Tampereen ammattikorkeakoulu

Valaistus Unity 3D:ssä

Oona Liukkonen

OPINNÄYTETYÖ
Kesäkuu 2022

Tietojenkäsittelyn tutkinto-ohjelma
Game Production

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietojenkäsittelyn tutkinto-ohjelma
Game Production

LIUKKONEN, OONA:
Valaistus Unity 3D:ssä

Opinnäytetyö 44 sivua
Kesäkuu 2022

Tämän opinnäytetyön tarkoituksena oli tutkia valojen ominaisuuksia ja niiden käyttöä ja merkitystä pelivalaistuksessa sekä tutustua Unity 3D -pelimoottorin tarjoamiin valaistustyökaluihin indie-pelin kehittäjän näkökulmasta.

Opinnäytetyössä käsiteltiin valon tärkeimmät ominaisuudet sekä erilaiset valotyypit, joilla valaistus voidaan toteuttaa. Lisäksi tutustuttiin Unityn esirakennettuihin renderöintiputkiin ja vertailtiin niiden keskeisiä eroja valaistukseen liittyen. Lisäksi tutkittiin erilaisia valaistuksen optimointia edistäviä tekniikoita.

Työssä tarkasteltiin myös varjoja ja niiden ominaisuuksia ja tutkittiin, millaisia ongelmia varjoihin liittyy ja miten ne korjataan Unityssä. Opinnäytetyössä tutustuttiin myös siihen, mistä erilaiset valaistukseen liittyvät asetukset Unitystä löytyvät ja mitä asetukset sisältävät. Opinnäytetyöprosessissa tehtiin vaiheittainen esimerkkiympäristön valaistus, jossa hyödynnettiin opinnäytetyössä käsiteltyjä työkaluja ja valaistustekniikoita.

Valaistusta suunniteltaessa on oleellista määrittää valonlähteet ja tiedostaa valojen funktio pelissä. Myös kohdealustan ja sen rajoitteiden tunteminen on keskeistä jo valaistuksen suunnitteluvaiheessa. Valaistusta luotaessa on löydettävä tasapaino laadun ja optimoinnin välillä, ja monissa tilanteissa huomattiin, että valaistusta pystyttiin optimoimaan ilman että se vaikutti merkittävästi valojen ja varjojen laatuun. Valaistuksen kannalta oleellisimmiksi optimointitavoiksi osoittautuivat valokartoitus ja valoprobet, sillä niillä pystyttiin merkittävästi vähentämään piirtokutsujen määrää suoritusaikana. Opinnäytetyössä nähtiin, että on oleellista tehdä valaistuslaskelmia ennen suoritusaikaa aina kun se on mahdollista. Vertailtavista renderöintiputkista Universal Render Pipeline osoittautui sopivimmaksi indiepelin kehitystiimille, koska sillä on eniten kohdealustoja sekä indiepelitiimin tarpeiden kannalta keskeiset ominaisuudet. Sitä pystyy myös muokkaamaan C#-koodikielellä vastaamaan paremmin eri projektien tarpeita.

---

valaistus, peliympäristö, Unity 3D

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Business Information Systems
Game Production

LIUKKONEN, OONA:
Lighting in Unity 3D

Bachelor's thesis 44 Pages
June 2022

The purpose of this thesis was to study the properties of lights and their use and significance in game lighting, as well as to inspect the lighting tools provided by the Unity 3D game engine from the point of view of an indie game developer.

The main properties of light and the different light types, with which lighting can be implemented, are covered in this thesis. In addition, Unity's pre-built render pipelines are examined and their differences in terms of lighting are compared. Various techniques to improve lighting optimization are investigated. The thesis also examines shadows and their properties. A step-by-step lighting of an example environment was made, utilizing the tools and lighting techniques learned in the thesis process.

It was found to be essential to know the light sources and to be aware of the function of the lights in the game. Knowledge of the target platform and its limitations are also essential already at the design stage of lighting. When creating lighting, a balance must be found between quality and optimization. It was seen in the thesis process that calculations related to lighting should be computed before the runtime whenever possible.

Lighting, game environment, Unity 3D

SISÄLLYS

1 JOHDANTO 6
2 YLEISTÄ VALOISTA 7
2.1 Valojen ominaisuudet 7
2.2 Valaistuksen tavoitteet 8
3 VALAISTUS UNITY 3D:SSÄ 11
3.1 Renderöintiputket 11
3.1.1 Sisäänrakennettu renderöintiputki 13
3.1.2 Universal Render Pipeline 14
3.1.3 High Definition Render Pipeline 16
3.2 Valaistusasetukset Unityssä 18
3.2.1 Valaistusasetukset 18
3.2.2 Inspektori 20
3.3 Global Illumination 23
3.3.1 Reaaliaikainen Global Illumination 23
3.3.2 Beikattu Global Illumination 24
3.4 Lampputyypit 28
3.5 Varjot 30
3.5.1 Varjokartoitus 30
3.5.2 Varjokaskadit 31
3.5.3 Kallistumat 32
3.5.4 Cookiet 33
3.5 Jälkikäsittely 34
4 ESIMERKKIVALAISTUKSEN TOTEUTUS 35
5 POHDINTA 41

LÄHTEET 42

LYHENTEET JA TERMIT

Assetti	Unityn sisäinen esine tai asia.
CPU-kierros	Yhden prosessoritoiminnon suorittamiseen kuluva aika.
Suoritusaika	Englanniksi runtime. Aika, jolloin peli on toiminnassa.
Unity/Unity 3D	Unity Technologiesin pelimoottori ja pelinluontialusta

1 JOHDANTO

Tässä opinnäytetyössä pyritään kuvaamaan, mitä valaistukseen sisältyy Unity 3D:ssä ja miten valaistus toteutetaan. Idea opinnäytetyöhön lähti havainnosta, kuinka pienten budjettien indiepeleissä valaistus on usein toteutettu heikosti tai sitä ei ole optimoitu tarpeeksi. Vaikka peliobjektit olisivat hienot ja huolellisesti toteutetut, pelin graafinen ilmiasu jää keskinkertaiseksi ilman hyvää valaistusta ja huonosti optimoituna se vaikuttaa vahvasti suorituskykyyn.

Opinnäytetyön tavoitteena on myös selvittää, miten pienen indiepelitiimin on mahdollista toteuttaa valaistus Unityssä sen omilla työkaluilla. Valaistukseen on olemassa useita maksullisia ja ilmaisia työkaluja, joita voi lisätä Unity-projektiin Unityn Asset–kaupasta, mutta tämä opinnäytetyö ei kata niitä. Opinnäytetyöhön kuuluu esimerkkiympäristön vaiheittainen valaistus Unityssä. Esimerkkiympäristön tavoitteena on olla näyttävä, auttaa pelin tunnelmassa ja olla optimoitu toimimaan indiepelissä.

Koska aiheen termistö ei ole vielä kokonaan vakiintunut Suomessa, opinnäytetyössä käytetään paljon englanninkielisiä termejä. Termistöä käännettäessä huomattiin, että tekstin luettavuus ja yhdistäminen muihin tietolähteisiin vaikeutui joissain tapauksissa.

Tässä opinnäytetyössä valaistuksella tarkoitetaan erilaisten valo- ja varjovaikutelmien aikaansaamista pelimoottorissa. Pelimoottoreissa valaistus jäljittelee visuaalisesti reaalimaailman valoja.

2 YLEISTÄ VALOISTA

Ennen ympäristön valaistuksen aloittamista on tiedettävä, mitä sillä tavoitellaan, minkälainen tunnelma ympäristöön halutaan ja onko jotain, mihin katsojan katsetta halutaan ohjata. Valoja ei voi vain lisäillä ilman, että suunnitellaan, mistä valonlähteistä valo tulee. Näin silloinkin, kun valonlähde ei olisi katsojalle näkyvissä. Kun valonlähteen sijainti ja tyyppi tiedetään, voidaan selvittää, minkälaisia ominaisuuksia kyseisellä valolla kuuluu olla. Paras tapa oppia valoista on tutkia oikean elämän valoa tai tarkastella valoa referenssikuvista. Melkein mikä tahansa adjektiivi voidaan katsoa valon ominaisuudeksi. (Birn, 2013, luku 1).

2.1 Valojen ominaisuuksia

Valon suunnalla voidaan viestiä tunnelmaa. Alhaalta päin tuleva valo luo uhkaavan asetelman ja lähellä kohdetta oleva valo synnyttää suuret varjot, kuten kuvassa 1. Takaa tulevaa valoa kutsutaan rim-valoksi, ja se luo mystisen tunnelman ja tuo parhaiten esille kohteen siluetin. Jos valaistaan ulkoympäistöä, valolla on rooli ajan ilmaisussa. Aamulla ja illalla sivusta tuleva valo saa aikaiseksi pitkät varjot, toisin kuin keskipäivällä auringonvalon tullessa lähes suoraan ylhäältä. Katseen kohdetta suoraan edestäpäin osoittavaa pehmeää valoa kutsutaan tasavaloksi (eng. *flat light*). Se ei näytä juurikaan syvyyttä, sillä sen luomat varjot ovat minimaaliset. (Shaban, 2022, osa 2).

Kuva 1. Valonlähteen sijainnin vaikutus varjoihin.

Pieni valonlähde luo teräväreunaiset varjot ja iso valonlähde pehmeämmät varjot. Tähän kuitenkin vaikuttaa myös valonlähteen etäisyys. Esimerkiksi aurinko, joka on valtava valonlähde, on meistä niin kaukana, että se käyttäytyy kuin pieni valonlähde ja luo siten ”kovan” varjon. Kun auringonvalo taas osuu pilviin, jotka ovat meitä lähellä ja isoja, valo jakautuu ja muodostaa pehmeät varjot. Jos halutaan korostaa pieniä yksityiskohtia, käytetään teräväreunaisia varjoja ja pehmeät varjot sopivat esimerkiksi laajoihin maisemiin. (Shaban, 2020, osa 2).

Valon lämpötila mitataan kelvineillä (K) Mitä pienempi kelvin-luku on, sitä lämpöisempänä ja keltaisempana valo havaitaan. Vastaavasti mitä suurempi kelvin-luku on, sitä kylmempi ja sinisempi valo on. Kun tavoitellaan realistisuutta, oikean värilämpötilan valitseminen on tärkeää. Tilaan sopimaton värilämpötila saa pahimmillaan katsojan tuntemaan olonsa epämukavaksi. Kelvinin asteikolle 2000–3000 asettuvaa valoa kutsutaan usein lämpimäksi valkoiseksi. Se sopii parhaiten rauhalliseen tilaan, kuten talon olohuoneeseen tai makuuhuoneeseen. 4500–5000 kelviniä vastaa päivänvaloa ja sopii tiloihin, joissa kirkkaus on tärkeämpää kuin tunnelma. 5500 kelviniä ylittävä valo on kylmän sinertävää. Se sopii tilanteisiin, joissa tarvitaan mahdollisimman kirkas valaistus. (Lighting Tutor, 2022).

Valon värit jaetaan luonnollisiin ja keinotekoisiin. Luonnolliset värit lasketaan kelvineillä. Keinotekoisia värejä ovat esimerkiksi violetit ja vihreät valot.
Katsojan katseen ohjauksessa voi käyttää vastavärejä. Esimerkiksi sinertävässä ympäristössä pienikin oranssi kiinnittää nopeasti pelaajan huomion. (Shaban, 2022, osa 4).

2.2 Valaistuksen tavoitteet

Uskottavuus on tärkeä osa valaistusta. Vaikka ei tavoiteltaisi visuaalisesti fotorealismia, valaistuksen täytyy silti olla uskottava pelaajalle. Esimerkiksi jos auringonvalo paistaa suoraan ikkunasta huoneeseen, pelaaja olettaa että tämä valo on kirkkaampi kuin huoneen pöytälampusta tuleva valo. (Birn, 2013, luku 1).

3D-ympäristön valaistuksessa parempaan lopputulokseen pääsee usein lisäämällä valoja, vaikka ne eivät olisi niin sanotusti realistisia. Jos esimerkiksi himmeästi valaistussa peliympäristössä on yksityiskohta, johon pelaajan halutaan kiinnittävän huomiota, voidaan sitä valaista lisää ylimääräisellä, epärealistisella valolla. Samoin, jos jokin peliobjekti näyttää paremmalta, kun valo osuu siihen kulmasta, joka ei olisi oikeassa elämässä mahdollinen, ei tarvitse muokata koko ympäristöä, vaan haluttu lopputulos luodaan muokkaamalla valaistusta. Kuvassa 2 hahmon kasvoihin osuvaa valoa on lisätty valolla, joka ei ole lähtöisin liekistä. Realistisuuteen ei pidä pyrkiä liikaa pelikokemuksen kustannuksella. Esimerkiksi luolat ovat harvoin peleissä niin pimeitä kuin oikeassa elämässä. (Birn, 2013, luku 1; Shaban, 2022, osa 5).

Kuva 2. Hahmon kasvoja valaistu ylimääräisellä valolla.

Hyvällä valaistuksella pystytään ohjaamaan pelaajan katsetta ja varmistamaan, että mikään ei harhauta pelaajaa. Valaistuksella pyritään myös estämään pelaajan "eksyminen", niin että pelikokemus katkeilee. (Birn, 2013, luku 1; Pluralsight 2014).

Tunnelma syntyy valon eri ominaisuuksien kautta. Ympäristön lämpötila ja aika pystytään viestimään pelaajalle valojen ja varjojen avulla. Myös aikakautta voidaan kuvata valon väreillä: tulevaisuuteen sijoittuvassa ympäristössä voidaan käyttää valoissa keinotekoisia värejä. Valaistuksella voidaan myös tuoda esille tunteita: kylmillä valoilla, jotka langettavat tummia varjoja, saadaan aikaan surullista tai ahdistavaa tunnelmaa. Jos vihollispelihahmon vaarallisuutta tai julmuutta halutaan korostaa, voidaan se valaista alhaaltapäin siten, että hahmon kasvoihin syntyy tummat, uhkaavat varjot. (Shaban, 2022, osa 5, osa 7).

Eri lamppuja yhdistelemällä saadaan luotua monipuolinen valaistus. Valoa, joka toimii päävalona kutsutaan key-valoksi. Tämä päävalo kohdistuu valaistavaan kohteeseen ja simuloi tilan kirkkainta valoa, esimerkiksi ulkotilassa aurinkoa ja sisätilassa kattolamppua. Heikompia valonlähteitä simuloivia valoja kutsutaan fill–valoiksi, ja niiden varjot ovat usein pehmeämpiä kuin key–valon. Ne valaisevat kohteen eri kulmasta kuin key-valo. Pinnoilta kimpoavaa epäsuoraa valoa simuloidaan bounce-valolla. Spill-valoksi kutsutaan valoa, joka on heikompi ja pehmeämpi versio päävalosta. Sitä käytetään tilanteissa, joissa esimerkiksi hahmon kasvoihin jää liian tummia varjoja. Esimerkiksi kolmen valon tekniikassa päävalo valaisee muita valoja selkeämmin valaistavan objektin, toinen valo auttaa pehmentämään sen synnyttämää varjoa ja kolmas valaisee taustaa. (Birn, 2013, luku 5; Shaban, 2022, osa 7).

3 VALAISTUS UNITY 3D:SSÄ

3.1 Renderöintiputket

Renderöintiputken keskeisin tarkoitus on muuttaa 3D-näkymä kameran asennon ja suunnan perusteella (De Bock, 2013.) 2D-kuvaksi Unity tarjoaa käyttäjälle kolme erilaista esirakennettua renderöintiputkea: Build-in Render Pipeline, Universal Render Pipeline ja High Definition Render pipeline. Näistä kaksi jälkimmäistä ovat koodattavia renderöintiputkia, eli niitä on mahdollista muokata C#-koodilla tarpeita vastaaviksi. Näiden lisäksi on mahdollista myös luoda oma renderöintiputki käyttämällä Unityn tarjoamaa Scriptable Render Pipeline Core packagea tai lyhyemmin SRP Corea. (Unity Technologies, 2022a; Unity Technologies, 2022b).

Kun valitsee projektille sopivaa renderöintiputkea, on otettava huomioon, mille alustoille peliä kehitetään ja kuinka korkealaatuista valaistusta tavoitellaan. Eri putket käyttävät eri renderöintipolkuja (eng. *rendering path*). Ne ovat operaatiosarjoja, jotka vaikuttavat siihen, miten valaistus prosessoidaan. Renderöintiputki valitaan Unity-projektia luotaessa, ja sitä voi halutessaan vaihtaa myöhemmin. On kuitenkin huomioitava, että vaihdettaessa eri renderöintiputkien välillä kaikki Unity-assetit ja koodit eivät välttämättä ole yhteensopivia uuden putken kanssa. (Unity Technologies, 2022b).

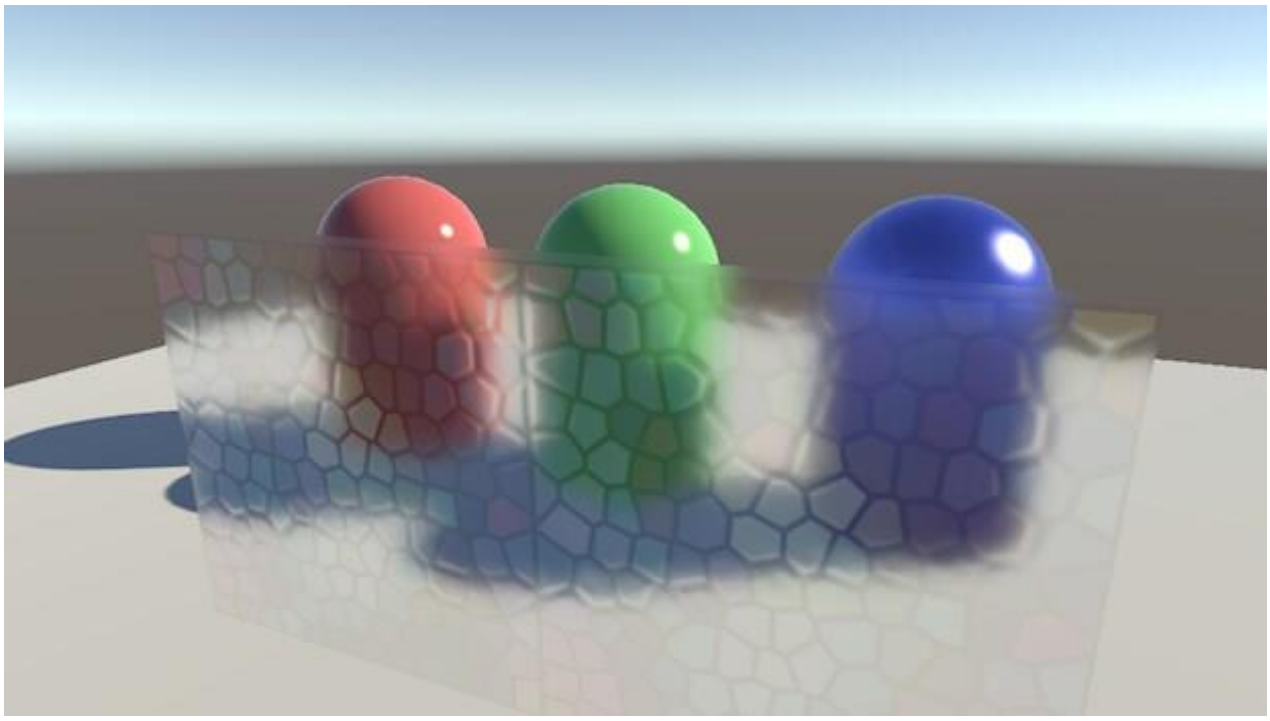
Renderöintipolut ovat erilaisia tekniikoita renderöidä ympäristö. Näistä kaksi yleisintä tapaa peleissä ovat suora renderöinti (eng. *Forward Rendering*) ja viivytetty renderöinti (*Deferred Rendering*). Suora renderöinti voidaan jakaa kahteen: Single-Pass ja Multi-Pass. Nämä renderoijat aloittavat näkymän piirtämisen selvittämällä mitkä objektit ovat kameran näköalueeseen. Tämän jälkeen Unity käy läpi kaikki kameran näköalueen sisäiset objektit aloittaen kamerasta lähimmistä ja renderöi ne tässä järjestyksessä. (Borromeo, 2020, luku 8) Suorassa renderöinnissä jokainen ympäristön objekti renderöidään kerran jokaista valoa kohti, jonka vaikutusalueella objekti on. Myös silloin kun valo ei

vaikuta objektiin, esimerkiksi kun välissä on seinä. Samat objektit siis saatetaan renderöidä useita kertoja riippuen lähellä olevien valojen määrästä. Kun valojen määrä on pieni, suora renderöinti on nopeaa. Käytettäessä viivytettyä renderöintiä valaistuksen renderöintikustannukset ovat verrannollisia valon valaisemien pikselien lukumäärään, eivätkä valojen määrään niin kuin suorassa renderöinnissä, joten kun peliympäristössä on paljon valoja, viivytetty renderöinti on nopeampaa suoraan verrattuna. Viivytetty renderöinti ei kykene renderöimään läpinäkyviä pintoja, joten joskus paras vaihtoehto on käyttää sekä suoraa että viivytettyä renderöintiä. Esimerkiksi Grand Theft Auto V -videopelissä ympäristö renderöidään ensin viivytetyllä renderöinnillä ja sen jälkeen ympäristön läpikuultavat osat renderöidään erikseen suoralla renderöinnillä. (Courrèges, 2015; Unity Technologies 2020e).

Piirtokutsu (eng. Darw Call) on prosessorin lähettämä pyyntö näytönohjaimelle piirtää geometriaa näytölle. Jokainen kutsu vaatii prosessorin ja näytönohjaimen välistä keskustelua. Yksinkertaistettuna prosessori (CPU) ”tietää”, mitä halutaan piirtää ja näytönohjain (GPU) ”tietää”, miten se piirretään. CPU:n täytyy muuttaa näkymä datamuotoon, jota GPU osaa lukea. Tämä prosessi vaatii huomattavan määrän CPU-kierroksia ja muistin kaistanleveyttä. Tämä tapahtuu joka kuvan (*frame*) kohdalla sovelluksen ollessa käynnissä, joten optimoinnin kannalta piirtokutsujen määrä on hyvä pitää mahdollisimman pienenä. Liiallinen määrä piirtokutsuja ilmenee esimerkiksi huonontuneena ruudunpäivitysnopeutena. Yksittäisellä piirtokutsun monimutkaisuudella on kuitenkin rajansa. Objektin ulkonäön ja siihen vaikuttavien valojen laskeminen on monimutkaista, joten suoran renderöinnin Single-Pass suorittaa valaistuksen laskemisen yksinkertaistettuna. Se tarkoittaa, että käytössä on vähemmän ominaisuuksia ja valaistuksen laatu on heikompi, kuin Multi-Passia käytettäessä. Multi-Passissa jokainen objekti lasketaan kerran suhteessa jokaiseen siihen vaikuttavaan valoon. Esimerkiksi, jos objektiin vaikuttaa kolme valoa, se renderöidään kolmesti. Tällöin käytettäviä ominaisuuksia on enemmän, mutta piirtokutsujen määrä kasvaa. Single-Passia käytettäessä objektiin voi vaikuttaa samanaikaisesti enimmillään kahdeksan eri valoa. Multi-Passilla ei ole tällaista rajoitusta. (Borromeo, 2020, luku 8).

3.1.1 Sisäänrakennettu renderöintiputki

Sisäänrakennettu renderöintiputki (eng. *Build-In Render Pipeline*) on Unityn vanhin renderöintiputki ja käytössä oletusputkena Sisäänrakennettua renderöintiputkea voidaan muuntaa vaihtamalla sen renderöintipolkujen välillä. Oletusarvona on suora renderöinti. Projektin renderöintipolkua voi vaihtaa projektin grafiikka-asetuksista (edit>project settings>graphics). Projektin kameran asetuksissa voi valita, haluaako käyttää grafiikka-asetuksissa määritettyä polkua vai vaihtoehtoisesti jotain toista polkua. Lisäksi putkea voidaan laajentaa käyttämällä komentopuskureita (eng. *CommandBuffer*). Komentopuskuri sisältää erilaisia renderöintikäskyjä, joilla voidaan muokata sisäänrakennetun renderöijän ominaisuuksia ja vaikuttaa siten esimerkiksi valaistusun. Koodin sisällä voidaan luoda komentopuskureita ja lisätä renderöintikäskyjä. Kuvassa 2, kun ympäristö ja läpinäkymättömät objektit on renderöity, luotu kuva on sumennettu ja siitä on tehty ominaisuus varjostimelle. Kuvan lasin varjostin pystyy siten käyttämään sumennettua kuvaa efektin muodostamiseksi. (Unity Technologies, 2022c; Unity Technologies, 2022d).



Kuva 3. Komentopuskureilla luotu sumennusefekti.

3.1.2 Universal Render Pipeline

Universal Render Pipelinella on Unityn oletusputkista laajin määrä kohdealustoja, eli se soveltuu niin mobiililaitteille suunnattujen pelien kehittämiseen kuin virtuaalitodellisuuteen tai suorituskykyisemmille tietokoneille suunnattuihin projekteihin. (Unity Technologies, 2022e).

Universal Render Pipeline eli URP käyttää oletuksena Single-Pass -suoraa renderöintiä. Se tarkoittaa, että kaikki yhteen objektiin vaikuttavat valot lasketaan yhdessä piirtokutsussa. URP on rakennettu scriptable render pipeline -teknologian päälle. Ja se on skaalautuva eli sitä voi muokata tarpeita vastaavaksi. URP tukee suoran renderöinnin lisäksi myös viivytettyä renderöintiä. Jotta viivytettyä renderöintiä voisi käyttää, on grafiikka-asetuksista vaihdettava Scriptable Render Pipeline Settings -kohtaan None. Tämän jälkeen voidaan URP-assetissa (URP Universal Renderer asset) vaihtaa renderöintipolku suorasta viivytetyksi. Kun polku on vaihdettu viivytetyksi Unity käyttää sitä ainoastaan pääkameralla renderöitäessä. Kamerat, joiden renderöintitila on Inspectorissa vaihdettu

ylittäväksi (eng. overlay), renderöidään käyttäen suoraa renderöintiä. Näin voidaan hyödyntää kumpaakin renderöintipolkua. (Borromeo, 2021, luku 8; Unity Technologies 2021).

URP:n ja muiden vertailtavien putkien välillä on muutamia keskinäisiä eroavaisuuksia. Tietyt valaistusasetukset Unityn Light Component Inspectorissa ovat saatavilla ainoastaan URP:ssa. (Unity Technologies, 2022f).

Universal Additional Light Data –komponentin avulla Unity varastoi valoon liittyvää dataa ja URP kykenee laajentamaan ja päällekirjoittamaan Unityn Standard Light -komponentin toimintoja. Peliobjektilla, johon sisältyy valokomponentti, pitää URP:tä käyttäessä olla myös Universal Additional Light Data -komponentti. Unity luo sen automaattisesti, kun valo luodaan. (Unity Technologies, 2022g).

Kamera-komponentilla on URP:lle tyypillisiä ominaisuuksia inspectorissa. Universal Additional Camera Data -komponentti laajentaa kamera-komponentin toimintoja ja antaa Unityn tallentaa kameraan liittyvää lisädataa. (Unity Technologies, 2022h).

URP käyttää renderöintiin Physically Based Rendering -tekniikkaa eli lyhennettynä PBR-tekniikkaa. Se tarkoittaa, että materiaaleihin voidaan lisätä eri ominaisuuksia tuottavia karttoja. Tällaisia ovat esimerkiksi kuten albedo, joka antaa materiaalille perusvärin, tai metallisuuskartta, joka vaikuttaa materiaalin kiiltävyyteen. PBR-materiaalit tarjoavat joukon muuttujia, joiden avulla pystytään saavuttamaan yhdenmukaisuutta eri materiaalityyppien välillä ja erilaisissa valaistusolosuhteissa. (Unity Technologies, 2022i).

URP ei tue kustomoituja varjostimia, joten varjostin on valittava URP:n omista vaihtoehdoista. URP Lit shader sopii lähes kaikille realistisille materiaaleille.
Complex Lit shader on tarkoitettu materiaaleille, jotka vaativat monimutkaisempia valaistuslaskelmia. Simple Lit shaderia käytetään muiden kuin PBR-materiaalien tuomiseen URP:hen. Näitä ovat esimerkiksi materiaalit, jotka on luotu sisäänrakennetussa renderöintiputkessa. Se ei tue varjostinkaavion käyttöä

Baked Lit shader sopii tilanteisiin, joissa ei käytetä ollenkaan reaaliaikaisesti päivittyviä valoja. Unlit Shader sopii tilanteisiin, joissa materiaaleihin ei kohdistu ollenkaan valoa. (Unity Technologies, 2022j).

Varjostinkaavio (eng. Shader Graph) on Unityn visuaalinen varjostimenrakennustyökalu, joka on automaattisesti lisättynä Unity-projekteihin, jotka käyttävät ohjelmoitavaa putkea kuten URP:tä ja HDRP:tä. Varjostimia luodaan yhdistelemällä säätimiä kaaviokehyksessä, ja muutokset käsiteltävässä varjostimessa tulevat esiin reaaliajassa rakennustyökalun esikatseluikkunassa. (Borromeo, 2020, luku 6; Unity Technologies, 2022k).

Universal Render Pipeline Asset määrittää useita graafisia ominaisuuksia ja laatuasetuksia Universal Render Pipelinessa. Projektissa voi olla useita URP-assetteja samalle ympäristölle. Esimerkiksi jos peliä kehitetään sekä tietokoneelle että konsolille, niille kummallekin voidaan luoda omat URP-assetit vastaamaan niiden tarpeita. Assetteja voidaan helposti vaihtaa grafiikka-asetuksista. Assettia ei kuitenkaan voi vaihtaa eri renderöintiputken assettiin (HDRP/SRP), koska ne eivät ole yhteensopivia. (Unity Technologies, 2020a).

3.1.3 High Definition Render Pipeline

High Definition Render Pipeline eli HDRP on korkeatarkkuisin Unityn esirakennetuista renderöintiputkista. HDRP käyttää URP:n tavoin PBR-valaistusta ja -materiaaleja, ja tukee sekä suoraa että viivytettyä renderöintiä. Renderöintipolkua pääsee vaihtamaan HDRP:tä käytettäessä laatuasetuksista. HDRP käyttää laskentavarjostin -tekniikkaa. (eng. C*ompute Shader technique*) Laskentavarjostimet ovat varjostinohjelmia, jotka toimivat näytönohjaimella normaalin renderöintiputken ulkopuolella ja vaativat siten yhteensopivan näytönohjaimen. Tämä rajaa mahdollisia kohdealustoja.

Kun halutaan kehittää AAA-pelejä, teknisiä demoja tai animaatioita esimerkiksi mainoksiin, HDRP on paras valinta. HDRP ei ole yhteensopiva vanhempien alustojen kanssa eikä se sovi mobiilipelien kehittämiseen. (Unity Technologies, 2022l).

HDRP on ainoa Unityn esirakennetuista renderöintiputkista, joka tukee RTX-pelien kehittämistä. Säteenseurannan avulla päästään käsiksi dataan, joka ei näy näytöllä. Sitä voidaan käyttää niin sijainti-, normaali- kuin valaistustietojen laskemiseen. Käytettäessä valaistusssa säteenseurantaa, Unity simuloi aitoa valoa ja seuraa, miten se kimpoaa ja muuttuu ympäristön pinnoilla. Tämä tuottaa kaikista realistisimman tuloksen mutta on erittäin raskas prosessoitava. Tämän vuoksi säteenseuranta vaatii RTX-yhteensopivan näytönohjaimen. Lista yhteensopivista näytönohjaimista löytyy Unityn dokumentoinnista. (Unity Technologies, 2020b).

Säteenseuranta ei automaattisesti ole käytössä projektissa. Se toimii ainoastaan silloin, kun projekti käyttää DirectX 12 -ohjelmointirajapintaa. Unity projekteissa toimii oletusarvoisesti DirectX 11 –ohjelmointirajapintaa, joka täytyy vaihtaa Render Pipeline Wizardilla. (Window>Render Pipeline>HD Render Pipeline Wizard>HDRP + DXR>Fix All) Tämän jälkeen projekti tukee säteenseurantaa.
Säteenseurannan ollessa käytössä, voidaan määrittää milloin sitä käytetään. Esimerkiksi kun säteenseurantaa halutaan käyttää oletuksena projektin asetuksista, HDRP Default Settings -välilehden alta voidaan valita Default Frame Settings -kohdasta kamera ja tämän alta Rendering-kohdasta “säteenseuranta”. Kun säteenseurantaa käytetään vain tietyssä kamerassa, inspectorissa voidaan General -kohdassa valita käyttöön Custom Frame Settings. Tällöin kehysasetukset vaikuttavat vain tähän kameraan ja rendering-kohdasta voidaan valita säteenseuranta. (Unity Technologies, 2020b).

Painamalla HDRP-assettia projekti-ikkunassa, voidaan sen inspectorissa määrittää haluttu RTX-taso. Taso 1 tasapainottaa suorituskyvyn ja laadun. Sitä käytetään peleissä ja muissa korkean kuvanopeuden sovelluksissa. Tämä taso ei tue valaistusssa moninkertaista pinnoilta kimpoamista.

Taso 2 on huomattavasti resurssi-intensiivisempi kuin taso 1. Se mahdollistaa laadukkaammat tehosteet ja tukee useita valon kimpoiluja pinnoilta. Se mahdollistaa myös polunjäljittimen (*eng. path tracing*). Polun jäljitin lähettää säteitä kamerasta ja jäljittää niitä, kunnes ne osuvat heijastavaan tai taittavaan pintaan. Sitten se muuttaa säteen suuntaa pinnan ominaisuuksien mukaan ja toistaa prosessia, kunnes säde saavuttaa valonlähteen. Säteiden sarja kamerasta valoon muodostaa "polun". Tämä on HDRP:n resurssi-intensiivisin säteenseurantamenetelmä. (Unity Technologies, 2020c).

HDRP käyttää kanavapakattuja (eng. channel packed) tekstuureja, jotka varastoivat useita materiaalikarttoja yhteen tekstuurin. Näiden avulla on mahdollista luoda kaikista vaativimpia materiaaleja, joilla on monenlaisia ominaisuuksia. Pienessä pelinkehitystiimssä ei oletettavasti ole aikaa ja tarvetta, lukuisien karttojen luomiseen ja yksinkertaisemmatkin materiaalit tuottavat halutun lopputuloksen. (Unity Technologies, 2020d).

3.2 Valaistusasetukset Unityssä

Kun sopiva renderöintiputki on valittu ja Unity–projekti on saatu auki, päästään luomaan ja muokkaamaan valaistusta. Unityn valaistusasetukset löytyvät yläpalkin Window-kohdan Rendering–sarakkeesta, ja ne voi halutessaan kiinnittää Unityn oletusnäkymään, niin että valaistusasetuksiin pääsee jatkossa nopeammin. Renderöintiputkien välillä on eroja, mutta asetukset ovat pääasiassa samat.

3.2.1 Valaistusasetukset

Valaistusasetuksien yläreunasta löytyy sarakkeet Scene, Environment ja Baked Lightmaps. Tämän lisäksi sisäänrakennetussa renderöintiputkessa on erillinen

sarake reaaliaikaisille valokartoille. Scene ja Environment sisältävät asetuksia, kun taas valokartta-kohdissa pääsee tarkastelemaan luotuja valokarttoja.

Scene-näkymän asetuksia ei pääse muokkaamaan, ennen kuin on luotu uudet valaistusasetukset painamalla New Lighting Settings -kohdasta. Tämä luo Assets-kansioon Lighting Settings –assetin, ja sen voi nimetä haluamallaan tavalla. Tämän jälkeen asetuksia pääsee muokkaamaan.

Ylimpänä valaistusasetuksista löytyy Realtime Lighting, josta voi valita käyttöön Unityn Realtime Global Illuminationin (GI) eli Enlightenin. Sen alta löytyvän reaaliaikaisen ympäristövalaistuksen voivalita käyttöön, ainoastaan jos myös reaaliaikainen GI ja beikattu GI ovat päällä. Realtime Lighting –kohdan alta löytyvät Mixed Lighting –asetukset, joista pystytään valitsemaan beikattu GI sekä valaistustila. (Unity Technologies, 2022m).

Kun valaistustilana on Baked Indirect, yhdistetyt valot (eng. *mixed lights*) käyttäytyvät reaaliaikaisten valojen tavoin. Dynaamiset objektit vastaanottavat reaaliaikaista suoraa valoa ja luovat reaaliaikaisia varjoja. Staattiset objektit vastaanottavat suoran reaaliaikaisen valon lisäksi myös valokarttaan beikattua epäsuoraa valoa. Myös staattiset objektit luovat reaaliaikaisia varjoja, mikä voi näkyä suorituskyvyssä. Suorituskykyä voi parantaa säätämällä varjojen etäisyyttä pienemmäksi. (Unity Technologies, 2022n).

Kun valaistustilana on Subtractive, kaikki yhdistetyt valot tuottavat suoraa ja epäsuoraa beikattua valoa. Näiden lisäksi pääsuunnallinen valo (eng. *main directional light*) tuottaa reaaliaikaisia varjoja dynaamisiin peliobjekteihin. Koska varjot on tässä valaistustilassa beikattu valokarttoihin, Unity ei pysty tarkasti yhdistämään beikattuja ja reaaliaikaisia varjoja. Sen sijaan reaaliaikaisten varjojen väriä ja vahvuutta pystyy säätämään manuaalisesti, niin että ne vastaavat beikattuja varjoja. Subtractive-valaistustila sopii projekteihin, joissa tarvitaan ainoastaan yhtä reaaliaikaisia varjoja tuottavaa valoa ja joissa pyritään säästämään suorituskyvyssä. (Unity Technologies, 2022o).

Kuten baked direct, myös shadowmask yhdistää reaaliaikaista suoraa valaistusta ja beikattua epäsuoraa valaistusta. Erona muihin valaistustiloihin shadowmaskissa varjot beikataan omaan erilliseen varjokarttaansa. Sen avulla Unity pystyy suoritusaikana yhdistämään reaaliaikaiset ja beikatut varjot. (Unity Technologies, 2022p).

Valokartoitusasetuksista päästään vaihtamaan Enlighten, Progressive CPU ja Progressive GPU -valokartoittajien välillä. Progressive GPU on yleisesti ottaen nopeampi renderöinnissä, kuin Progressive CPU, ja sitä kannattaa käyttää kun käytössä on näytönohjain. Progressive Updates -valintaruutu vaikuttaa siihen, priorisoidaanko beikkaamisessa alue, joka on näkyvissä scene-näkymässä. Suuren ympäristön beikkaamiseen voi kulua paljon aikaa, ja valinnalla voidaan nopeuttaa muutoksen näkymistä pelimaiseman siinä kohdassa, jota halutaan tarkastella. (Unity Technologies, 2022m).

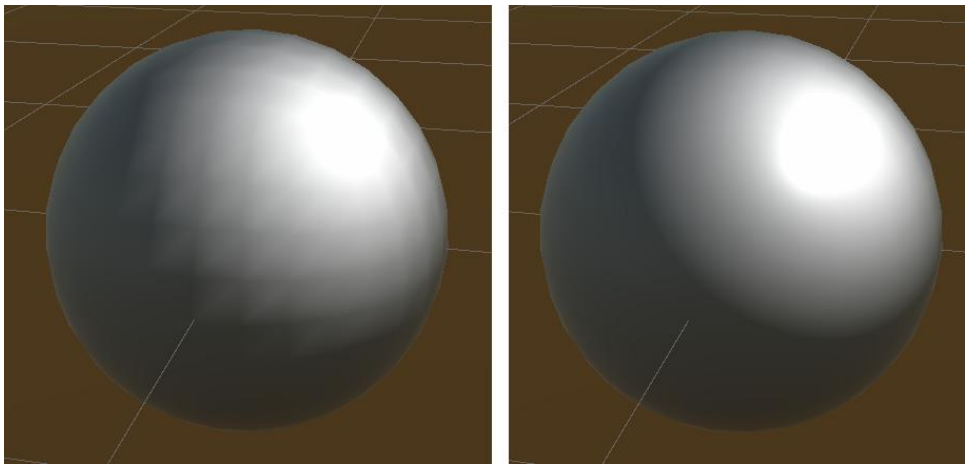
Ympäristöasetuksista pääsee valitsemaan ympäristön skyboxin sekä määrittämään, minkä mukaan ympäristövalo luodaan. Auringonlähde-kohtaan voidaan määrittää suunnallinen valo silloin, kun käytössä on muuttuva skybox, kuten Unityn oletus-skybox. Tällöin auringonlähteeksi merkittyä valoa kääntämällä skybox vaihtaa väriään ja vastaa eri päivänaikoja suunnatun valon mukaan. Kun suunnattu valo osoittaa suoraan alas, skybox on sinisimmillään, sivulle osoittaessaan iltaisen keltainen ja suoraan ylöspäin osoittaessaan skybox vaihtuu öiseksi taivaaksi. Tämän avulla peliin voidaan luoda vuorokausikierros. Realtime Shadow Color –kohdassa voidaan määrittää varjojen väri, kun valaistustilana käytetään Substractivea. (Unity Technologies, 2022m).

3.2.2 Inspektori

Inspektori on Unityn tarkasteluikkuna ja se näyttää yksityiskohtaiset tiedot valitusta peliobjektista, mukaan lukien kaikki siihen liitetyt komponentit ja niiden

ominaisuudet. Inspektorista päästään siis muokkaamaan esimerkiksi valojen ominaisuuksia tai sitä, miten valot vaikuttavat peliobjekteihin. (Unity Technologies, 2022q).

Kun valittuna on valo-peliobjekti, voidaan Inspektorissa sen valotyyppiä vaihtaa type-kohdasta. Muokattavissa on valon ominaisuuksia, esimerkiksi sen voimakkuus ja väri. Rendering Mode -kohdasta voidaan määrittää valon tärkeys. Kun valo asetetaan tärkeäksi, se renderöidään aina pikselikohtaisesti, kuten kuvassa 3 (Borromeo, 2021) oikealla. Ei-tärkeäksi merkitty valo renderöidään vertex-kohtaisesti, kuten kuvassa 3 vasemmalla. Kun tämä asetus on jätetty automaattiseksi, Unity renderöi valon läheisten valojen kirkkauden sekä laatuasetusten mukaan, joko pikseli- tai vertex-kohtaisesti. (Borromeo, 2021; Unity Technologies, 2022q).



Kuva 4. Vertex-kohtainen ja pikselikohtainen valaistus. (Borromeo, 2021).

Valon tilaa voi vaihtaa beikatun, reaaliaikaisen ja yhdistetyn välillä. Reaaliaikainen valo lasketaan nimensä mukaisesti reaaliajassa joka kuvalla (eng. *frame*). Tämän takia se on suorituskyvyn kannalta vaativin valaistustila. (Unity Technologies, 2022r).

Beikattu valo tarkoittaa sitä, kun Unity laskee valaistuksen etukäteen ja tallentaa sen valokarttoihin eli suuriinn tekstuureihin, jotka sitten asetetaan pelitilassa

ympäristön päälle simuloimaan valoa. Beikatut valot eivät vaadi suorituskykyä pelitilassa, mutta ne eivät muutu tai liiku. (Unity Technologies, 2022r).

Yhdistetty valaistus on yhdistelmä sekä reaaliaikaista että beikattua valaistusta. Esimerkiksi aurinkona toimiva suunnallinen valo kannattaa asettaa yhdistetyksi, koska silloin se osallistuu valokartan luontiin sekä vaikuttaa pelin ollessa käynnissä dynaamisiin peliobjekteihin. Unityn inspectorissa peliobjektin voi asettaa staattiseksi, tämä tarkoittaa sitä, että objektin varjot beikataan mukaan valokarttaan. Jos tämän jälkeen objektia siirretään ilman että kartta luodaan uudestaan, objektin luoma varjo jää alkuperäiseen sijaintiinsa eikä liiku objektin mukana. (Borromeo, 2021; Unity Technologies, 2022r).

Staattisuuden lisäksi peliobjektien vaikutusta Global Illuminationiin voidaan vaihtaa Inspektorissa. Esimerkiksi ympäristössä, jossa valaistus ei muutu, kokonaan varjossa olevat peliobjektit on hyvä laittaa osallistumaan Global Illuminationiin ainoastaan valoprobejen avulla. Tällöin objekti ei vie tilaa valokartasta, vaan sen valaistus määräytyy lähimpien valoprobejen mukaan. (Continisio, 2020; Unity Technologies 2022q).

Inspektorissa pystyy myös määrittämään, miten peliobjektista syntyy varjoja ja vaikuttavatko muut varjot itse objektiin. Varjot voi laittaa halutessaan kokonaan pois, tai vaikka vaihtaa asetusta niin, että peliobjekti ei näy mutta luo silti varjon ympäristöönsä. Kaksipuolisuus varjojen luonnissa tarkoittaa sitä, että varjot luodaan objektin pinnan kummaltakin puolelta. Esimerkiksi peliobjekti *Plane* on suorakaiteen muotoinen objekti, jolla on tavallaan vain yksi puoli, ja jos valo tulee sen takaa, se kulkee sen läpi muodostamatta varjoa. Jos kaksipuoleisuus on valittuna, Plane muodostaa varjon, tuli valo sitten sen miltä puolelta tahansa. (Unity Technologies, 2022q).

3.3 Global Illumination

Global Illumination on järjestelmä, joka kuvastaa valoa, joka kimpoaa pinnoilta toisille eikä rajoitu pelkästään pinnoille, joihin valo osuu suoraan. Global Illuminationin simuloiman valon pystyy jakamaan kahteen osaan. Ensimmäinen osa on suora valo, siis valo, joka kulkee suoraan valonlähteestä valaistavalle pinnalle. Toinen osa on epäsuora valo, joka tarkoittaa valoa, joka kimpoaa pinnan kautta toiselle pinnalle. Global Illumination auttaa luomaan realistisempaa valaistusta peliympäristöön. Unity tarjoaa kahta erilaista Global Illumination -järjestelmää, Baked Global Illuminationia ja Realtime Global Illminationia. (Unity Technologies, 2020e).

3.3.1 Reaaliaikainen Global Illumination

Realtime Global Illumination eli Unityn Enlighten tulee poistumaan Unitystä muutaman vuoden sisällä, ja sen tukeminen lopetetaan. Unity oletettavasti pyrkii kehittämään sille korvaajan. (Mortensen, 2019.)
Enlighten ei perustu valokarttojen beikkaamiseen vaan käyttää esilaskettua näkyvyysdataa nopeauttamaan reaaliaikaisten valojen laskentaa. Enlighten on tarkoitettu käytettäväksi, kun ympäristössä on jokin valaistuksen kannalta merkittävä, hitaasti liikkuva asia, esimerkiksi liikkuva aurinko. Enlighten ei sovi nopeasti muuttuville valoille, koska se kasvattaa merkittävästi CPU-kierrosten määrää. Lisäksi Enlightenia käytettäessä voi esiintyä viivettä, joka ei haittaa suurissa ja hitaissa valonmuutoksissa mutta näkyy nopeissa muutoksissa. Viivettä voi pyrkiä vähentämään laskemalla reaaliaikaisen valokartan kokoa tai lisäämällä CPU:n käyttöä laatuasetuksista. Enlighten jakaa ympäristön pinnat osiin ja määrittää niiden näkyvyyden toisiinsa. Suoritusaikana tätä tietoa hyödynnetään siinä, kuinka Unity laskee reaaliaikaisen valon kimpoilun pinnoilla. Valokartan reaaliaikainen päivitys on laskennallisesti raskasta, joten sen laskemiseen kuluu useampi kuva. Reaaliaikaista Global Illuminationin kanssa voidaan käyttää myös

reaaliaikaista ympäristövalaistusta eli ympäristövaloa, joka päivittyy reaaliajassa. (Unity Technologies 2022i).

3.3.2 Beikattu Global Illumination

Beikattu Global Illumination sisältää valokartoituksen, valoprobet, ja heijastusprobet.

Valokartoituksessa näkymän valaistusinformaatio tallennetaan tekstuuriin ennen suoritusaikaa, eli se *beikataan*. Tekseli eli texture pixel on nimensä mukaisesti tekstuurin pikseli. Objektien tekselien kokoa voi muokata Unityn Inspectorissa. Mesh rendererin alta löytyy Lightmapping -kohdasta Scale in Lightmap, josta voidaan asettaa kerroin, jonka mukaan objektin osuus kartassa skaalautuu. Mitä suurempi luku, sitä enemmän tilaa objekti käyttää valokartasta, eli siihen vaikuttavat valot ja varjot ovat korkeampiresoluutiollisia ja näyttävät siis paremmilta. Esimerkiksi pelimaisemassa taustalla tai pelaajan ulottumattomissa olevien objektien osuutta valokartassa kannattaa laskea laittamalla niiden skaalauskerroin pieneksi, sillä jos pelaaja ei koskaan pääse tarkastelemaan niitä läheltä, pienempikin resoluutio riittää. (Unity Technologies 2020e, luku 2).

Valokartan toimintaa voidaan hahmottaa kuvittelemalla jokaisesta kartan tekselistä lähtevän säde, joka pyrkii löytämään valonlähteen. Tämä säde kimpoilee pinnoilta ja kun se löytää esimerkiksi sinisen taivaan, josta ympäristö saa valoa, se palaa takaisin pinnalle, josta se kimposi. Säde jättää siihen sinertävää valoa ja jatkaa sitten kohti alkuperäistä tekseliään menettäen jokaisella kimpoamisella sinisen valon voimakkuutta. Säteen kimpoilujen määrää pystyy säätämään Inspektorissa. Kimpoilujen lisääminen kasvattaa rendreöintiin kuluvaa aikaa, joten se kannattaa pitää riittävän pienenä. Joskus valokarttaan jää kohinaa. Tämä johtuu siitä, että tekseleistä lähtevät säteet liikkuvat satunnaisiin suuntiin ja löytävät siten eri määriä valoa. Kohinasta pääsee eroon lisäämällä kimpoilujen määrää tai käyttämällä suodatusta, joka on oletusarvoisesti päällä. Sen voi halutessaan joko laittaa pois

päältä tai vaihtaa kehittyneeseen tilaan, joka avaa joukon eri säädöksiä, joilla suodatusta voi muokata. Suora suodatin –kohdassa voidaan vaihtaa Gullianin ja A-Trousin välillä. Gulliani sumentaa pinnan valoa radius–liukusäätimellä määritetyn määrän, A-Trous vertaa reunoja ja pyrkii välttämään toisiinsa kuulumattomien objektien, esimerkiksi maan ja seinän, yhteensumentumista. (Continisio, 2020, 18:50).

Valoprobe (eng. *Light probe*) on piste, joka tallentaa valaistustiedon ympäriltään näkymästä. Kun dynaaminen peliobjekti, esimerkiksi pelaaja, kulkee ympäristössä, se voidaan valaista lähimpien valoprobejen mukaan. Valoprobet tallentavat valaistusinformaatiota valokartan tavoin, mutta sen sijaan, että ne tallentaisivat pinnoille osuvan tiedon valosta, valoprobet tallentavat tiedon valosta, joka kulkee tyhjässä tilassa eli "ilmassa".

Light Probe group on joukko valoprobeja, jotka asetellaan peliympäristöön siten, että ne kattavat koko alueen, jolla dynaamiset esineet liikkuvat. Valoprobeja pyritään laittamaan ympäristöön mahdollisimman vähän, mutta kuitenkin aina kun valo muuttuu, esimerkiksi varjojen rajoille tai lamppujen läheisyyteen. Alueilla, joilla ei ole paljon muutoksia valaistusssa, riittää vähemmän probeja kuin alueilla, joissa on paljon erilaista valoa. Valoprobeja käytettäessä voidaan säästää myös valokartan tilaa. Esimerkiksi staattiseksi merkittyä peliobjektia, joka on kokonaan varjossa, ei kannata tallentaa valokarttaan, vaan se voidaan asetuksista asettaa ottamaan valaistustietonsa valoprobeista. Näin muiden alueiden varjot voidaan tallentaa tarkemmin, ilman että varjokartan kokoa pitää kasvattaa. Valoprobeja tarvitaan ainoastaan alueille, joihin pelaaja pääsee tai joissa on liikkuvia asioita. (Continisio, 2020, 43:00).

Heijastusprobet (eng. *Reflection Probes*) ovat valoprobejen tavoin pisteitä, jotka tallentavat tietoa ympäristöstään. Heijastusprobe luo ympäristöstään kuution muotoisen peilikuvan kuutiokartaksi, jota voidaan käyttää ympäristön pintojen heijastuksissa, kuten kuvan 5 ikkunassa. Heijastusprobejen vaikutusalueet voidaan määrittää Unityssä siten, että esimerkiksi ulkotilassa on käytössä eri probe kuin

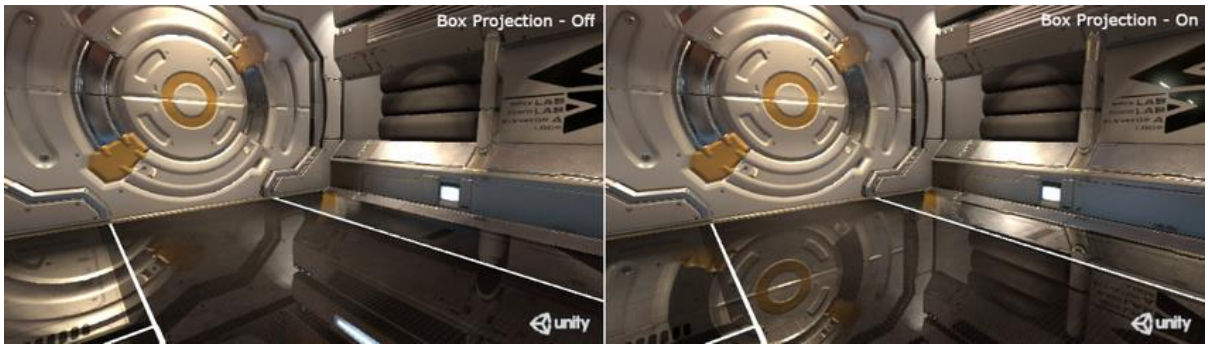
sisätiloissa. Tällöin esimerkiksi pelaajan aseessa näkyvä heijastus vastaa ympäristöänsä eikä heijasta sisätilassa taivasta.

Heijastusprobejen luoma heijastuma ei ole kuitenkaan täysin realistinen, mutta lähes aina se on riittävä eikä pelaaja tule kiinnittämään siihen liikaa huomiota. Peili, jossa pelaajan halutaan näkyvän, kannattaa luoda käyttämällä kameraa, joka renderöi peilin näkymän. Sen saa näkyviin render-tekstuuriin. Jos kuitenkin projekti vaatii täysin aidot heijastukset, Screen Space Rendering luo heijastukset aidosti ympäristön mukaan mutta se on raskas prosessoitava ja saatavilla ainoastaan High Definition Render Pipelinessa. Heijastusprobeja kannattaa asettaa niiden ympäristön merkittävien ja suurten objektien lähelle, joiden oletetaan näkyvän heijastuksissa. Pienet objektit, joilla on vahva visuaalinen efekti, esimerkiksi pieni nuotio, saattavat tarvita oman heijastusprobensa. Probejen vaikutusalueet voidaan kuvata kuutioina ympäristössä. Kun käytössä on useita heijastusprobeja, nämä alueet usein menevät osittain päällekkäin ja tällöin Unity valitsee heijastuksen sen proben mukaan, jonka alueella heijastava objekti on eniten. Tätä voi halutessaan vaihtaa muuttamalla probejen tärkeysjärjestystä (eng. *Importance Properties*). Heijastusprobejen heijastuksia voi myös yhdistää toisiinsa (eng. *blending*). Tällöin Unity pehmentää siirtymää probejen välillä. Proben asetuksista voi valita, halutaanko sen heijastavan skyboxia ja halutaanko sen olevan beikattu vai päivittyvän suoritusaikana. (Continisio, 2020, 51:45; Unity Technologies, 2020e).

Kuva 5. Heijastusprobella luotu heijastus ikkunassa.

Toisinaan kaksi heijastavaa pintaa heijastaa toisensa (eng. *Interreflections*), esimerkiksi kaksi peiliä vastakkain luovat loputtomasti heijastumia. Unityssä heijastuksen kimpoilun määrän pystyy määrittämään Environment-asetuksissa olevan Environment Reflections –kohdan Bounces-valinnasta. Koska kimpoilujen määrän lisääminen pidentää vastaavasti aikaa, joka koko heijastuksen beikkaamiseen menee, kannattaa kimpoilujen määrää kasvattaa yhtä suuremmaksi vain silloin, kun tiedetään, että toisiaan heijastavat pinnat ovat selkeästi näkyvissä probeissa. (Unity Technologies, 2016a).

Kun peliobjektit heijastavat ympäristöään, heijastusproben luoman kuutiokartan oletetaan olevan äärettömän kaukana kyseisestä objektista. Tämä yleensä toimii hyvin, mutta varsinkin sisätiloissa saattaa ilmetä ongelmia, kuten kuvassa 6 näkyy (Unity Technologies, 2016). Sisätilan seinät eivät ole äärettömän kaukana heijastavasta peliobjektista. Box size propertyn koko pitää asettaa vastaamaan sisätilan mittoja. (Donzallaz, 2020, 42:00 ; Unity Technologies, 2016a).



Kuva 6. Boxprojectionin luoma ero tilan heijastuksissa. (Unity Technologies, 2016).

3.4 Lampputyypit

Unity 3D:ssä on erilaisia lampputyyppejä, joita lisäämällä voidaan valaista ympäristöä. Näistä neljä yleisintä ovat suunnallinen valo, pistevalo, spottivalo ja aluevalo. Unityssa valoja pääsee lisäämään yläpalkin GameObject –kohdan alta löytyvästä Light -sarakkeesta. (Unity Technologies, 2016b).

Suunallinen valo (eng. *Directional light*) jäljittelee auringonvaloa. Se lähettää säteitä koko näkymään kulmasta, johon se on asetettu. Suunnallisen valon sijainnilla ei ole merkitystä, sillä se valaisee kaikki objektit yhtä kirkkaasti, riippumatta niiden etäisyydestä. (Borromeo, 2020, luku 8).

Pistevalo (eng. *point light*) jäljittelee hehkulamppua. Se lähettää valonsäteitä monisuuntaisesti, ja sen sijainti ja voimakkuus määrittävät, mitkä objektit se valaisee. Mitä kauempana objektit ovat pistevalosta, sitä heikommin se ne valaisee. Pistevalon voimakkuus on kääntäen verrannollinen lähteen neliöön, eli se käyttäytyy niin kuin oikea valo. Pistevalo käyttää kuutiokarttaa valoihin. Tämän takia pistetvalo on raskaampi prosessoitava kuin spottivalo. Esimerkiksi katulampussa kannattaa enemmin käyttää spottivaloa kuin pistevaloa, vaikka pistevalo tuntuisikin realistisemmalta vaihtoehdolta. Kuten kuvasta 7 näkee, spottivalo tuottaa lähestulkoon saman tuloksen kuin pistevalo. (Unity Technologies, 2021, Borromeo, 2020, luku 8).

Spottivalo (eng. s*pot light*) valaisee kartion muotoisen alueen, jossa valo on kirkkaimmillaan kartion kärjessä ja heikkenee sen levetessä. Samoin kuten pistevalon sijainnilla myös spottivalon sijainnilla on väliä. Se eroaa pistevalosta siinä, että sen kulmalla on merkitystä, sillä se määrittää mitkä objektit valaistaan. Spottivalo sopii tilanteisiin, joissa halutaan valaista ainoastaan kohde ympäristöä häiritsemättä. (Borromeo, 2020, Shaban, 2022, osa 6).

Kuva 7. Vasemmalla on käytössä spottivalo ja oikealla pistevalo.

Aluevalo (eng. *area light*) simuloi suorakaiteen muotoista valoa, jossa valonsäteet leviävät tasaisesti kaikkiin suuntiin kyseisen muodon pinta-alalta. Tällä voidaan jäljiteelä esimerkiksi ikkunasta tulevaa valoa. Koska aluevalo on huomattavan raskas prosessorille, sitä ei voi käyttää reaaliajassa vaan ainoastaan beikattuna valokarttaan. Aluevalo valaisee objektin monesta suunnasta yhdellä kertaa, joten lopputulos on yleensä pehmeämpi ja hienovaraisempi kuin muita lampputyyppejä käytettäessä. (Shaban, 2020, osa 6; Unity Technologies, 2020e, luku 9).

Mikä tahansa peliobjekti voidaan myös asettaa valonlähteeksi lisäämällä siihen emissiivinen materiaali. Tätä ei kuitenkaan kannata käyttää ainoana valonlähteenä, koska emissiivisyys näkyy ympäristössä ainoastaan beikattuna. Tämän takia emissiivisyysefektiä voidaan voimistaa lisäämällä emissiivisen pinnan läheisyyteen jokin Unityn lampuista, esimerkiksi pistevalo. (Unity Technologies, 2020e, luku 10).

3.5 Varjot

Varjot ovat monessa projektissa tärkeä osa valaistusta, mutta niiden laskeminen voi olla raskasta. Tämän takia on löydettävä tasapaino suorituskyvyn ja laadun välillä ja optimoitava varjot kohdealustalle sopiviksi. Joskus paras vaihtoehto voi olla jopa luopua kokonaan varjojen käytöstä. (Borromeo, 2020, luku 8).

Unityn varjojen ominaisuuksia pystyy säätämään grafiikka-asetuksista. Unityssä voi valita kovien ja pehmeiden varjojen väliltä. Pehmeät varjot ovat yleensä realistisempia, mutta ne vaativat enemmän prosessointia. Pehmeät varjot on syytä valita käyttöön etenkin silloin, kun projektin varjojen reunat ovat liian pikseliset. Varjojen voimakkuutta pystyy säätämään voimakkuus-kohdasta, ja varjojen laatua pystyy parantamaan resoluutio–kohdasta. (Unity Technologies, 2022s).

3.5.1 Varjokartoitus

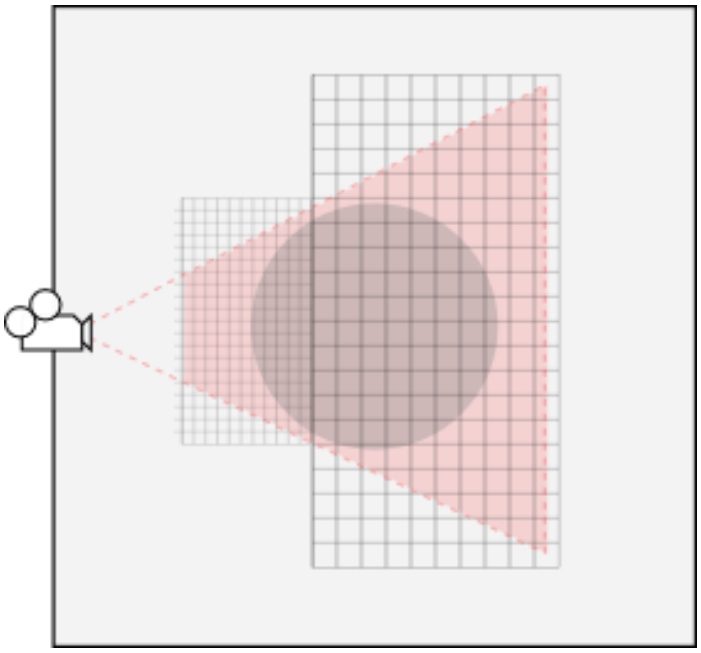
Termejä “varjokartta” ja “valokartta” käytetään toisinaan samassa merkityksessä, mutta Unityssä ne eivät tarkoita samaa. Valokartta voi sisältää värejä, kun taas varjokartta on mustavalkokuva ympäristöstä, joka on renderöity valonlähteen näkökulmasta. Tummat kohdat indikoivat kaukaisempia asioita ympäristössä, kun taas vaaleat varjokartan kohdat ilmaisevat asioita lähellä valonlähdettä. Tämän avulla tiedetään, mille pinnoille valonlähteestä lähtevät säteet osuvat. Luonnollisesti osat, jotka eivät näy valonlähteen näkökulmasta, ovat ympäristössä varjossa. Varjokartan resoluutio määräytyy laatuasetuksista säädettävän

varjoresoluution ja lampputyypin mukaan. Universal Render Pipelinea käytettäessä varjojen resoluutiota vaihdetaan URP-assetin inspektorissa. (Borromeo, 2021, luku 8).

3.5.2 Varjokaskadit

Suunnallista valoa käytetään usein simuloimaan auringonvaloa. Varsinkin suurissa ulkoympäristöissä, joissa pelaaja liikkuu ensimmäisessä tai kolmannessa persoonassa, näköalue on suuri. Kun yksittäinen reaaliaikainen valo valaisee ison alueen tai koko näkymän, syntyy näkymän reaaliaikaisiin varjoihin ongelma, josta käytetään nimitystä *perspective aliasing*. Tämä on sitä, kun kameraa lähellä olevat varjokartan pikselit ovat suuria eli ne näyttävät kulmikkailta ja huonolaatuisilta. Toisin sanoen perspective aliasing johtuu siitä, varjo skaalataan valonlähteen mukaan, mikä usein ei ole samassa kohtaa kameran kanssa. (Continisio, 2020, 4:40; Borromeo, 2021).

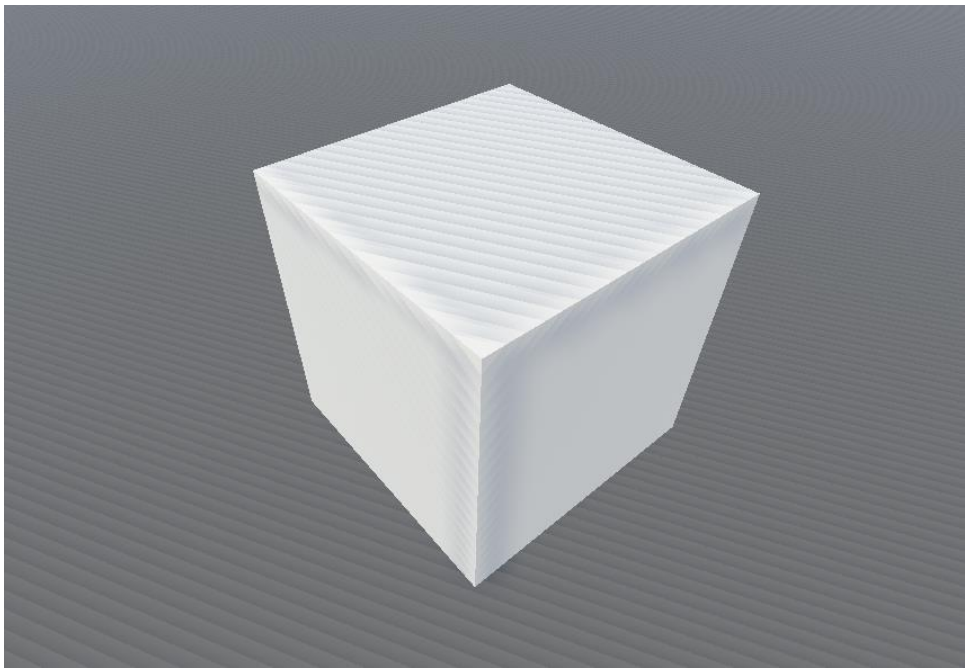
Helpoin ja nopein tapa varjojen laadun parantamiseen on käyttää Unityn pehmeät varjot –valintaa, joka sumentaa varjojen reunat. Varjokartan resoluutiota voidaan myös kasvattaa. Edellä mainitut tavat käyttävät kuitenkin enemmän muistia, ja jos tavoitteena on tehdä mahdollisimman optimoitu peli, on parempi käyttää varjokaskadeja (eng. *shadow cascades*). Varjokaskadeja käytettäessä Unity jakaa kameran näköalueen erikokoisiin osiin, jotka suurenevat kameran näköalueen mukaan. Jokainen osa käyttää periaatteessa samaa varjokarttaa, mutta lähempänä kameraa olevat osat käyttävät pienennettyä versiota varjokartasta, kuten kuvassa 8 (Unity Technologies, 2022) voidaan havaita. Ainoastaan kartan koko pienenee, resoluutio pysyy samana. Kameraa lähempänä olevat varjot ovat tarkkoja eivätkä kauimmaiset varjot ole resoluutioltaan liian suuria. Unityssä voi valita käyttääkö nollaa, kahta vai viittä kaskadia. Mitä enemmän kaskadeja käyttää, sitä tarkempia kameraa lähellä olevat varjot ovat. (Donzallaz, 2020, 42:00; Borromeo 2021, luku 8).



Kuva 8. Kameran näköalue jaettuna kaskadeihin. (Unity Technologies, 2022).

3.5.3 Kallistumat

Joskus varjoja renderöidessä syntyy virheellistä pintojen itsevarjostusta. Tämä johtuu siitä, kun varjokartan pikselien sijainnit lasketaan liian kaukaisiksi. Syynä voi olla esimerkiksi matalaresoluutioinen varjokartta tai suodatus. Tätä kutsutaan varjoakneksi, ja se ilmenee esimekiksi raitoina tai muina kuvioina varjoissa, kuten kuvassa 9. Valokallistumat säätelevät varjoja ja pintojen tummumista, ja niitä säätelemällä varjot saadaan “puhdistettua”. Kallistumia säädettäessä on pidettävä mielessä se, että jotkin ei-halutut varjokuviot näkyvät vain hyvin läheltä katsottuina, ja jos pelaaja ei koskaan pääse niin lähelle pintaa, ei kallistumia tarvitse säätää. (Borromeo, 2021 Continisio, 2020).



Kuva 9. Varjoaknea kuutiossa ja maassa.

3.5.4 Cookiet

Cookie on valon päälle asetettava maski, jolla voidaan muuttaa valon ulkonäköä, väriä ja voimakkuutta niillä pinnoilla, joille se osuu. Cookiet ovat tehokas tapa simuloida monimutkaisia valaistustehosteita ilman raskasta prosessointia. Kuvassa 10 (Unity Technologies, 2022) on cookiella luotu valoefekti. Cookieita käytettäessä on otettava huomioon, että pistevalo tarvitsee kuutiotekstuurin, kun taas muut valot käyttävät 2D-tekstuuria. Cookieita ei voida käyttää suunnallisen valon kanssa, jos käytetään suoraa renderöintiä. Beikatut cookiet saa käyttöön projektiasetusten Editor-kohdasta. Grafiikka-asetuksista löytyy valintaruutu “Enable baked cookies support”. (Unity Technologies, 2022t).

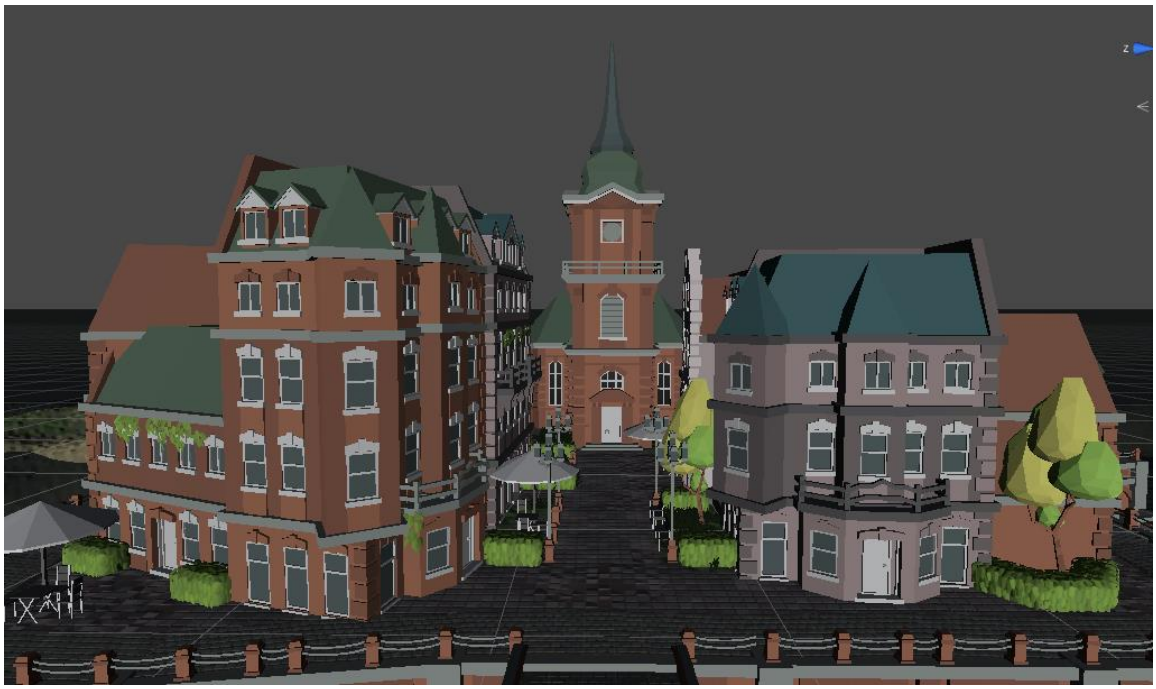
Kuva 10. Valoefekti luotuna Cookiella. (Unity Technologies, 2022).

3.5 Jälkikäsittely

Valaistuksen viimeistely tehdään jälkikäsittelyn avulla. Jälkikäsittely on pääpiirteissään samanlainen URP:in ja HDRP:n kohdalla, mutta HDRP tarjoaa joitakin kehittyneempiä efektejä. Sisäänrakennetun renderöintiputken jälkikäsittely-ominaisuudet ovat URP:n ja HDRP:n ominaisuuksia suppeammat. Jotta päästään jälkikäsittelemään, on aluksi luotava projektin hierarkiaan Global Volume -assetti. Tämän assetin tehtävänä on lisätä jälkikäsittely näkymän kameraan, ja sen asetuksista voidaan myös määrittää, käytetäänkö käsiteltyä aluetta globaalisti, eli kaikkialla ympäristössä, vai vain määritellyllä alueella. Tämän ansiosta voidaan luoda jälkikäsittelyllä esimerkiksi vedenalainen näkymä kameraan silloin, kun kamera on pelissä veden alla. Jotta Global Volume -objektiin päästään lisäämään efektejä, pitää sille ensin luoda profiili Profile-kohdassa. Tämän jälkeen jälkikäsittelyefektejä pääsee lisäämään Add Override -kohdasta. Listan kaikista efekteistä kuvauksineen löytää Unityn dokumentoinnista. (Unity Technologies 2020f).

4 ESIMERKKIVALAISTUKSEN TOTEUTUS

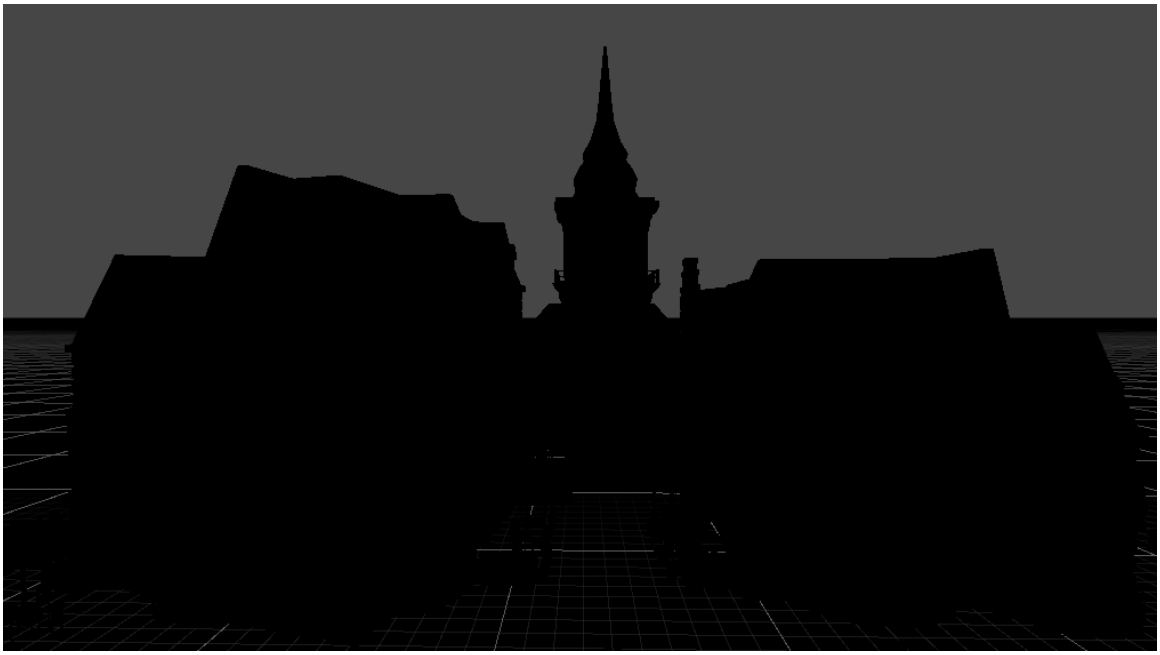
Käytän valaistusesimerkissäni Universal Render Pipelinea, koska se vaikutti parhaiten tarpeisiini soveltuvalta vaihtoehdolta. Tähän ympäristöön en tarvitsisi HDRP:n ominaisuuksia, mutta sisäänrakennetun putken jälkikäsittelyominaisuudet ovat liian suppeat tässä tapauksessa. Valaistavaksi ympäristöksi valitsin Unityn Asset–kaupasta löytämäni LowPolyBrickHouses –paketin, joka on nähtävissä kuvassa 11. Paketti on ilmaiseksi ladattavissa, ja sen tekijä on BrokenVector. Tavoitteeni oli luoda ympäristöön kirkas ja aamuinen valaistus, josta tulee lämmin ja myönteinen tunnelma. Halusin näkymästä sellaisen, mikä omasta mielestäni näyttäisi pelissä mielenkiintoiselta. Lisäksi tavoitteenani oli tehdä ympäristöstä optimoitu ja peliin sopiva.



Kuva 11. Peliympäristö ennen valaistuksen aloittamista.

Koska tätä pakettia ei ole tehty yksinomaan Universal Render Pipelinelle, materiaalit näkyivät kaikki aluksi pinkkeinä. Tämä oli kuitenkin helppo korjata Edit-sarakkeen alta löytyvässä Render Pipeline -kohdassa, missä pystyy päivittämään kaikki projektin materiaalit UniversalRP-materiaaleiksi. Materiaalien päivittämisen jälkeen poistin ympäristöstä kaikki valot, skyboxin sekä asetin ympäristövalon

lähteeksi mustan värin. Lopuksi loin uuden valaistusasetukset–assetin ja otin sen käyttöön. Näin ollen kaikki rakennukset muuttuivat kokonaan mustiksi kuten kuvassa 12 näkyy. Tämä ei ole pakollinen välivaihe, mutta itse pidän niin sanotusti puhtaalta pöydältä aloittamisesta.



Kuva 12. Kaikki valonlähteet poistettuina näkymästä

Aloitin valaistuksen lisäämällä näkymään skyboxin ja vaihtamalla sen ympäristön valonlähteeksi. Seuraavaksi lisäsin suunnallisen valon simuloimaan auringonvaloa. Valitsin valotiloista beikatun, koska ajattelin, että haluan ympäristöstäni optimoidun ja beikattu valo on suoritusaikana sopivampi kuin reaaliaikainen- tai yhdistetty valo. Varjot olivat oletusarvoisesti pehmeät, ja jätin ne sellaisiksi. Kuvassa 13 vasemmalla on näkymä ympäristöstä pelkän ympäristövalon valaisemana ja oikealla ensimmäisen beikkauksen jälkeen, kun suunnallinen beikattu valo vaikuttaa siihen.

Kuva 13. Ennen ja jälkeen ensimmäisen beikkauksen.

Tavoitellessani aamuaurinkoista vaikutelmaa käänsin suunnattua valoani niin että sain ympäristööni pitkiä varjoja. Vaihdoin valon väriä keltaisemmaksi ja lisäsin sen voimakkuutta. Laskin valokartan resoluutiota, jotta renderöintiaika olisi lyhyempi. Huomasin maassa suuria eroja varjojen laaduissa. Vierekkäin oli hyvin tarkka ja sumea varjonraja. Vaihdoin Unityn piirustustilan Baked Lightmappiin, jotta sain beikkaamani valokartat näkyviin ympäristöni pinnoille. Vaihdoin objektien inspektoreista niiden valokarttojen skaalauskertoimia niin, että varjoihin ei enää tullut selkeitä eroja vierekkäisillä pinnoilla. Merkitsin kaikki paikallaan pysyvät objektit staattisiksi.

Lisäsin ympäristöön heijastusproben, koska halusin että talojen ikkunat heijastaisivat ympäristöään. Valitsin beikatun tilan, koska en kokenut, että heijastusten pitäisi päivittyä reaaliajassa. Heijastusproben lisääminen toi ympäristöön lisää valoa, ja päivitin ympäristövaloa ja suunnattua valoa sen mukaisesti. Kuvassa 14 vasemmalla on päivitetty ympäristö.

Seuraavaksi lisäsin ympäristöön valoprobejoukon. Asettelin probeja kaikille alueille, joihin kuvitteellinen pelaaja voisi ympäristössä mennä, kuten kuvassa 14 oikealla näkyy. Loin ympäristöön pelaajaa kuvastavan kapseli-peliobjektin, jolla kokeilin probejoukon toimivuutta ja korjasin sen avulla kohtia, kunnes olin tyytyväinen lopputulokseen. Tämän jälkeen kävin läpi kaikki peliobjektit, jotka olivat kokonaan varjossa. Koska ympäristöni valaistus ei muutu, koin järkeväksi vaihtaa kokonaan varjossa olevat objektit vastaanottamaan Global Illuminationia

valoprobeista eikä valokartoista. Tämän ansiosta muille objekteille jäi enemmän tilaa valokartassa, ja niihin osuvien varjojen ja valojen laatu parani. Tämän jälkeen lisäsin vielä hieman keltaisuutta suunnattuun valooni ja beikkasin näkymän jälleen.

Kuva 14. Pitkät varjot ja valoprobet ympäristössä.

Lopuksi lisäsin jälkikäsittelynä bloomia ja säädin valkotasapainoa. Muutin näkymän tummimpia kohtia hieman punertavimmiksi, koska se toi mielestäni kuvaan lämpöä ja teki ympäristöstä mielenkiintoisemman. Olin tyytyväinen lopputulokseeni, sillä pehmeät, pitkät varjot ja lämpimät värit loivat mielestäni tavoittelemaani aamuista tunnelmaa. Kokeilin myös lisätä ympäristööni sumua, mutta mielestäni se ei sopinut tähän tilanteeseen. Lopuksi kasvatin valokartan ja heijastusten resoluutiota viimeistä beikkausta varten, jotta lopputuloksessa olisi mahdollisimman hyvälaatuiset varjot ja heijastukset. Kuvassa 15 on nähtävänä ympäristön lopullinen valaistus.

Kuva 15. Ympäristö jälkikäsittelyn jälkeen.

Tein ympäristöstä myös öisen version tummentamalla skyboxia ja vaihtamalla kirkkaan ja keltaisen suunnallisen valon himmeämpään sinertävään valoon simuloimaan kuuta. Vaihdoin myös jälkikäsittelyasetuksiani niin, että lopputulos olisi öisen sinertävä. Öiseen maisemaan sumu sopi mielestäni hyvin, joten vaihdoin sen päälle valaistusasetusten ympäristö-sarakkeesta. Lisäsin ympäristön valopylväisiin valot. Valitsin spottivalot niihin, jotka valaisevat vain alaspäin ja pistevalot niihin, joiden piti valaista myös ympärilleen, esimerkiksi talojen seiniä. Asetin nämä valot beikatuiksi ja päivitin valoprobejoukkoa, niin että ne toimivat oikein muuttuneen valaistuksen kanssa.

Halusin tehdä ympäristössä olevasta hevospatsaasta mielenkiinnon kohteen pelaajalle. Patsaan päävalona toimii alhaaltapäin tuleva spottivalo. Vieressä olevaan valopylvääseen lisäsin pistevalon, koska halusin sen pehmentävän hevospatsaan varjoja maan valaistuksen lisäksi. Sinertävä ympäristövalo luodaan tummennetun skyboxin mukaan ja toimii mukavana vastavärinä lamppujen kellertävälle valolle. Lisäsin vielä pienen spottivalon tuomaan valoa hevosen ja

ratsastajan kasvoihin, vaikka niihin ei realistisesti tulisi tässä asetelmassa niin paljon valoa. Kuvassa 16 on valaistu patsas.

Kuva 16. Valaistu patsas.

POHDINTA

Koska Unity kehittää valaistustyökalujaan jatkuvasti, huomasin, että monet kirjat sisälsivät osittain tai kokonaan vanhentunutta tietoa. Tämän takia päädyin käyttämään lähteenä usein Unity Documentationia. Aluksi vältin YouTube –videoiden käyttämistä lähteenä, koska epäilin niiden luotettavuutta. Pystyin kuitenkin itse tarkistamaan niissä esitetyt väitteet Unityssä.

Valaistus Unityssä saattaa tulevaisuudessa tulla muuttumaan kilpailevien pelimoottorien valaistustekniikoiden kehittyessä. Esimerkiksi Unreal Enginen Lumen -valaistusjärjestelmän voisi sanoa toimivan suorana kilpailijana Unityn säteenseurannalle, tuottamalla vastaavaa valaistuslaatua olematta yhtä raskas prosessoitava. Muutenkin pelialustojen kehittyminen tulee näkymään valaistuksen luomisessa, kun valaistus pystyy olemaan monimuotoisempaa kuin nyt.

Valaistus on merkittävä osa peliympäristöä ja jo ennen valaistuksen aloittamista koin tärkeäksi kerätä ylös kaikki ominaisuudet ja tehtävät mitä valoilta haluttaisiin. Koin sen helpottavan itse valaistusprosessia. Valojen lisäksi peliympäristössä on huomioitava myös varjot, sillä pelaaja ei ole aina pelkästään ympäristön valoisissa osissa. Pelinkehityksessä valaistus pitää suunnitella niin, että se näyttävät hyvältä kaikista suunnista.

Vaikka renderöintiputket eivät varsinaisesti ole osa valaistusta, koin niiden erojen ymmärtämisen tärkeäksi. Vertaillessani eri renderöintiputkia, URP osoittautui mielestäni parhaiten soveltuvaksi indietiimien pelinkehitykseen. Sillä on monipuolisimmat kohdealustat, ja siitä löytyy kaikki ominaisuudet, joita pieni kehitystiimi tarvitsee. URP:lla on suurin osa HDRP:n ominaisuuksista, mutta sitä on karsittu siten että se on suorituskykyisempi. Opinnäytetyön yhteydessä teetettiin sähköpostikysely suomalaisille pelitaloille siitä, mitä renderöintiputkea he käyttävät peliprojekteissaan. URP oli selkeästi suosituin vaihtoehto, sillä kaikki vastanneet, jotka käyttivät Unityä, kertoivat käyttävänsä URP:tä.

LÄHTEET

Birn J. 2013, Digital Lighting and Rendering

Borromeo, N. A. 2020, Hands-On Unity 2020 Game Development

Borromeo, N. A. 2021, Hands-On Unity 2021 Game Development - Second Edition

Continisio C. 2020. Harnessing Light with URP and the GPU Lightmapper | Unite Now 2020. Lataaja Unity. Viitattu 1.2.2022 Harnessing Light with URP and the GPU Lightmapper | Unite Now 2020, https://www.youtube.com/watch?v=hMnetI4-dNY&t=2585s&ab_channel=Unity

Courrèges A. 2015, GTAV – Graphics Study, luettu 4.2.2022 https://www.adriancourreges.com/blog/2015/11/02/gta-v-graphics-study/

De Bock S. 2013, Introduction to the Graphics Pipeline, luettu 4.2.2022 https://www.gamedev.net/tutorials/programming/graphics/introduction-to-the-graphics-pipeline-r3344/

Lighting Tutor, n.d. Kelvin Color Temperature Scale, https://www.lightingtutor.com/kelvin-color-temperature/

Mortensen J. 2019, blogikirjoitus, luettu 5.2.2022. https://blog.unity.com/technology/enlighten-will-be-replaced-with-a-robust-solution-for-baked-and-real-time-giobal

Pluralsight 2014 Light Up Your World: How Lighting Makes All the Difference for Games. https://www.pluralsight.com/blog/film-games/understanding-the-importance-of-lighting-for-games

Shaban K. 2022, Master Lighting in Blender 3D | #Series2 Julkaisija, SkillShare. Viitattu 5.3.2022. https://www.skillshare.com/classes/Master-Lighting-in-Blender-3D-Series2/666895490

Unity Technologies, 2016a, Advanced Reflection Probe Features, luettu 1.3.2022. https://docs.unity3d.com/530/Documentation/Manual/AdvancedRefProbe.html

Unity Technologies, 2016b, Types of ligh, luettu 2.3.2022. https://docs.unity3d.com/540/Documentation/Manual/Lighting.html

Unity Technologies, 2020a Universal Render Pipeline Asset, luettu 9.4.2022. https://docs.unity3d.com/Packages/com.unity.render-pipelines.universal@7.1/manual/universalrp-asset.html

Unity Technologies, 2020b, Getting started with ray tracing, luettu 7.12.2021. https://docs.unity3d.com/Packages/com.unity.render-pipelines.high-definition@7.1/manual/Ray-Tracing-Getting-Started.html

Unity Technologies, 2020c, Ray-Traced Global Illumination, luettu 7.12.2021. https://docs.unity3d.com/Packages/com.unity.render-pipelines.high-definition@7.1/manual/Ray-Traced-Global-Illumination.html

Unity Technologies, 2020d, Mask and detail maps, luettu 7.12.2021. https://docs.unity3d.com/Packages/com.unity.render-pipelines.high-definition@10.2/manual/Mask-Map-and-Detail-Map.html

Unity Technologies, 2020e Introduction to Lighting and Rendering, luettu 14.5.2022. https://learn.unity.com/tutorial/introduction-to-lighting-and-rendering#

Unity Technologies, 2020f, Post Processing in the Universal Render Pipeline, luettu 3.3.2022. https://docs.unity3d.com/Packages/com.unity.render-pipelines.universal@7.1/manual/integration-with-post-processing.html

Unity Technologies, 2021 Deferred Rendering in URP, luettu 3.5.2022, 28.5.2022. https://docs.unity3d.com/Packages/com.unity.render-pipelines.universal@12.0/manual/rendering/deferred-rendering-path.html

Unity Technologies, 2022a Render Pipelines, luettu 2.1.2022. https://docs.unity3d.com/Manual/render-pipelines.html

Unity Technologies, 2022b SRP Core, luettu 1.4. 2022. https://docs.unity3d.com/Packages/com.unity.render-pipelines.core@12.1/manual/index.html

Unity Technologies, 2022c Render Pipelines, luettu 1.4. 2022. https://docs.unity3d.com/Manual/render-pipelines-overview.html

Unity Technologies, 2022d CommandBuffer, luettu 1.1.2022. https://docs.unity3d.com/Manual/GraphicsCommandBuffers.html

Unity Technologies, 2022e Universal Render Pipeline, luettu 1.1.2022. https://docs.unity3d.com/Packages/com.unity.render-pipelines.universal@13.1/manual/index.html

Unity Technologies, 2022f Comparing URP and build in Render Pipeline, luettu 3.5.2022. https://docs.unity3d.com/Packages/com.unity.render-pipelines.universal@13.1/manual/universalrp-builtin-feature-comparison.html

Unity Technologies, 2022g Universal Additional Data, luettu 4.5.2022. https://docs.unity3d.com/Packages/com.unity.render-pipelines.universal@14.0/manual/lighting.html

Unity Technologies, 2022h Universal Renderer, luettu 4.5.2022. https://docs.unity3d.com/Packages/com.unity.render-pipelines.universal@12.1/manual/urp-universal-renderer.html

Unity Technologies, 2022i Shading models in Universal Render Pipeline, luettu. 1.5.2022 https://docs.unity3d.com/Packages/com.unity.render-pipelines.universal@7.1/manual/shading-model.html

Unity Technologies, 2022j Shadiers and Materials, luettu 1.5.2022. https://docs.unity3d.com/Packages/com.unity.render-pipelines.universal@14.0/manual/shaders-in-universalrp.html

Unity Technologies, 2022k Shader Graph, luettu 9.4.2022. https://unity.com/features/shader-graph

Unity Technologies, 2022l High Definition Render Pipeline overview, luettu 10.4.2022. https://docs.unity3d.com/Packages/com.unity.render-pipelines.high-definition@12.1/manual/index.html

Unity Technologies, 2022m, Lighting Settings Asset, luettu 10.5.2022. https://docs.unity3d.com/Manual/class-LightingSettings.html

Unity Technologies, 2022n, Lighting Mode: Baked Indirect, l uettu 6.4.2022. https://docs.unity3d.com/Manual/LightMode-Mixed-BakedIndirect.html

Unity Technologies, 2022o, Lighting Mode: Substractive, luettu 6.4.2022. https://docs.unity3d.com/Manual/LightMode-Mixed-Subtractive.html

Unity Technologies, 2022p, Lighting Mode: Shadowmask, luettu 6.4.2022. https://docs.unity3d.com/Manual/LightMode-Mixed-Shadowmask.html

Unity Technologies, 2022q Inspector, luettu 20.4.2022. https://docs.unity3d.com/Manual/UsingTheInspector.html

Unity Technologies, 2022r Light Modes, luettu 1.1.2022. https://docs.unity3d.com/Manual/LightModes.html

Unity Technologies,2022s, Lights, luettu 4.5.2022. https://docs.unity3d.com/Manual/class-Light.html

Unity Technologies 2022t Cookies, luettu 1.5.2022. https://docs.unity3d.com/Manual/Cookies.html