 [viitattu 25.11.2022]. Saatavilla: https://www.hobbyconsolas.com/noticias/animal-crossing-new-leaf-solo-ofrecera-dlc-gratuito-44800

15. Dota 2 Workshop - Character Art Guide. [Kuva]. Steampowered.com [viitattu 9.12.2019]. Saatavilla: https://support.steampowered.com/kb/9334-YDXV-8590/dota-2-workshop-character-art-guide

16. The Mean Greens - Plastic Warfare on Steam [Kuva]. Steampowered.com [viitattu 26.11.2022]. Saatavilla: https://store.steampowered.com/app/360940/The_Mean_Greens__Plastic_Warfare/

17. Unity - Scripting API: Texture.isReadable [online]. docs.unity3d.com [viitattu 04.05.2022]. Saatavilla: https://docs.unity3d.com/ScriptReference/Texture-isReadable.html

18. Körner E. Working with physically based shading: A practical approach - unity technologies blog. Unity Blog Website. [online]. Blogs.unity3d.com 2015 [viitattu 09.12.2019]. Saatavilla: https://blogs.unity3d.com/2015/02/18/working-with-physically-based-shading-a-practical-approach/

19. LearnOpenGL - Gamma Correction [online]. learnopengl.com [viitattu: 21.10.2021]. Saatavilla: https://learnopengl.com/Advanced-Lighting/Gamma-Correction

20. What every coder should know about gamma | John Novak [Kuva]. johnnovak.net 2016 [viitattu 29.05.2022]. Saatavilla: https://blog.johnnovak.net/2016/09/21/what-every-coder-should-know-about-gamma/

21. Character Model - Texturing with Lazy Unwrapping – YouTube [Video]. youtube.com (2017, -09-29) [viitattu 05.03.2021]. Saatavilla: https://www.youtube.com/watch?v=DD84GSJiTVc

22. File.WriteAllBytes(string, byte[]) method (system.IO). [online]. [viitattu 05.03.2020]. Saatavilla: https://docs.microsoft.com/en-us/dotnet/api/system.io.file.writeallbytes

23. Gradient Texturing for Beginners [EP 4] #Beginner #Tutorial #Blender #lowpoly – YouTube [Video]. YouTube.com 2020 [viitattu 25.11.2022]. Saatavilla: https://www.youtube.com/watch?v=uOyiZaioX1U

24. Unity - Manual: Material charts [online]. docs.unity3d.com. [viitattu 29.05.2022]. Saatavilla: https://docs.unity3d.com/2022.2/Documentation/Manual/StandardShaderMaterialCharts.html

25. Unity - Scripting API: Texture.isReadable [online]. docs.unity3d.com. [viitattu 24.11.2022]. Saatavilla: https://docs.unity3d.com/ScriptReference/Texture-isReadable.html

26. City Girl 🗂️ (@citygirltime) / Twitter [online]. twitter.com. [viitattu 25.09.2022]. Saatavilla: https://twitter.com/citygirltime/status/1492218551003860994/photo/1