OULUN AMMATTIKORKEAKOULU

Petri Launimaa

REAALIAIKAISEN SÄTEENSEURANTATEKNOLOGIAN HYÖDYNTÄMINEN UNREAL ENGINE -PELIPROJEKTISSA.

REAALIAIKAISEN SÄTEENSEURANTATEKNOLOGIAN HYÖDYNTÄMINEN UNREAL ENGINE -PELIPROJEKTISSA.

Petri Launimaa
Opinnäytetyö
Syksy 2021
Tietojenkäsittely
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietojenkäsittely

Tekijä(t): Petri Launimaa
Opinnäytetyön nimi: Reaaliaikaisen säteenseurantateknologian hyödyntäminen Unreal Engine -peliprojektissa.
Työn ohjaaja(t): Matti Viitala
Työn valmistumislukukausi ja -vuosi: Syksy 2021 Sivumäärä: 37 + 2 liitettä

Tässä opinnäytetyössä käsiteltiin säteenseurannan tuen lisäämistä Unreal Engine -peliprojektiin, sen tuomia hyötyjä ja sen mahdollistamia uusia mekaniikoita. Työssä analysoitiin hieman sitäkin, kuinka säteenseurannan lisääminen vaikutti ruutunopeuksiin ja pelattavuuteen. Lisäksi käsiteltiin demosta kerättyä statistiikkaa ja muodostettiin sen avulla kaavioita, jotka löytyvät liitteestä 2.

Tavoitteena ollut pelidemon rakentaminen onnistui kohtalaisen hyvin: säteenseurantaan pohjautuvia efektejä saatiin lisättyä peliprojektiin, mutta pelidemossa säteenseuranta ei vielä toiminut pelimekaniikkana. Toinen tavoite eli ruutunopeuksien analysointi täyttyi suunnitellusti, ja tiedoista saatiin muodostettua kaavioita.

Asiasanat: videopelit, peliohjelmointi, peligrafiikka

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Business Administration, Information Technology

---

Author(s): Petri Launimaa
Title of thesis: Utilizing real time raytracing as a part of an Unreal Engine game project
Supervisor(s): Matti Viitala
Term and year when the thesis was submitted: Autumn 2021
Number of pages: 37 + 2 appendices

This thesis was about utilizing raytracing support in an Unreal Engine game project and about considering its benefits and the new game mechanics it enables, and about shortly analyzing how enabling raytracing support affected frame rates and gameplay. Additionally, gathering all the statistics from the demos that ran and formed graphs of them included in appendix 2.

The goal of developing a game demo was met moderately, there were ray-traced effects in the demo, but they weren't enough of a game mechanic on their own. The secondary goal, which was analyzing the framerates was met successfully and the data could be formed into graphs.

Keywords: videogames, game programming, game graphics

SISÄLLYS

KESKEISET KÄSITTEET

Arcade

Tässä opinnäytetyössä arcade-pelillä tarkoitetaan peliä, jossa edistyminen ei ole pysyvää, vaan pelimaailma nollautuu jokaisen yrityksen jälkeen.

Denoiser

Kohinan poistoon käytetty algoritmi. Nykyään useat denoiserit ovat tekoälypohjaisia, kuten Nvidian OptiX ja Intelin OpenImageDenoise.

Direct Illumination

Valaistuksessa käytetty termi, joka tarkoittaa suoraa valaistusta, jossa valoa ei heijasteta ja hohtavia pintoja ei ole olemassa.

Global Illumination

Global illumination on nimensä mukaisesti valaistukseen käytetty algoritmi, jonka mukaisesti heijastetaan valoa scenessä olevien objektien pinnoista suoran valaistuksen (direct illumination) lisänä.

Kohina

Kuvassa näkyvä epätasaisuus. Toisinaan liian puhtaaseen kuvaan voidaan haluta lisätä kohinaa, jotta kuva voidaan helpommin yhdistää oikeilla kameroilla otettuihin kuviin.

LOD

3D-mallien yksinkertaistetut versiot, Level Of Detail -mallit, auttavat pelimoottoria piirtämään vain tarpeelliset geometriat kohteen täyttäessä pienemmän osan ruudusta.

Nvidia RT Core, AMD Ray Accelerator

Näytönohjainsiruvalmistajien brändinimet säteenseurannan laitteistokiihdytyksen mahdollistaville sirun osille.

Offline renderer

Filmiteollisuudessa ja esimerkiksi tuotekuvaamisessa käytettävä kuvan tuottamiseen tarkoitettu renderöijä, joka tuottaa yksittäisiä kuvia ja joka mahdollistaa monimutkaisempien algoritmien hyödyntämisen kuin reaaliaikaiset vastineet. Esimerkkejä: Blender Cycles, OTOY Octane, Arnold, Pixar RenderMan.

PBR

PBR eli Physics Based Rendering eli fysiikkaperusteinen renderöinti. Joukko renderöintitekniikoita, jotka perustuvat fysiikan mallinnukseen.

Pelattava ruutunopeus

Erittäin subjektiivinen määrite, joka riippuu pelin tahdista, nopeammissa peleissä, esimerkiksi toimintapeleissä nykymieltymys on yleisesti yli 120 ruutua sekunnissa, kun taas hitaammille peleille 60 ruutua sekunnissa on sopiva ruutunopeus. Hitaimmille tarinapeleille jopa 30 tai 24 ruutua sekunnissa voi sopia (Gapo 2021). Tässä opinnäytetyössä tavoitteeksi on asetettu 60 ruutua sekunnissa keskitasoisella näytönohjaimella.

Pelisilmukka

Pelin logiikan silmukka, jossa pelin esimerkiksi pelitason voittaminen johtaa pelimaailman nollaamiseen ja uudelleenaloitukseen.

Post-Processing

Post-Processing (suom. jälkikäsittely) tarkoittaa kuvan jälkikäsittelyä. Kuvaa käsitellään varsinaisen varjostin- ja rasterointiosuuden renderöimisen jälkeen. Esimerkkeinä jälkikäsittelystä voidaan mainita Bloom- (suom. hehku) ja Depth of Field (suom. syväterävyys) -efektit.

Rajapinta

Esimerkkejä ovat DirectX 11 ja DirectX 12. Muita rajapintoja ovat esimerkiksi Open GL ja Vulkan. Rajapinta sitoo ohjelmiston ja laitteiston yhteen.

Ray Casting

Säteenseurantaan pohjautuva tapa mitata etäisyyksiä 3D-ympäristössä.

Raytracing

Raytracing eli säteenseuranta tarkoittaa valonsäteiden heijastamisien ja risteämisien simulointia 3D-ympäristössä.

Real Time Renderer

Reaaliaikaista kuvaa tuottava renderöijä, joka käyttää yleensä yksinkertaisia algoritmeja ja täten parantaa renderöintinopeutta pyrkien reaaliaikaiseen renderöintiin.

RTX

Näytönohjainsiruvalmistaja Nvidian-brändi säteenseurannan laitteistokiihdytykselle.

Sample

Sample (suom. näyte) on polunseurantateknologioihin liittyviin tekniikoiden kanssa käytetty termi. Pikseleitä ”samplaamalla” saadaan aikaan kuva, jossa on vähemmän kohinaa. Myös jotkin Post-processing-efektit ”samplaavat”.

Scene

3D-grafiikan tuottamisessa yleisesti käytetty termi, joka viittaa kuvattaviin esineisiin ja niiden miljööseen. Voi sisältää myös esimerkiksi HDRi-valaistuskartan, joka simuloi ulkoista valaistusta.

Screen space

Koordinaatisto, jota käytetään 2D-kuvalle ja joka on tuotettu rasteroimalla 3D-ympäristö ja jota käytetään muun muassa screen space -heijastusten tekemiseen sekä joihinkin valaistuksen efekteihin. Hyödyntää 3D-kuvan tuottamisessa tehtyä syvyyskarttaa

Shader

Shaderit eli varjostimet ”värjäävät” tietyn verteksin tai pikselin.

Shader call

Shader callit aloittavat shaderin eli varjostimen prosessoinnin. Mitä enemmän shader calleja pitää tehdä, sitä pitempään ruudun renderöiminen kestää.

1 JOHDANTO

Opinnäytetyön tarkoituksena on pohtia ja testata, kuinka helppoa on säteenseurannan hyödyntäminen Unreal Engine -peliprojektissa, sekä valjastaa se pelimekaniikaksi. Pelimekaniikkana toimii tässä vain yksinkertainen peili, jonka avulla pelaaja löytää kohteensa. Säteenseuranta on kirjoitushetkellä suosittu aihe tekniikasta kiinnostuneiden parissa, ja siruvalmistajien markkinointi keskittyy kirjoitushetkellä yleisesti tähän aiheeseen.

Säteenseurannasta voi tulla merkittävä osa tulevaisuuden pelikehitystä, ainakin ensisijaisesti kaikkein näyttävimmissä peleissä, joissa pelin ulkoasu on tärkeämpää kuin esimerkiksi nopeampitempoisissa kilpailuhenkisissä peleissä, joista kaikki ylimääräinen karsitaan pois. Seuraavissa luvuissa käsitellään säteenseurannan taustaa ja kohinanpoiston tärkeyttä reaaliaikaisen säteenseurannan toteuttamisessa sekä toteutetaan pelidemo. Pelidemo toimii statistiikan keräystyökaluna, jolla vertaillaan lyhyesti eri korttisarjojen suorituskykyä.

2 SÄTEENSEURANNAN TAUSTAA

Fotorealististen kuvien tuottaminen on yksi tietokonegrafiikan suurimmista päämääristä. Eri renderöintiohjelmat tuottavat kuvia eri tavoilla, Näistä säteenseurantaan pohjautuvat tavat ovat tyypillisimpiä, kun halutaan tuottaa kuva, jota voidaan pitää käytännössä fotorealistisena. Kun kuvia halutaan tuottaa esimerkiksi elokuviin tai mainontaan, käytetään yleensä offline-renderöijiä, joissa kuvan tuottamiseen kulutettu aika ei ole prioriteettina (Keller 2015).

Säteenseurannan pohjaidea juontuu jo vuodelle 1525, jolloin Albrecht Dürer esitteli sen alkeita (Hofmann 1990). Tekniikkaa on vasta viime aikoina päästy hyödyntämään grafiikan renderöimisessä laitteiston laskentatehon kehittyessä.

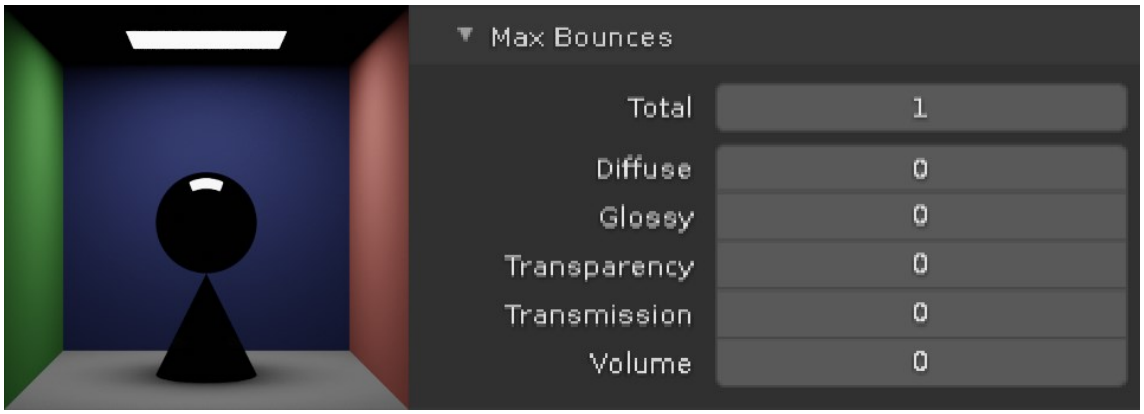
2.1 Säteenseuranta ja polunseuranta

Säteenseurannalla simuloidaan valonsäteiden heijastamisia ja risteämisiä 3D-ympäristössä. Säteenseuranta pohjaa kaiken säteiden heijastamisien ja risteämisen laskennan. Se toimii perustana polunseurannalle, ja se on luontainen ratkaisu valaistuksen ja heijastusten renderöintiin (Liu 2018).

Polunseuranta on säteenseurantaa hyödyntävä tekniikka, jossa hyödynnetään kehittyneitä valaisualgoritmeja, kuten esimerkiksi Monte Carloa. Jokaisen heijastuvan säteen suunta ja tyyppi arvotaan, ja kun näitä otteita saadaan tarpeeksi aikaan, niistä saadaan muodostettua kuva, jossa kappaleiden väliset säteet otetaan huomioon (Lafortune & Yves 1993).

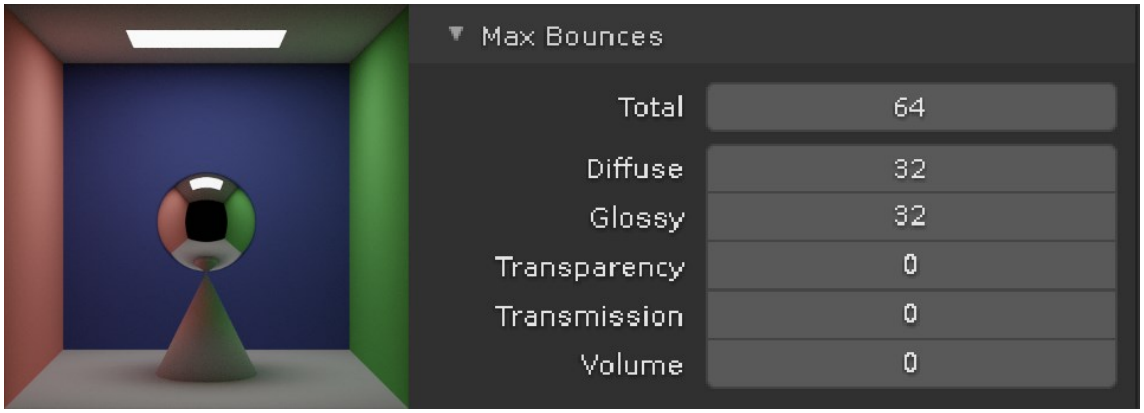
2.2 Säteenseurannan esimerkkejä

Ensimmäinen esimerkki simuloi yksinkertaistetusti pelkkää säteenseurantaa, kun taas toinen esimerkki simuloi kehittyneempää polunseurantaa. Kuvioissa 1 ja 2 esitetään ilmaisen ja vapaan lähdekoodin Blender-ohjelmistolla simuloiden, miltä pelkkä direct illumination -säteenseuranta ja miltä kehittyneempi global illumination -polunseuranta näyttävät. Kummatkin esimerkit käyttävät 512 näytettä per pikseli (ilman kohinanpoistoa). Kuviossa 1 simuloidaan direct illumination -tekniikkaa. Valo heijastuu vain yhdellä heijastuksella. Huomaa, kuinka katosta ei heijastu mitään, jolloin se näyttää pimeältä, ja kuinka pallo ei heijasta mitään muuta kuin valonlähteen eikä pallon alla oleva kartio saa valoa pallon läpi.



KUVIO 1. Direct illumination säteenseuranta, ei heijastuksia.

Kuviossa 2 simuloidaan global illumination -tekniikaa. Kuviossa 2 esitetään 32 heijastusta diffuse- ja glossy- säteille Niitä on yhteensä 64. Huomaa, kuinka katto ja kartio heijastavat nyt valoa, ja pallo heijastaa ympäristöä.



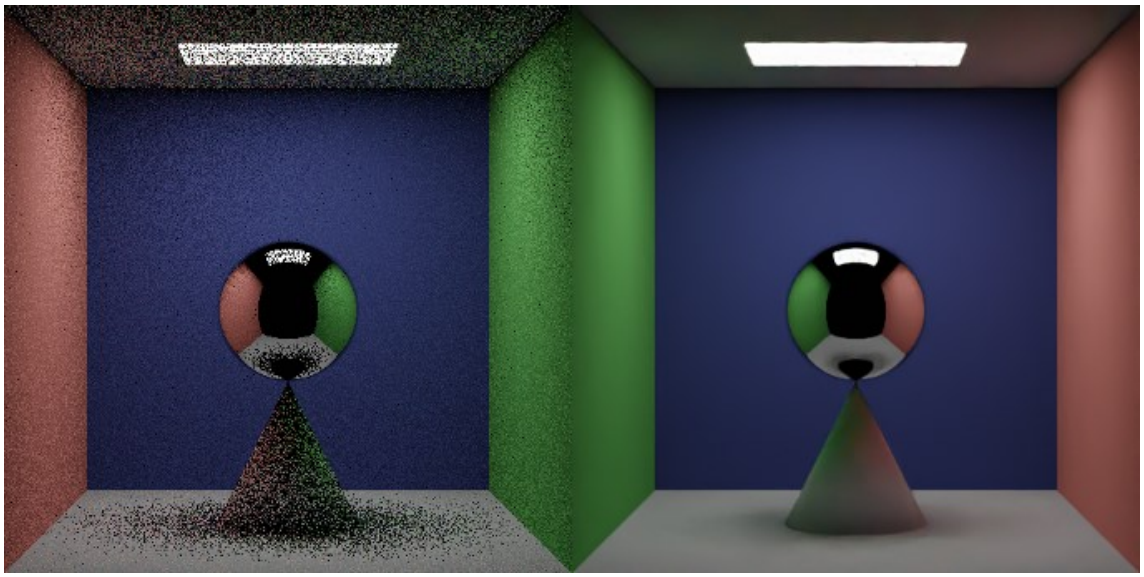
KUVIO 2. Global illumination -polunseuranta, 64 heijastusta.

2.3 Reaaliaikainen säteenseuranta

Realliaikaisia säteenseurantaan pohjautuvia efektejä on jo aiemmin käytetty videopeleissä: screen space -heijastuksilla alettiin kokeilla yksinkertaista säteenseurannan hyödyntämistä syvyyskartan avustamana, mutta reaaliaikaiset säteenseurantaan pohjautuvat tekniikat ovat tulleet ajankohtaiseksi vasta viime aikoina, kun laitteistokiihdytyksen mahdollistavat näytönohjaimet ovat tulleet markkinoille ja niitä tukevia rajapintoja on alettu kehittää. Näissäkin esimerkeissä on lähinnä vain yhden tyyppistä säteenseurantaa, kuten heijastuksia, eikä koko kuvaa ole tuotettu säteenseurantatekniikoin, vaan käytössä on hybridirenderöintitekniikka, joka mahdollistaa säteenseurannan efektien liittämisen perinteisellä rasterointitekniikalla tuotettuun kuvaan.

Säteenseurantatekniikoiden ohjelmistoemulointia on jo aiemmin nähty joissain peleissä. Esimerkkinä tästä mainittakoon Minecraft-pelin useat kolmannen osapuolen lisäosat, jotka lisäävät screen space -heijastukset ja screen space global illumination -valaistuksen. Eri peleissä on käytössä eri efektejä. Niistä jotkin lisäsivät vain heijastukset tai vain varjostuksen, mutta jotkin pelit lisäsivät useampia efektejä. Reaaliaikaisuuden ylläpitämiseksi pelimoottorit renderöivät säteenseurannan yleensä matalammalla resoluutiolla ja näytemäärällä ja käsittelemällä tuloksen kehittyneillä kohinanpoistoalgoritmeilla (Liu 2018).

Kohinanpoisto on tärkeässä osassa erityisesti reaaliaikaisessa säteenseurannassa. Kohinanpoistolla voidaan usein nopeuttaa koko renderöintiprosessia ja samalla parantaa kuvanlaatua ilman näytemäärän nostamista. Kuviossa 3 esitellään simuloitua versiota kohinanpoistosta, joka on toteutettu Blender-ohjelmistolla. Vasemmanpuoleinen osa kuviossa 3 on tuotettu 1 näytteellä per pikseli ja 1 heijastuksella. Oikeanpuoleinen osa kuviosta 3 on samalla tavalla tuotettu kuva, mutta se on käsitelty OpenImageDenoise-algoritmilla. Aiempi 512 näytteen ja 64 heijastuksen kuvion 2 renderöinti kesti testilaitteistolla noin 8 sekuntia. Kuvion 3 vasemmanpuolimmaisen kuvan renderöinti kesti 0,08 sekuntia eli tapahtui sata kertaa nopeammin, kuin kuvion 2 kuvan renderöinti. Kuvion 3 oikeanpuolimmaisen kuvan renderöintiin kului yhteensä 0,4 sekuntia eli tapahtui 20 kertaa nopeammin, kuin kuvion 2 kuvan renderöinti.



KUVIO 3. Kohinanpoisto Blender-ohjelmistossa.

2.4 Säteenseurannan laitteistokiihdytys

Säteenseurantaa voidaan joko kiihdyttää ohjelmistolla, kuten näytönohjaimilla esimerkiksi CUDA:lla tai sille tarkoitetulla laitteistokiihdytyksellä (Luebke & Parker 2008). DirectX 12 -rajapinta mahdollisti säteenseurannan joko laitteisto- tai ohjelmistokiihdytyksellä, mutta DirectX 12 Ultimate -rajapinta vaatii laitteistokiihdytyksen. Säteenseurannan laitteistokiihdytyksen toteutustavat riippuvat laitteiston siruvalmistajasta. Suurimassa osassa tämän hetken suosituimmista näytönohjaimista ei vielä ole laitteistokiihdytystä (Valve Software 2021).

Nvidia ei ole juurikaan selittänyt piiriensä toimintatapoja, mutta julki on tullut ainakin seuraava: RT Coret ovat prosessointiytimiä, joiden tarkoitus on kiihdyttää säteenseurannan laskentaa erillään perinteisemmistä shader-ytimistä. Rinnakkain toimivat shader callit menevät ensin säteenseurantaytimen läpi ja palaavat shaderille, minkä jälkeen pelimoottorista riippuen kuva käsitellään poistamalla kuvasta kohina esimerkiksi koneoppimispohjaisella jälkikäsittelyvaiheella. Reaaliaikaisen kuvan tuottaminen useammalla näytteellä on vielä pääosin mahdotonta. (GamersNexus 2020).

AMD:n RDNA2-alusta tukee DXR 1.1 -rajapintaa. Säteenseurantakiihdyttimistä ei kerrota paljoa, mutta AMD:n toteutustapa näyttää olevan sulaututetumpi. AMD julkaisi myös oman kohinanpoistajan RDNA2-alustalle (Advanced Micro Devices 2020b).

3 PELIMOOTTORI JA SÄTEENSEURANNAN TAUSTATEKNIIKKA

3.1 Unreal Engine

Unreal Engine syntyi Doomin ja Quaken inspiroimana vuonna 1995 ”Unreal”-peliä varten. Siitä myöhemmin kehittyi yksi suosituimmista pelimoottoreista. Vuonna 2014 Epic Games julkaisi uudelleen Unreal Engine 4:n uudella lisenssimallilla, joka pohjautui kuukausimaksuun sekä 5 prosenttiin myytyjen pelien voitosta rojalteina (Nutt 2014).

Myöhemmin pelimoottorin käyttö muuttui muuten ilmaiseksi, paitsi että Epic vaatii 5 prosentin rojaltit yli 3 000 dollaria vuosineljänneksessä tuottavilta peleiltä, ja uusimpana muutoksena 5 prosentin rojalti vaaditaan, jos tuotot ovat yhteensä yli miljoonan dollarin. Vähemmän kuin miljoona dollaria tuottaneilta peleiltä ei vaadita rojalteja. Epic Games myös päätti, että heidän oman Epic Games Storen kehittyessä kauppapalvelussa julkaistavien Unreal Engine -pohjaisten pelien rojaltit ovat osana 12 prosentin kustannuksia, kun pelejä julkaistaan kauppapalvelussa (Wilde 2020).

Reaaliaikainen säteenseuranta lisättiin Unreal Engineen versiossa 4.22, joka julkaistiin huhtikuussa 2019. Versioon lisättiin säteenseurantateknologialla tehostetut global illumination, -heijastukset, heijastusten taittuminen, varjostus ja ambient occlusion -efektit (Epic Games 2019). Myöhemmissä versiossa ominaisuuksien suorituskykyä ja vakautta on paranneltu. Unreal Enginen versiossa 4.25 säteenseurannan tuesta tuli virallinen (”Production ready”). Tämä tarkoitti käytännössä sitä, että sen toimintaan liittyvät viat olivat pääosassa korjattuja, ja sitä alettiin tukea virallisesti (Epic Games 2020).

Unreal Engine 5 julkistettiin 13.5.2020, ja se sisältää uuden dynaamisen valaistus- ja global illumination -ratkaisun nimeltään ”Lumen”. Lumeniin sisältyy myös heijastusten kiihdytys. Yhtenä merkittävänä osana uudessa moottorissa on ”Nanite”-niminen ominaisuus, joka mahdollistaa raskaiden ja erittäin tarkkojen 3D-mallien käyttämisen ilman etukäteen tapahtuvaa mallien yksinkertaistusta (LOD). Moottori automaattisesti ja reaaliaikaisesti skaalaa malleja sopiviksi (Laine 2020).

3.2 DirectX 12 (DXR) ja muut rajapinnat

Windows 10:n lokakuun 2018 versiosta lähtien Microsoftin DirectX 12.1:n ohjelmistorajapinnassa on uutuutena DirectX Raytracing (DXR), joka mahdollistaa säteenseurannan heijastusten laskennan DirectX -rajapinnan avulla. Tämä on toteutettu yhdessä näytönohjainvalmistajien kanssa, ja valmistajasta riippuen se käyttää laitteistoa eri tavoin. Yleisimmillään tämä mahdollistaa säteenseurannan efektien toteuttamisen laitteistolla ohjelmiston sijaan.

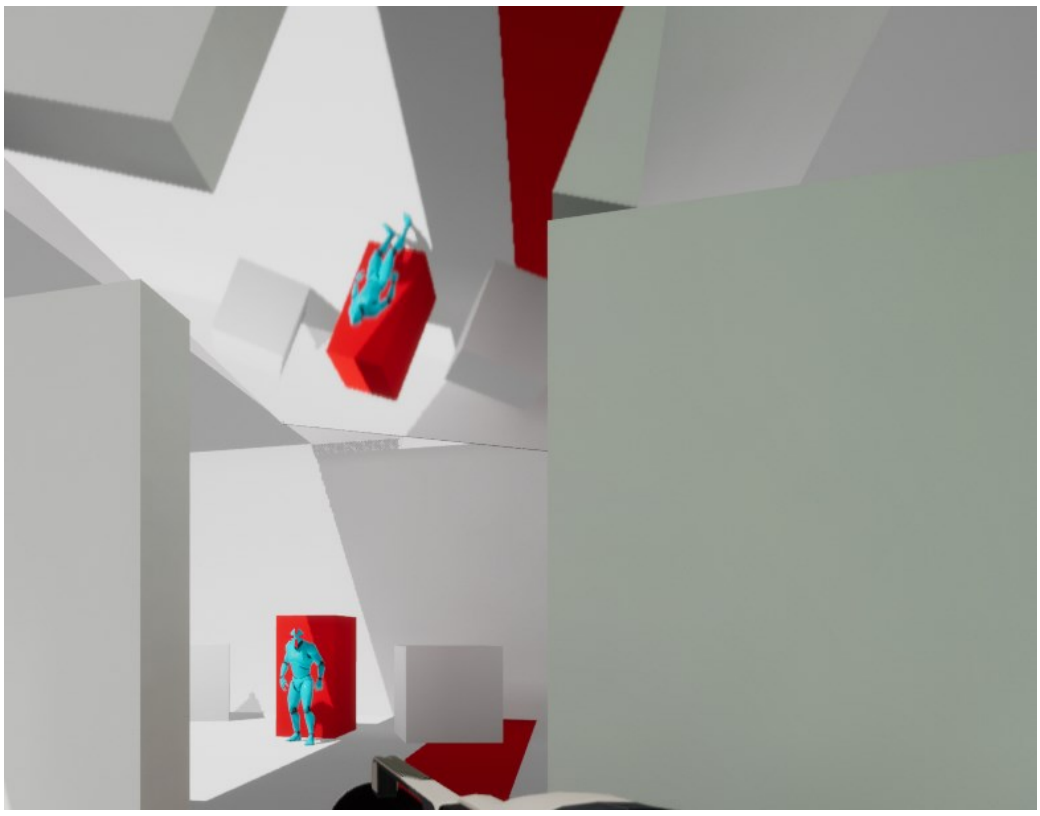
Alun perin DirectX 12 oli saatavana vain Windows 10:lle ja uudemmille käyttöjärjestelmille, mutta huhtikuussa 2018 Microsoft julkaisi aikeensa jakaa DirectX 12 -tukea vain tietyille peleille Windows 7 -käyttöjärjestelmän kanssa. Edellinen DirectX 11 oli julkaistu Windows 7 -käyttöjärjestelmän yhteydessä vuonna 2009, ja alun perin DirectX 12 julkaistiin vuonna 2015. DirectX Ultimate julkaistiin vuonna 2020 eli vuosi ennen Windows 11:n julkaisua, ja se on ollut saatavilla sekä Windows 10- että Windows 11 -käyttöjärjestelmille.

Vulkanin kehittäjä Khronos Group julkaisi säteenseurannan tuen laitteistolla tai ohjelmistoemuloinnin kautta vuoden 2020 GDC-konferenssissa. Siruvalmistajien ajurit käsittävät samat laitteet kuin DXR-tuelle, paitsi että tuessa on mukana myös vanhempia kortteja (Gamefromscratch 2020). Useilla pelimoottoreilla, esimerkiksi Unreal Enginellä, Sourcella, Unityllä ja CryEnginellä, on tehty jonkinasteisia Vulkan-pelejä tai versioita eri alustoille (PC Gaming Wiki 2021).

4 PELIDEMON KEHITTÄMINEN

Toteutettavalle pelidemolle tehtiin pelisuunnitteludokumentti (liite 1). Käytännön osuudessa toteutetaan Unreal Engineä ja sille saatavia lisäosia hyödyntäen lyhyt peliprojekti, joka toimii demona ja mahdollistaa eri laitteistokokoonpanojen, efektien sekä efektien optimoinnin vertailun. Käytännön osuudessa käytettävä Unreal Engine 5 käyttää DXR-rajapintaa säteenseurannan kiihdyttämiseen.

Demon yksinkertaisempi peili voitaisiin käytännössä tehdä planar reflection-tekniikalla, mutta silloin heijastuksen laatu kärsii. Lisäksi muut kuin tasapintaiset muodot eivät toimi tällä tekniikalla. Aiempaa planar reflection-tekniikkaa ei siis oteta huomioon tässä pelidemossa. Esimerkkinä tästä on kuvio 4.



KUVIO 4. Katossa ollut säteenseurantaa hyödyntänyt peili on korvattu planar reflection -heijastuksella.

4.1 Laitteistovaatimukset

Säteenseuranta vaatii minimissään näytönohjaimelta DirectX version 12:n tukea sekä joko laitteistotason säteenseurannan kiihdytyksen tai laiteajurin mahdollistamana ohjelmoidun DirectX Raytracing rajapinnan emuloinnin. Täyden tuen saavat näytönohjainkortit, joilla on DirectX 12 Ultimate -tuki. Raportin kirjoitushetkellä rajapintaa voivat hyödyntää laitteistokiihdytyksenä Nvidian näytönohjaimista Turing- ja Ampere-sarjojen kortit, ja AMD tukee säteenseurannan laitteistokiihdytystä RX 6000 -sarjan näytönohjainkorteissa (Hachman 2020). Käytännössä lähinnä vain Nvidian RTX-brändin alaiset näytönohjainkortit sekä AMD:n RX 6000 -sarjan kortit kykenevät useimpien DXR-kiihdytettyjen efektien tuottamiseen kohtuullisella suorituskyvyllä (Advanced Micro Devices 2020a). Kirjoitushetkellä uusimmat pelikonsolit, kuten Sonyn Playstation 5 (Sony 2021) ja Microsoftin Xbox Series X ja S, tukevat myös säteenseurannan laitteistokiihdytystä (Microsoft 2021a) (Microsoft 2021b).

Muiden kuin säteenseurannan laitteistokiihdytystä hyödyntävien näytönohjaimien teho säteenseurannan efektien tuottamisessa on yleisesti katsottu hyvin heikoksi, ja useissa peleissä säteenseurannan efektit on lukittu muilla kuin laitteistokiihdytyksen mahdollistavilla näytönohjaimilla (Walton 2019). Nvidia on sallinut näytönohjaimen CUDA-rajapinnan avulla ohjelmistokiihdytyksenä DXR-efektien tuottamisen Nvidian Pascal -sarjan näytönohjaimilla (GTX 1060 6GB ja nopeammat) sekä Nvidian GTX 1660- ja nopeammilla näytönohjaimilla (Mujtaba 2019).

4.2 Alati muuttuva kehitysympäristö

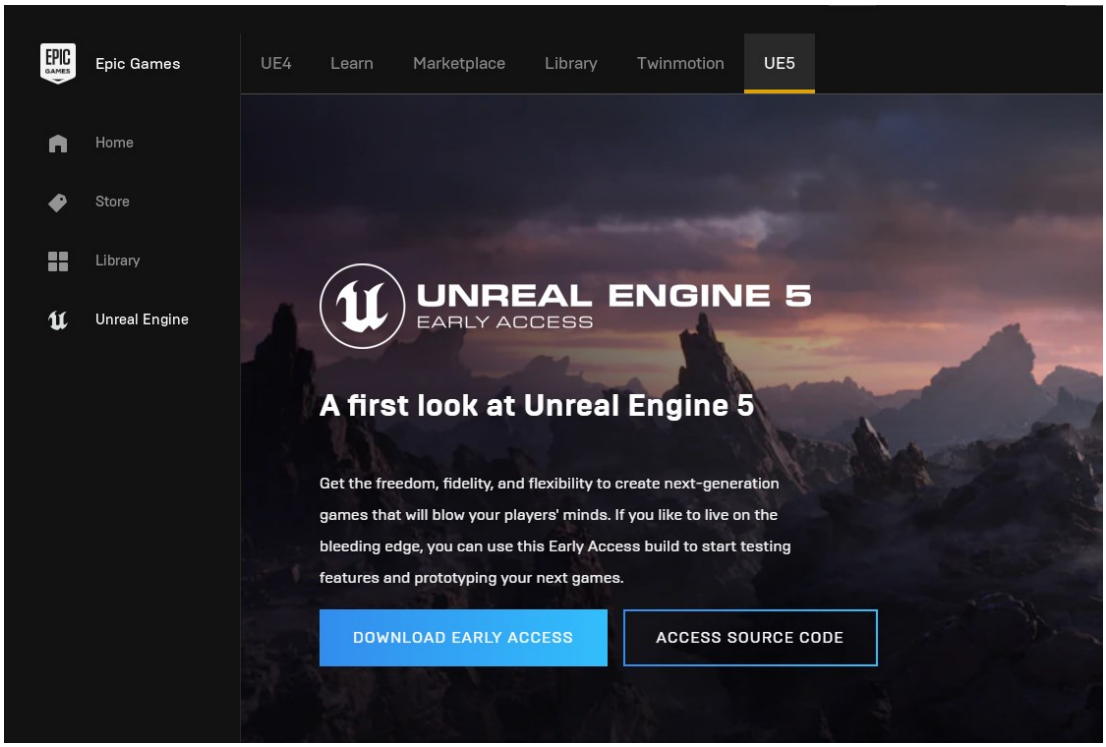
Valitettavasti kirjoitushetkellä uuden laitteiston saatavuus on erittäin huonoa, ja tämä pakottaa tekemään osan testeistä vain minimivaatimukset täyttävällä laitekokoonpanolla. Unreal Enginen versiossa 5 (Early Access) Nvidian Pascal-sarjan näytönohjaimien emuloitu säteenseurannan tuki on oletuksena pois käytöstä. Tämä voitiin kuitenkin sallia peliprojektin ”Config”-kansiosta löytyvästä ”DefaultEngine.ini”-tiedostosta.

Säteenseurantaan pohjautuvien global illumination -efektien laskenta ei juurikaan sovellu reaaliaikaisiin käyttötarkoituksiin niiden hitauden vuoksi. Unreal Engine 5:n kanssa on tarkoitus

käyttää siihen sisältyvää uutta Lumen-tekniikkaa valaistukseen ja heijastuksiin. Tässä demossa kuitenkin keskitytään vain säteenseurantaefekteihin. Myöhemmässä versiossa tai Unreal Engine 5:n täysjulkaisussa saatetaan tiputtaa säteenseurantaefektien tuki kokonaan uuden Lumen-tekniikan tieltä. Lumen on edelleen kehitteillä oleva tekniikka, joka ei vielä tue kaikkia ominaisuuksia, joten säteenseurannan tukea ei ole vielä otettu kokonaan pois käytöstä (Epic Games 2021b).

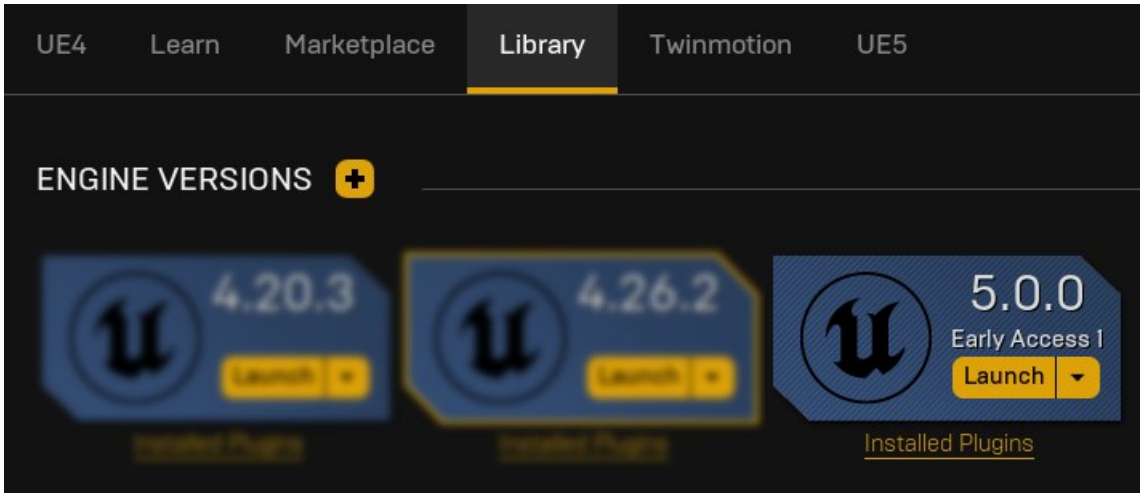
4.3 Unreal Enginen asentaminen

DirectX Raytracing vaatii vähintään Windows 10 -käyttöjärjestelmän version 1809. Tietokoneeseen on ladattu Epic Games Launcher, ja sen Unreal Engine -valikosta on ladattu vähintään Unreal Enginen versio 4.22. (Epic Games 2019.) Tässä tapauksessa on ladattu ja asennettu Unreal Enginen versio 5, joka on kirjoitushetkellä varhaisessa kokeilutilassa. Unreal Engine 5 ladataan painamalla käynnistimen yläpalkista ”UE5” ja klikkaamalla ”Download Early Access”-painiketta sivulla, kuten kuviosta 5 on nähtävissä.



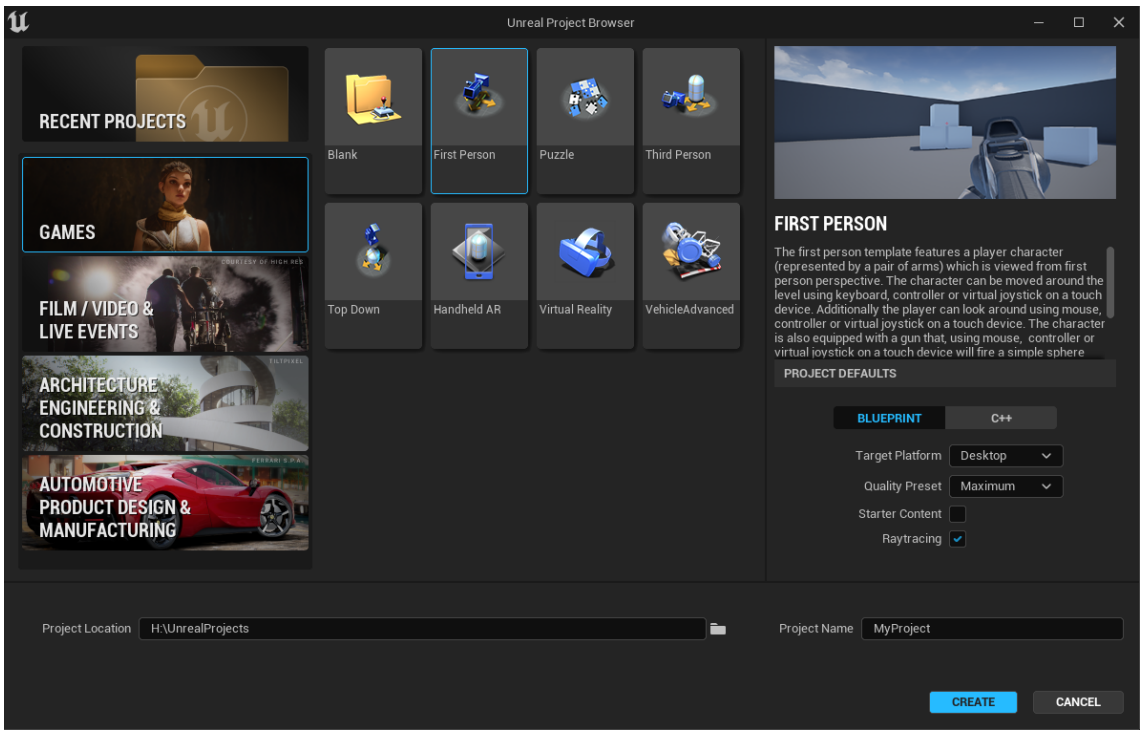
KUVIO 5. Kuvakaappaus Unreal Engine 5 -pelimoottorin asennussivusta.

Lataamisen ja asentamisen jälkeen Unreal Engine 5 näkyi käynnistimen Library-välilehdellä kuvion 6 osoittamalla tavalla.



KUVIO 6. Unreal Engine 5:n käynnistyspainike käynnistimessä.

Klikkaamalla ”Launch” sain näkyviin Unreal Project Browser -ikkunan (kuvio 7). Vasemmanpuoleisesta valikosta valitsin ”Games” ja aloitin First Person -templatea hyödyntävän projektin. On myös huomattavaa, että ”Project Defaults”-kohdasta valitsin ”Raytracing”-vaihtoehdon.



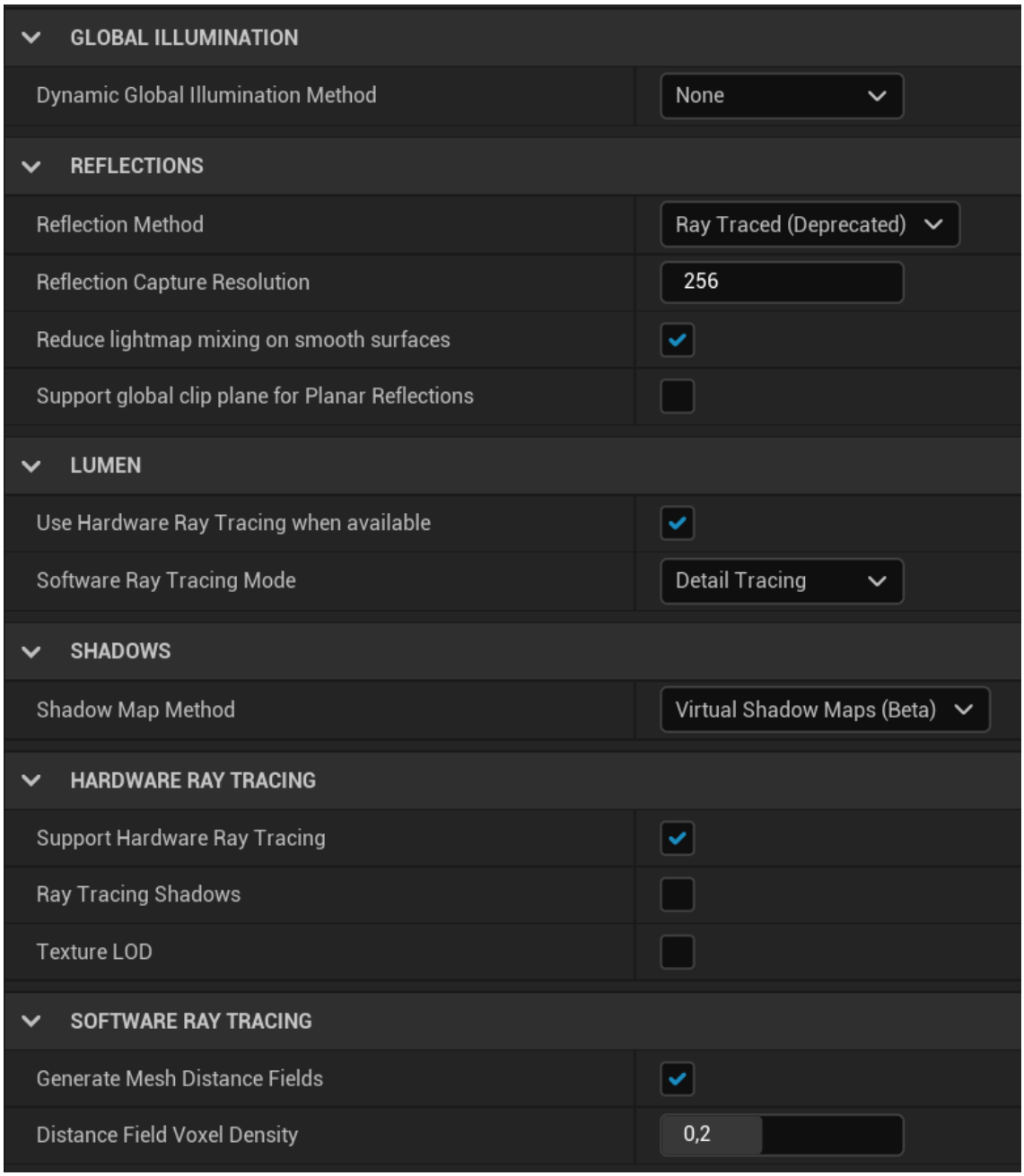
KUVIO 7. Unreal Project Browser. Korostettuna näkyvissä First Person -pohja.

4.4 Lyhyesti blueprinteistä

Unrealin blueprint on visuaalinen skriptausjärjestelmä, joka toimii koodin vastikkeena ja jota voidaan käyttää yhdessä koodin kanssa. Se mahdollistaa pelin skriptaamisen ilman koodin osaamista, ja sitä käytetään erityisesti visuaalisien elementtien skriptaamiseen sen helppokäyttöisyyden takia. Blueprint-järjestelmä on sulautettu pelimoottorin oliopohjaiseen ohjelmointirakenteeseen, mikä mahdollistaa koodin käyttämisen blueprinttien lisäosana. Koska järjestelmä on oliopohjainen, voidaan esimerkiksi koodissa tehdä luokka, johon voidaan pohjata blueprint, josta saadaan ohjattua koodipohjaisia toimintoja. Blueprint-skriptausta voidaan siis tukea koodin avulla ja toisin päin. Yleensä blueprinteillä tehdään pelin visuaaliset elementit, kuten menut ja pelaajan näytölle asetettavat tiedot, joita varten Unreal Enginestä löytyy sisäänrakennettu widget blueprint-editori (Epic Games 2021a).

4.5 Projektin asetukset

Projektista tehdään blueprint pohjainen sillä projekti ei vaadi C++-pohjan tarjoamia lisämahdollisuuksia. Projektin luomisen jälkeen optimoin post-processing-asetukset projektiin sopiviksi. Näitä asetuksia voidaan muokata useassa paikassa, kuten Project Settingsin Engine – Rendering -alavalikossa (kuvio 8). Tässä tapauksessa global illumination -kohdan alta asetettiin Dynamic Global Illumination Method pois päältä eli ensimmäiseen vaihtoehtoon valittiin ”None”.



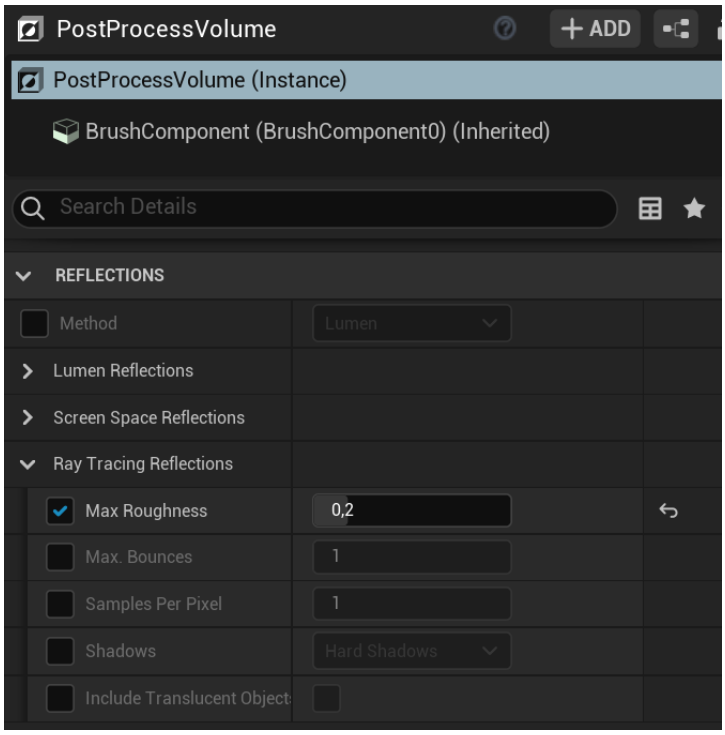
KUVIO 8. Unreal Enginen rendering-asetukset.

Reflections-kohdasta Reflection Method vaihdoin tässä tapauksessa vaihtoehtoon ”Ray Traced (Deprecated)”, jolloin heijastukset ovat säteenseurannan alla, kun taas global illumination -efektiä ei suoriteta.

Lumen-kohdasta asetetaan ”Use Hardware Ray Tracing when available” päälle sekä Hardware Ray Tracing kohdasta ”Support Hardware Ray Tracing” päälle.

4.6 Post Process Volume

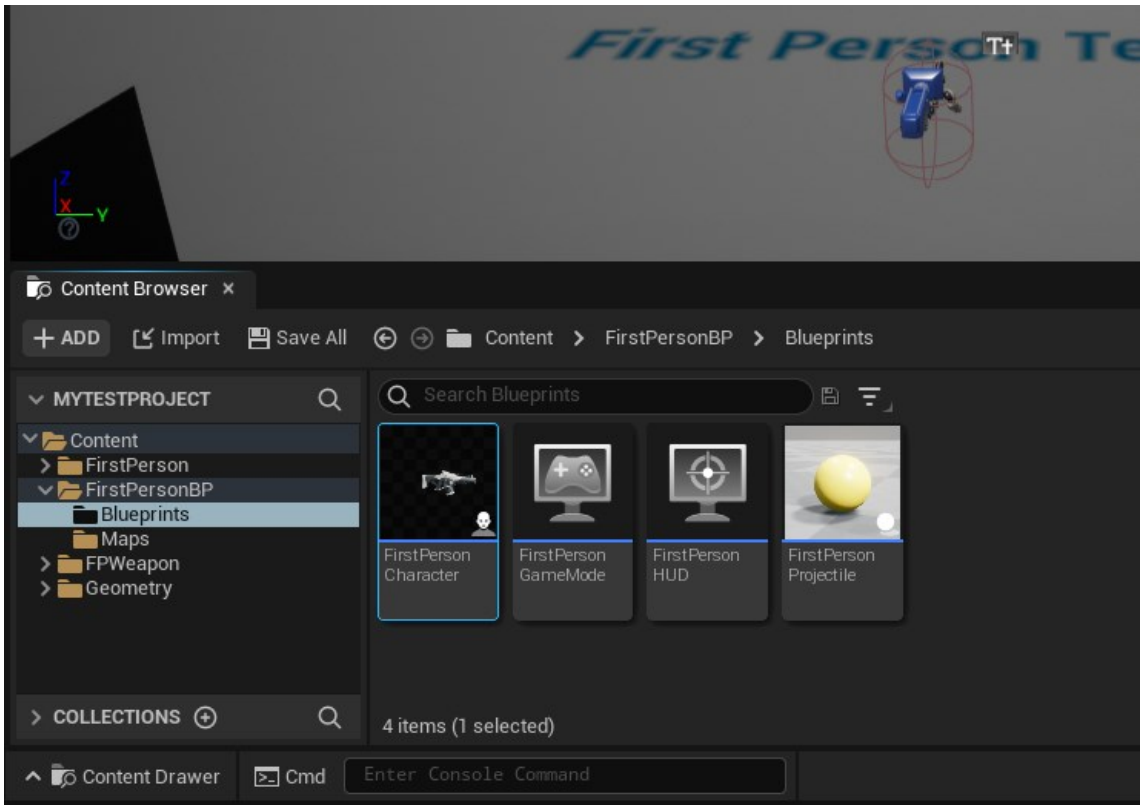
Käyttöliittymässä oikealla olevasta World Outlineristä etsitään PostProcessVolume, jolle asetetaan seuraavat asetukset: Reflections-kohdasta Max Roughness arvoksi annetaan 0,2, koska alle 0,2:n omaavat materiaalit käyttävät säteenseurantaa heijastuksiin. Näin tehdään kuvio 9:n osoittamalla tavalla. Muut optimoinnit tapahtuvat joko pelimoottorin konfiguraatiotiedostoissa tai yksittäisten materiaalien konfiguroinnissa.



KUVIO 9. Unreal Enginen PostProcessVolumen asetukset.

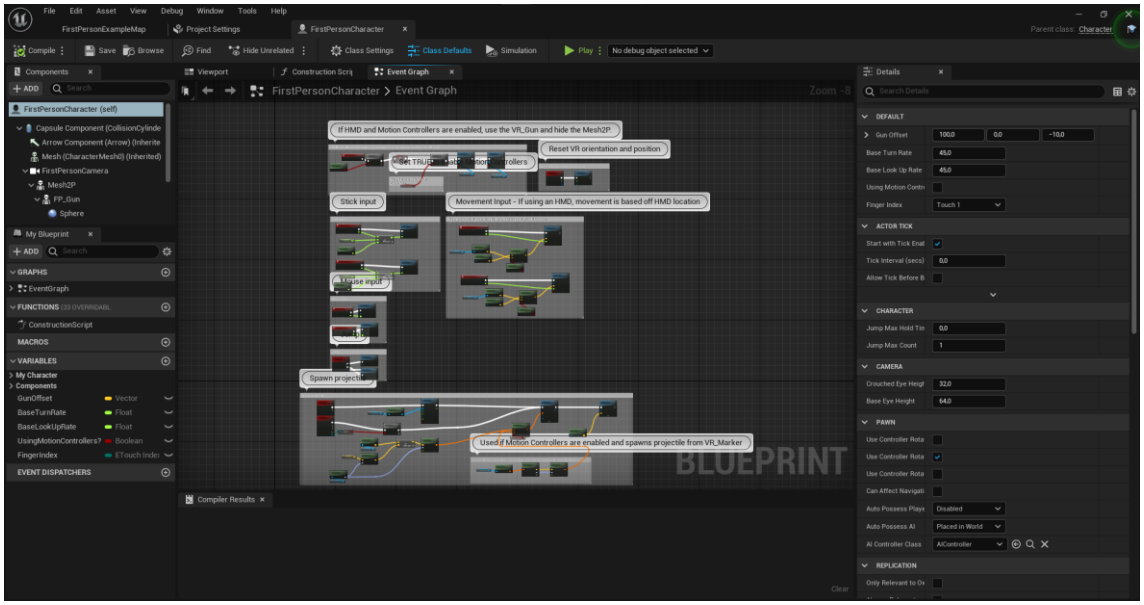
4.7 Pelaajahahmon kehittäminen

Pelihahmona toimii Unreal Enginen First Person Templaten mukana tuleva pelihahmon blueprint, jota muokkaan hieman yhdistämällä siihen Third Person Templaten mukana tulevan kolmannen persoonan pelihahmon, jotta hahmon heijastukset näkyisivät heijastuksissa. First Person Templaten mukana tuleva pelaajahahmon skeletal mesh -geometria näkyy vain sen omistavalle pelaajalle, eli esimerkiksi moninpelissä First Person Template -hahmosta näkyisi vain leijuva ase. First person template löytyy peliprojektin kansiosta ”FirstPersonBP/Blueprints” (kuvio 10).



KUVIO 10. First Person Character -blueprintin sijainti.

Ensin etsin Content Drawerista tai Browserista Content/FirstPersonBP/Blueprints/FirstPersonCharacter-blueprintin. Sitten tuplaklikkaan blueprintin auki (kuvio 11) ja muokkaan blueprintin käyttöön sopivammaksi yhdistelemällä siihen osia kolmannen persoonan hahmosta.



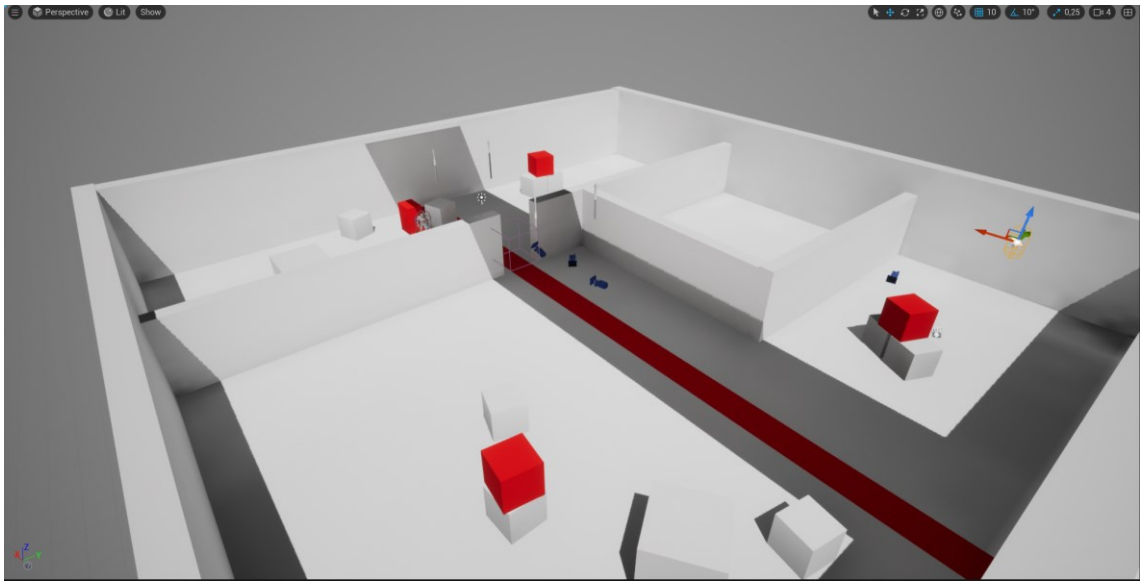
KUVIO 11. First Person Character Blueprintin sisältö.

Unrealin blueprint-hahmot toimivat yksinkertaisesti: pelihahmo on hahmoluokkaan perustuva blueprint tai C++- koodi, johon oletuksena kuuluvat collision-, arrow-, character mesh- ja character movement-komponentit. Näistä character movement sisältää pelihahmon liikehdintään liittyvät arvot. Character mesh sisältää pelihahmon muodon eli hahmon skeletal meshin, johon liittyvät myös animaatiot sekä materiaalit, arrow'n hahmon suuntaamista varten ja collisionin törmäysten tutkimiseen. Oletuksena on estää hahmon liikkuminen esineiden läpi.

Itse pelihahmon ohjelmointiin on monta tapaa. Näppäinpainallukset voidaan asettaa liikuttamaan hahmoa lähes missä tahansa Game Mode Base -luokkaan perustuvassa blueprintissä tai C++-koodissa. Tässä tapauksessa käytettiin First Person Character blueprintiä, sillä se tuli First Person Templaten mukana ja siinä on valmiina koodattuina kaikki tarvittavat liikkeet. Poistin hyppäämisen osuuden blueprintistä, sillä sitä ei tarvittu tässä demossa.

4.8 Peliympäristön kehittäminen

Peliympäristönä käytän yksinkertaista valkoista ympäristöä, joka muodostuu Unrealin First Person Templaten mukana tulleista laatikoista sekä omaan materiaaliin perustuvista materiaaleista. Teen ympäristöstä helpommin ymmärrettävän lisäämällä erivärisiä laatikoita. Laatikoiden väriä säädän Material Instancen parametreista. Kuvio 12 näyttää peliympäristön lintuperspektiivistä.



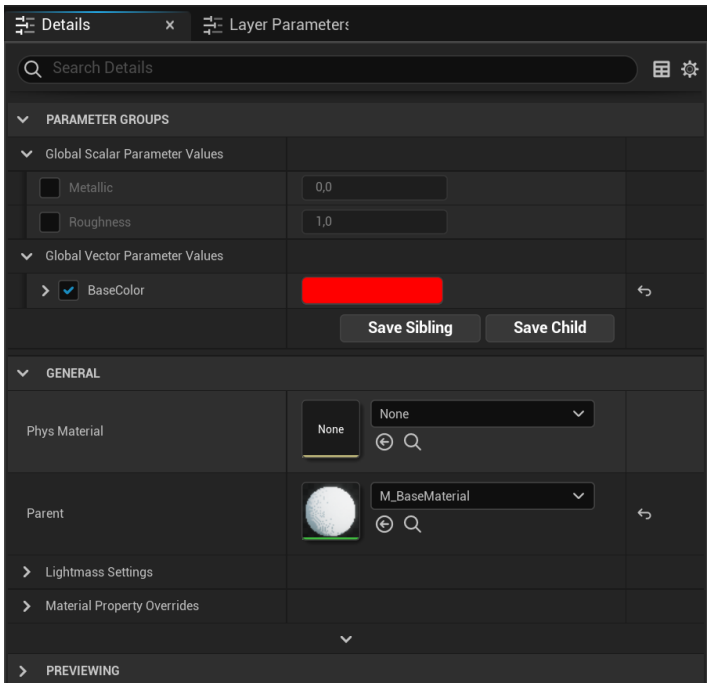
KUVIO 12. Pelikenttä.

4.9 Materiaalit

Unreal Engine käyttää sisäänrakennettua PBR-pohjaista shader-järjestelmää, jota voidaan helposti käyttää materiaalien avulla. Materiaaleissa määritetään muun muassa pinnan metallisuus, diffuusio, heijastuminen sekä läpinäkyvyys ja läpikuultavuus. Joillakin renderöintityypeillä (läpinäkymätön, läpikuultava, maskattu) on erilaisia ominaisuuksia. Kuviossa 13 esitetään projektin päämateriaali, joka sisältää värin, metallisuuden ja heijastumisen parametrit. Loin päämateriaaliksi ”M_BaseMaterialin”, jossa on Base Color-, Metallic- ja Roughness-parametrit. Materiaali käyttää muuten täysin oletusasetuksia. Päämateriaaliin luodaan säädettäviä parametrejä sekä asetetaan niille oletusarvot.

KUVIO 13. M_BaseMaterialin sisältö materiaalieditorissa.

Unreal Enginen materiaaleista voi tehdä niihin pohjautuvia materiaaleja (Material Instance), joille voidaan asettaa parametrien arvoja eri tavoin kuin päämateriaalissa ilman, että koko materiaali täytyy koota kokonaan uudelleen. Ne toimivat samaan tapaan kuin blueprintit, ja niillä on päämateriaali, johon ne pohjautuvat. Kuviossa 14 esitetään, miltä näyttää esimerkiksi projektin ”MI_Red Material Instance”.



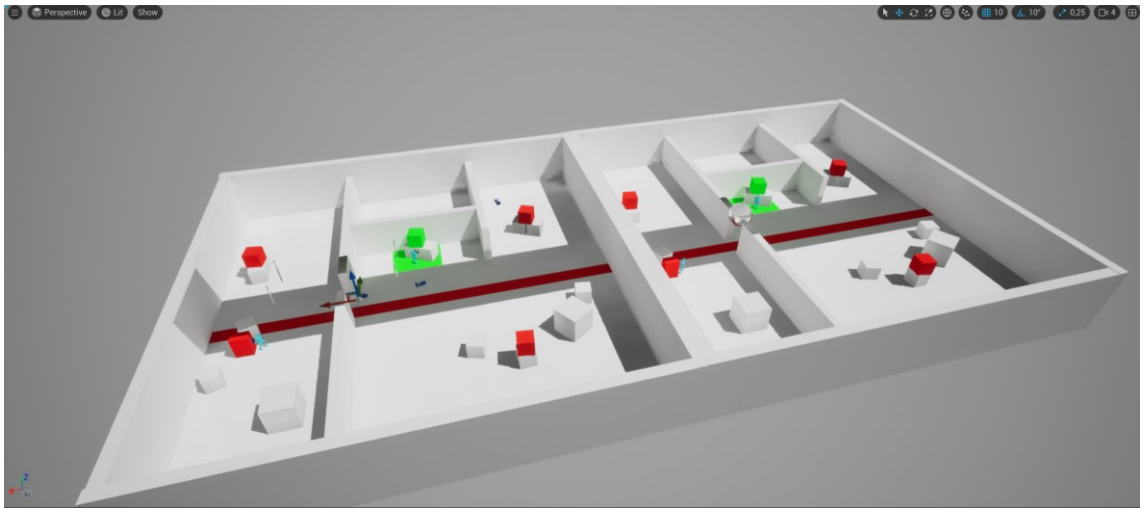
KUVIO 14. MI_Red material instacen sisältö.

Kuviossa 15 esitetään peliympäristöä pelaajan näkökulmasta. Pelaaja näkee peilin kautta vihollishahmon, joka ei näy suoraan pelaajan edessä. Peili voi olla minkä tahansa muotoinen tai suuntainen, ja se on reaaliaikainen. Peilit voivat myös heijastaa toisen heijastuksen kautta, mutta suorituskykysyistä tätä ei suositella. Kuviossa 15 esitetään 5 yhtäaikaista heijastusta. Pelaaja ei katso suoraan vihollispelaajaa, vaan näkee tämän vain heijastuksen kautta.

KUVIO 15. Viisi yhtäaikaista heijastusta eri suuntaan asetettujen peilien avulla.

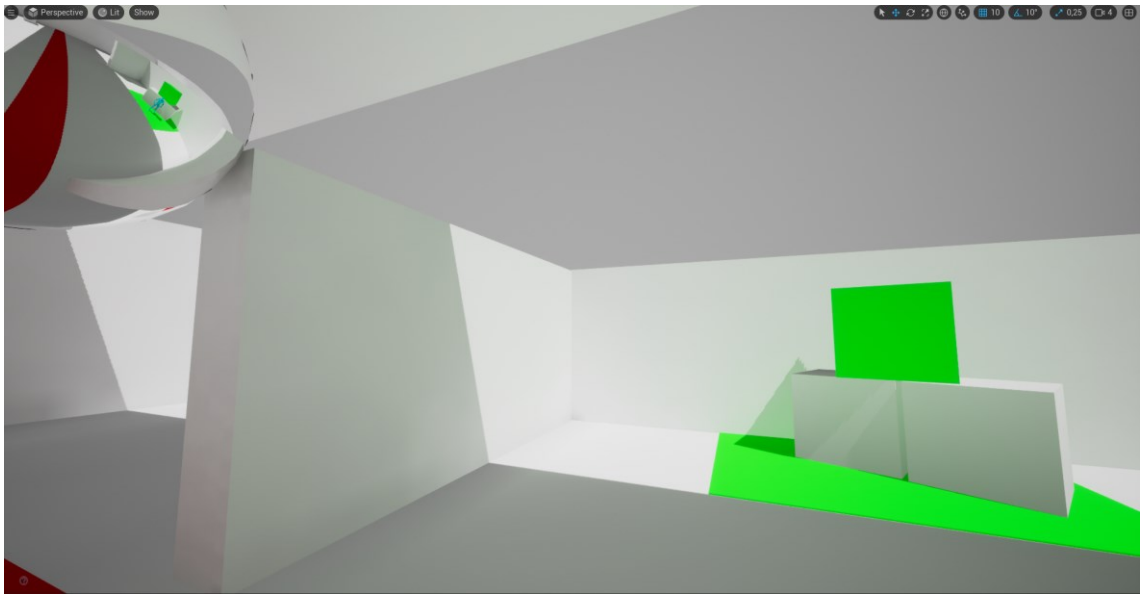
4.10 Pelilogiikan kehittäminen

Pelidemon logiikkana on yksinkertainen ensimmäisen persoonan ampumispelimekaniikka, ja pelisilmukkana toimii vielä yksinkertaisempi arcade-tyylinen pelisilmukka, joka aloittaa pelaajan jommassakummassa valmiiksi rakennetussa arcade-kentässä. Kuvio 16 esittää kumpaakin pelikenttää. Pelaajaa vastaan generoidaan vihollinen yhteen kahdesta valmiiksi asetetusta sijainnista.



KUVIO 16. Kummatkin pelikentät yhdessä.

Kentän tarkoituksena on hyödyntää säteenseurannan mahdollistamaa lähes virheetöntä heijastusta. Säteenseurannan rooli pelissä on toimia peilinä pelaajan ja vihollisten välillä. Toisessa kentistä pelaajan pitää ampua vihollinen katossa olevan peilin avulla ja toisessa katosta roikkuvan pallon avulla (kuvio 17). Peili voi olla teoriassa minkä tahansa muotoinen, mutta tässä pelidemossa käytetään vain yksinkertaisia muotoja.

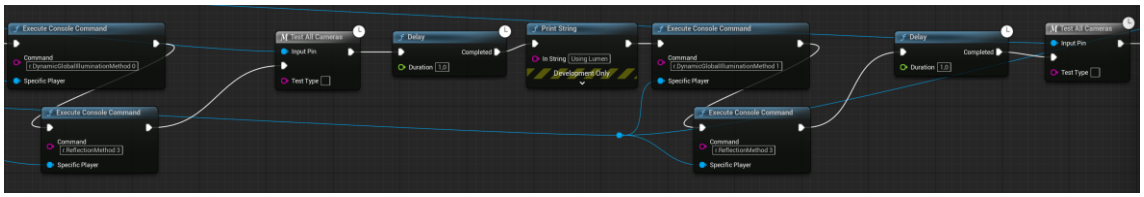


KUVIO 17. Pallon kautta näkyvä heijastus.

Demossa voi valita joko pelkän automaattisen testitilan tai interaktiivisen testitilan, jossa pelaaja voi itse pelata ja testata muun muassa pelaajan syöttämien liikkeiden viivettä. Pelaaja voi liikkua ympäriinsä pelikentässä, ampua laatikoita kumoon ja ampua vihollisen. Mikäli pelaaja ampuu vihollisen onnistuneesti, pelaaja palautetaan takaisin päävalikkoon.

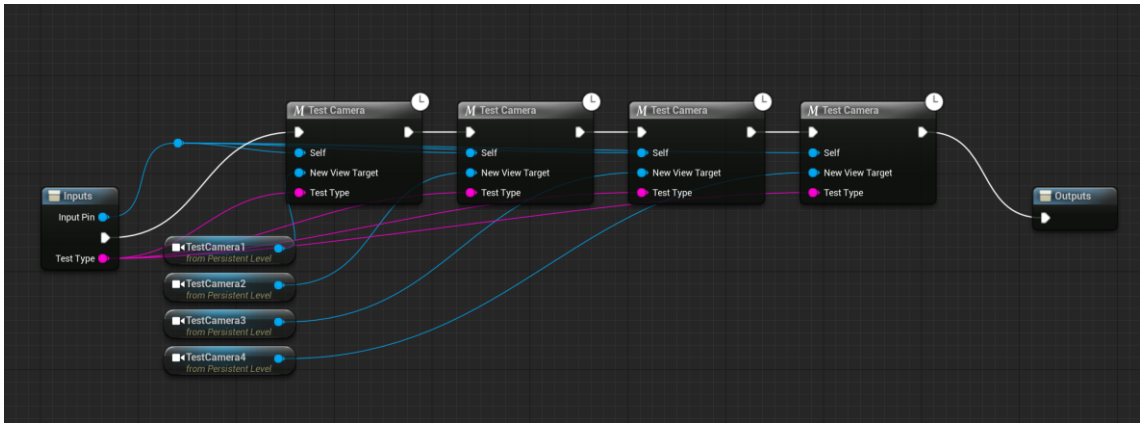
5 SUORITUSKYKYMITTAUS SEKÄ STATISTIIKAN KERÄYS JA KÄSITTELY

Yhtenä demon tarkoituksista on saada muodostettua dataa eri efektien järjestelmäkuormituksesta ja suorityskyvystä eri laitteistokokoonpanoilla. Statistiikan keräämiseen käytin blueprint-koodausjärjestelmää. Kuviossa 18 blueprint asettaa GI-tavan arvoon 0 (None), reflection-tavan arvoon 3 (Raytracing), käy läpi kaikki kamerat, odottaa sekunnin ajan ja tekee saman testin käyttäen uutta Lumen GI -järjestelmää (GI tapa 1) ja edelleen säteenseurantaa heijastuksiin. Tämä blueprint, jota kuvio 18 esittää, sisältää kahdesti makron ”Test All Cameras”.



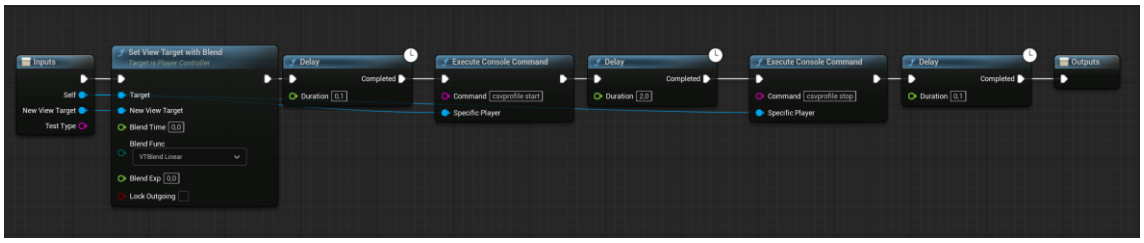
KUVIO 18. Asetuksia vaihtava blueprint.

Kuvio 19 esittää blueprintiä, joka käy läpi ja testaa kaikki valmiiksi kentälle asetetut kamerat kuviossa näkyvässä järjestyksessä. Tätä dataa ei lopulta käytetty kaavioihin. Sen sijaan käytettiin vain myöhempää kuviossa 21 esitettävää animaatioiden toistamisen tuottavaa dataa.



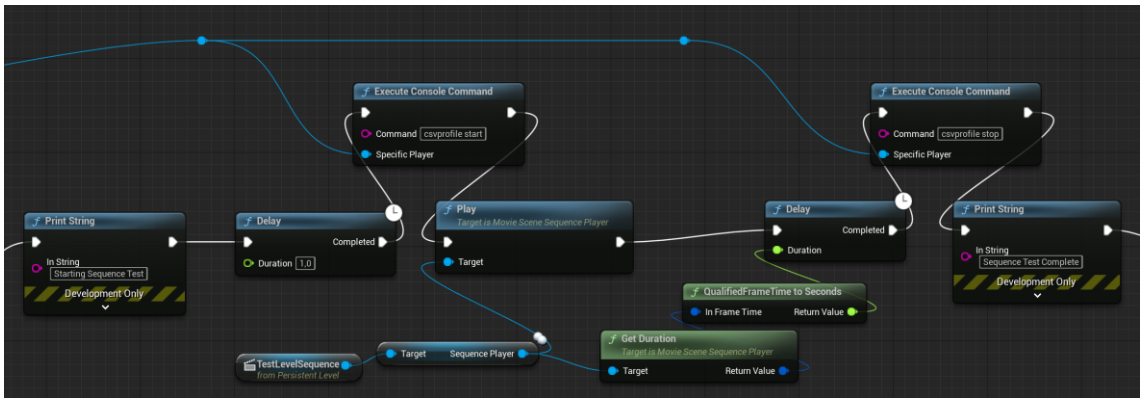
KUVIO 19. Kamerat läpi käyvä blueprint.

Makron ”Test Camera” sisältönä on kuvion 20 mukainen blueprint, joka asettaa pelin kameran, odottaa 0,1 sekuntia, aloittaa statistiikan keräämisen, odottaa 2 sekuntia ja lopettaa keräämisen, kunnes se odottaa 0,1 sekuntia ennen kuin siirtyy seuraavaan kameraan.



KUVIO 20. Kameroita vaihtava blueprint.

Tämän testausvaiheen jälkeen suoritetaan Unrealin sequence-editoria hyödyntävä testi, jossa kerätään samalla tavalla tietoa CSV tiedostoon. Tällä kertaa kuitenkin toistetaan valmiiksi animoitu kohtaus, kuten kuviosta 21 nähdään.



KUVIO 21. Tietoja tallentava blueprint, joka ohjaa myös animaatioiden toistoa.

Testi lopetetaan viiveen jälkeen, jonka kesto lasketaan TestLevelSequence-kohtauksen keston perusteella.

Pelimoottori tallentaa ”csvprofile stop”-komennon suorittamalla csv- tiedostot projektin Saved/Profiling/CSV-kansioon, josta data voidaan kopioida seuraavaan vaiheeseen, jossa tiedostot käsitellään ja jossa niistä muodostetaan kaavioita. Pyysin kaikkia testin tehneitä lähettämään kyseiset tiedostot ja muodostin niistä statistiikat ja useita kaaviota.

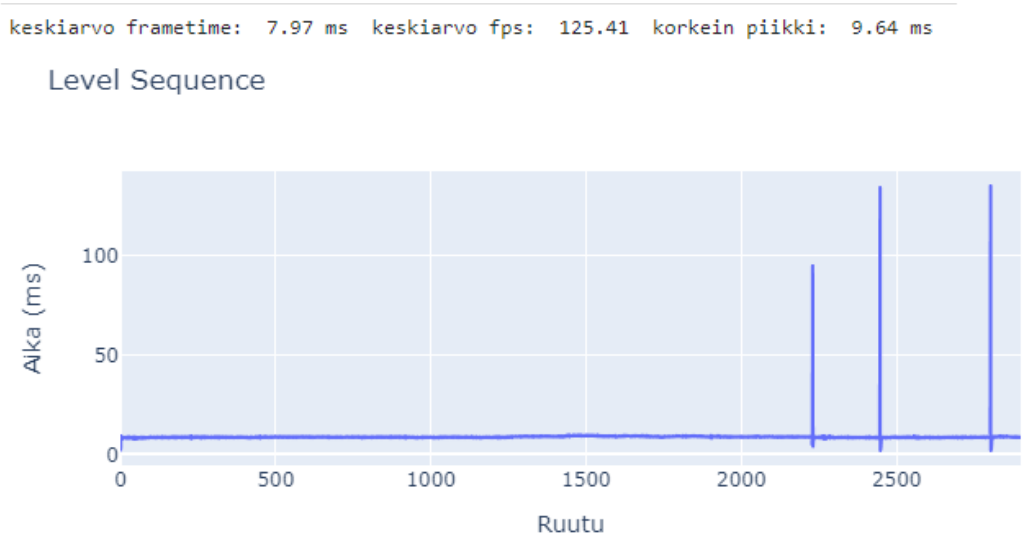
5.1 Statistiikka

Kuvio 22 esittää erillistä python-skriptiä, jonka avulla kerätty data käsiteltiin kaavioihin (liite 2). Samalla datasta laskettiin ruutunopeuden keskiarvo sekä ruutujen piirtämiseen käytetyn ajan keskiarvo.

```
import glob, os
import pandas as pd
import plotly.io as pio
import plotly.express as px

luettavat = ['GTX 1060 6GB/*.csv', 'GTX 1070/*.csv', 'RTX 2070/*.csv', 'RTX 2070 Super/*.csv', 'RTX 3060 Ti 2/*.csv']
filut = []
for filu in luettavat:
    filut.append(glob.glob(filu)[8])

nimi = 'Level Sequence'
nimet = ['GTX 1060 6GB', 'GTX 1070', 'RTX 2070', 'RTX 2070 Super', 'RTX 3060 Ti']

pio.renderers.default = "jupyterlab"

def plot(file, index):
    df = pd.read_csv(file, skipfooter=2, engine='python')
    df.columns = df.columns.str.replace('/', ' ')
    df = df.filter(['FrameTime'])

    #lasketaan keskiarvo frametime
    avgft = df['FrameTime'].mean()

    #lasketaan keskiarvo FPS
    avgfps = 1/(avgft/1000)

    print('keskiarvo frametime: ', round(avgft, 2), "ms  keskiarvo fps: ", round(avgfps, 2), " korkein piikki: ", round(df['FrameTime'].max(), 2), "ms")

    df.clip(0, 60)
    fig = px.line(df, x = df.index, y = 'FrameTime', labels={"index": "Ruutu", "FrameTime": "Aika (ms)"}, title=nimet[index])
    fig.show()

for index, filu in enumerate(filut):
    plot(filu, index)
```

KUVIO 22. Python-skripti, joka käsittelee kerätyn statistiikan.

Kerätty data luetaan luodusta CSV-tiedostosta. Tässä tapauksessa käytetään vain FrameTime-kolumnin dataa, josta muodostetaan keskiarvo sekä lasketaan ruutunopeuden keskiarvo. Tämän tiedon perusteella muodostetaan kaavio. Oheisessa kuviossa 23 esitetään esimerkkinä RTX 3060 Ti -näytönohjainkortin kaavio testistä. Liitteessä 2 on muiden testattujen näytönohjainkorttien statistiikasta luodut kaaviot.



KUVIO 23. Kerätystä statistiikasta muodostettu kaavio

5.2 Yhteenveto

Minimikokoonpanolla (GTX 1060 6GB) keskimääräinen ruutunopeus (FPS) alittaa 60, ja kaaviosta (liite 2) on nähtävissä, että ruutujen piirtäminen ei ole yhtä tasaista kuin muilla korteilla. Teknisesti tämä on minimi säteenseurannan piirtämiseen, mutta minkään monimutkaisemman kohtauksen piirtämisessä kortilla ei ole järkevää alkaa pelaamaan muita kuin tarinapohjaisia pelejä, joissa ei tarvita nopeaa ruudunpäivitysnopeutta. GTX 1070 suoriutui paremmin kuin GTX 1060, vaikka sekin pystyi vain 49 ruutuun sekunnissa. Tämäkään ei täytä yleisesti haluttua 60 ruudun sekuntinopeutta. Säteenseurannan laitteistokiihdytyksen sisältävä RTX 2070 suoriutui tehtävästä paremmin, ja siinä on havaittavissa selkeästi tasaisempi kaavio. RTX 2070:n ja RTX 2070 Superin keskiarvoruutunopeus on samankaltainen, mutta RTX 2070 Super suoriutui parhaiten. RTX 3060 Ti:n tulos oli selkeästi parempi ja jopa parempi kuin RTX 2070 Super -kortin, vaikka RTX 2070 Super käyttääkin melkein samaa sirua kuin RTX 2080. (Techpowerup 2021)

Kirjoitushetkellä näytönohjaimien hinnat ovat sirupuutteiden ja muiden syiden vuoksi todella kalliit, joten minkäänlaista hintavertailua on turha toteuttaa. Kirjoitushetkellä uusimmassa pelilevitysjätti Steamin laitteistokokoonpanokyselyssä suosituimmasta 20 kortista noin 12 ylipäätään kykenee säteenseurantaan, mukaan lukien ohjelmistokiihdytykseen, ja näistä tämän testin sekä teholuokitusten perusteella arvioiden viisi kykenee yksinkertaisen kohtauksen suorittamiseen pelattavalla ruutunopeudella (Valve Software 2021).

6 POHDINTA

Opinnäytetyön tavoitteena oli pohtia ja testata, kuinka helppoa säteenseurannan hyödyntäminen on Unreal Engine -peliprojektissa, sekä valjastaa se pelidemon avulla pelimekaniikaksi. Toteutin pelidemon ja jaoin sen ohjeineen muutamalle testaamiseen suostuneelle henkilölle, jotka testasivat laitteillaan pelidemon ja lähettivät tulokset minulle. Pelidemossa oli mahdollisuutena joko pelata demoa tai testata. Osalla testaajista oli ongelmana vanhat laiteajurit, jotka piti päivittää, ennen kuin demon pystyi käynnistämään.

Lopullisesta pelidemosta tuli enemmänkin statistiikan keräämiseen käytetty työkalu, jonka avulla näytönohjainkortteja vertailtiin. Pelin pelattavuus oli heikohkoa, sillä pelissä pystyi vain liikkumaan eri suuntiin ja ampumaan läpinäkyviä palloja. Lisäksi peilien toteutus testatussa demossa oli rajoittunutta. Peilejä olisi voinut olla enemmän, ja niillä olisi voinut tehdä vaikkapa optisia illuusioita. Muitakin peilin muotoja olisi voinut kokeilla sekä liikkuvaa maalitaulua. Pelidemon toteuttamiseen ei juurikaan kulunut aikaa, mutta itse raportin kirjoittamiseen kului huomattavan paljon enemmän aikaa, sillä se oli haasteellisempaa minulle.

Jos pitäisi aloittaa uudelleen tästä pisteestä, aloittaisin paremmalla pelimekaniikan suunnittelulla, johon pohjautuen projektiin saisi perustellusti lisättyä säteenseurannan tuen. Lisäksi statistiikan keräämisessä ei olisi tarvinnut kerätä juurikaan muuta, kuin ruutujen piirtoaikoja, ja statistiikan keräysvaiheessa tehdyt kuvakaappaukset olisivat voineet jäädä välistä, sillä ne aiheuttivat suuria piikkejä datassa, ja se ei ollut statistiikan keräämisen kohde.

LÄHTEET

Advanced Micro Devices 2020a. DirectX 12 Technologies. Hakupäivä 7.12.2021. https://www.amd.com/en/technologies/directx12.

Advanced Micro Devices 2020b. YouTube-video 24.11.2021. Hakupäivä 7.12.2021. https://www.youtube.com/watch?v=cC-DDAq3PCM.

Epic Games & Wilson, Jeff 2019. Unreal Engine 4.22 Released. Hakupäivä 21.04.2021. https://www.unrealengine.com/en-US/blog/unreal-engine-4-22-released.

Epic Games 2020. Unreal Engine 4.25 released! Hakupäivä 7.12.2021. https://www.unrealengine.com/en-US/blog/unreal-engine-4-25-released.

Epic Games 2021a. Blueprint Visual Scripting. Hakupäivä 7.12.2021. https://docs.unrealengine.com/4.27/en-US/ProgrammingAndScripting/Blueprints/.

Epic Games 2021b. Lumen Global Illumination and Reflections | Unreal Engine Documentation. Hakupäivä 7.12.2021. https://docs.unrealengine.com/5.0/en-US/RenderingFeatures/Lumen/.

Gamefromscratch 2020. YouTube-video 23.11.2020. Hakupäivä 1.12.2021. https://www.youtube.com/watch?v=4O65_Fvfrv8.

GamersNexus 2020. YouTube-video 16.9.2020. Hakupäivä 17.11.2021. https://youtu.be/AmNL2Cg2OO8?t=1160.

Gapo, Branko 2021. What Is A Good FPS For Gaming? GPU Mag 25.10.2021. Hakupäivä 27.11.2021. https://www.gpumag.com/good-fps-for-gaming/.

Hachman, Mark 2019. Microsoft ports DirectX 12 to Windows 7, giving some older PC games a performance boost. PC World 13.3.2019. Hakupäivä 7.12.2021. https://www.pcworld.com/article/403429/microsoft-ports-directx-12-to-windows-7.html.

Hackman, Mark 2020. Your PC is either DirectX 12 Ultimate-ready, or you're not a real gamer. PC Gamer 30.10.2020. Hakupäivä 7.12.2021. https://www.pcworld.com/article/393630/your-pc-is-either-directx-12-ultimate-ready-or-youre-not-a-real-gamer.html.

Hofmann, G.R. Who invented ray tracing? The Visual Computer 6, 120–124 (1990). https://doi.org/10.1007/BF01911003.

Keller, Alexander, Fascione, Luca, Fajardo, Marcos, Georgiev, Iliyan, Christensen, Per, Hanika, Johannes, Eisenacher, Christian & Nichols, Gregory 2015. The path tracing revolution in the movie industry. ACM SIGGRAPH. https://doi.org/10.1145/2776880.2792699.

Lafortune, Eric P. & Willems, Yves D. 1993. Bi-directional path tracing. Leuven: Department of Computing Science Katholieke Universiteit Leuven.

Laine, Petrus 2020. Epic Games julkisti Unreal Engine 5 -pelimoottorin. Io-Tech 13.05.2020. Hakupäivä 14.10.2021. https://www.io-tech.fi/uutinen/epic-games-julkaisi-unreal-engine-5-pelimoottorin/.

Liu, Edward 2018. Low sample count Ray Tracing with Nvidia's Ray Tracing denoisers. SIGGRAPH 2018. Hakupäivä 12.08.2021. https://www.nvidia.com/en-us/on-demand/session/siggraph2018-sig1847/.

Luebke, David & Parker, Steven 2008. Interactive ray tracing with CUDA. SIGGRAPH 2008. Hakupäivä 7.12.2021. https://www.nvidia.com/content/nvision2008/tech_presentations/Game_Developer_Track/NVISION08-Interactive_Ray_Tracing.pdf

Microsoft 2021a. Xbox Series S | Xbox. Hakupäivä 30.11.2021 https://www.xbox.com/fi-FI/consoles/xbox-series-s.

Microsoft 2021b. Xbox Series X | Xbox. Hakupäivä 30.11.2021 https://www.xbox.com/fi-FI/consoles/xbox-series-x.

Mujtaba, Hassan 2019. NVIDIA Enables DXR Real-Time Ray Tracing Support on GeForce 10 and GeForce 16 Series, GeForce GTX 1060 and Above – Drivers Available in April. Wccftech 18.3.2019. Hakupäivä 7.12.2021. https://wccftech.com/nvidia-enables-dxr-raytracing-support-non-rtx-geforce-pascal-turing-gpus/.

Nutt, Christian 2014. Epic's Tim Sweeney lays out the case for Unreal Engine 4. Gamasutra 21.03.2014. Hakupäivä 16.10.2021. https://www.gamasutra.com/view/news/213647/Epics_Tim_Sweeney_lays_out_the_case_for_Unreal_Engine_4.php.

PC Gaming Wiki 2021. List of Vulkan games. PC Gaming Wiki 2021. Hakupäivä 7.12.2021. https://www.pcgamingwiki.com/wiki/List_of_Vulkan_games.

Remedy Entertainment 2018. Experiments with DirectX Raytracing in Remedy's Northlight Engine. Remedy 19.3.2018. Hakupäivä 12.11.2021. https://www.remedygames.com/experiments-with-directx-raytracing-in-remedys-northlight-engine/.

Rose, Daniel 2020. The year of Real Time Ray Tracing. Medium 24.01.2020. Hakupäivä 26.11.2021. https://medium.com/gametextures/2020-the-year-of-real-time-ray-tracing-f8b29e89523b.

Sony 2021. PlayStation®5 | Play Has No Limits | PlayStation. Hakupäivä 7.12.2021. https://www.playstation.com/fi-fi/ps5.

Techpowerup 2021. NVIDIA TU104 GPU Specs | TechPowerUp GPU Database. Hakupäivä 9.12.2021. https://www.techpowerup.com/gpu-specs/nvidia-tu104.g854.

Tyler Wilde 2020. Unreal Engine games no longer owe royalties on their first $1M in revenue. PC Gamer 13.5.2020. Hakupäivä 14.11.2021. https://www.pcgamer.com/unreal-engine-games-no-longer-owe-royalties-on-their-first-dollar1m-in-revenue/.

Valve Software 2021. Steam Hardware & Software Survey. Hakupäivä 1.12.2021. https://store.steampowered.com/hwsurvey/Steam-Hardware-Software-Survey-Welcome-to-Steam.

Walton, Jared 2019. You can now enable ray tracing on GTX cards, but performance is low. PC Gamer 11.4.2019. Hakupäivä 14.11.2021. https://www.pcgamer.com/you-can-now-enable-ray-tracing-on-gtx-cards-but-performance-is-low/.

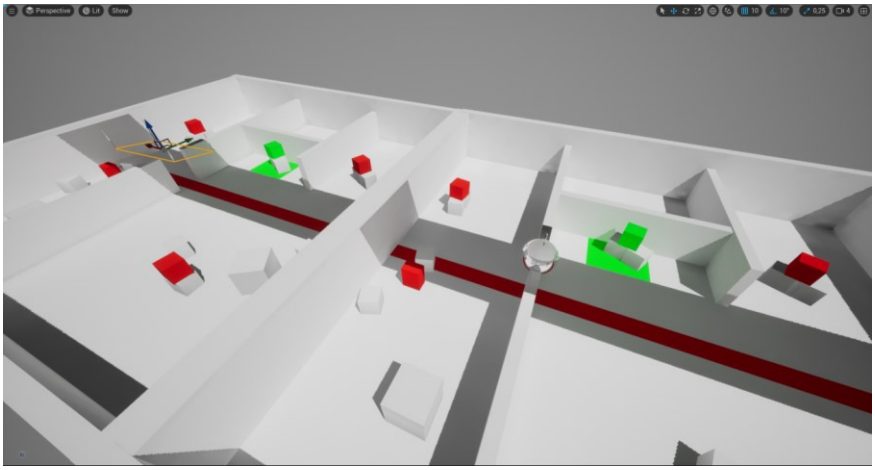
Game Design Document

DXR Testiprojekti - Petri Launimaa

Johdanto

Pelin tarkoituksena on toteuttaa yksinkertainen pelidemo, johon sisältyy pelimekaniikka, joka toimii vain säteenseurannan kanssa.

Tyyli ja grafiikka



Yksinkertainen tyyli, muuten väritön ympäristö, paitsi vihollisen ja sen ympäristön lähellä ovat kirkkaan väriset esineet. Yksinkertainen menu, josta voidaan valita joko pelitila tai testitila. Pohjana käytetään Unreal Enginen mukana tulevaa First Person Templatea, johon sisältyy pelaajan kädet sekä yksinkertainen ase. Vihollisten ulkonäkö on Unreal Enginen mukana tulevan Third Person Templaten mukana tulevan pelaajahahmon kaltainen paitsi, että vihollisten väriarvo on muutettu helpommin huomattavammaksi.

Pelimekaniikat

Pelimekaniikka perustuu First Person Templaten tarjoamaan First Person Shooter peliin. Pelaaja voi ampua näkymättömiä ammuksia suoraan eteenpäin. Twistinä tässä on, että vihollinen ei näy suoraan pelaajalle, vihollinen voidaan nähdä vain heijastusten kautta. Pelaajan pitää etsiä vihollinen heijastavan esineen avulla, joka roikkuu pelikentän keskellä katosta. Vihollinen voi ilmaantua jompaan kumpaan kahdesta eri paikasta tasossa. Viholliset ovat paikallaan eivätkä liiku niiden löytämisen helpottamiseksi.

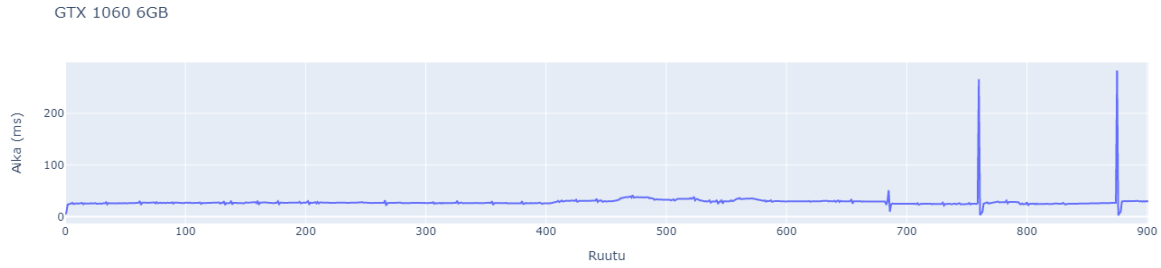
Rakenne ja laajuus

Pelissä on kaksi pelikenttää samassa tasossa vierekkäin, toisessa on katosta roikkuva heijastava peili, ja toisessa katosta roikkuva heijastava pallo. Pelikentässä on myös eri kokoisia laatikoita, joita pelaaja voi liikutella. Kun pelaaja on saanut tuhottua vihollisen, peli käynnistyy alusta 3 sekunnin laskennan jälkeen.

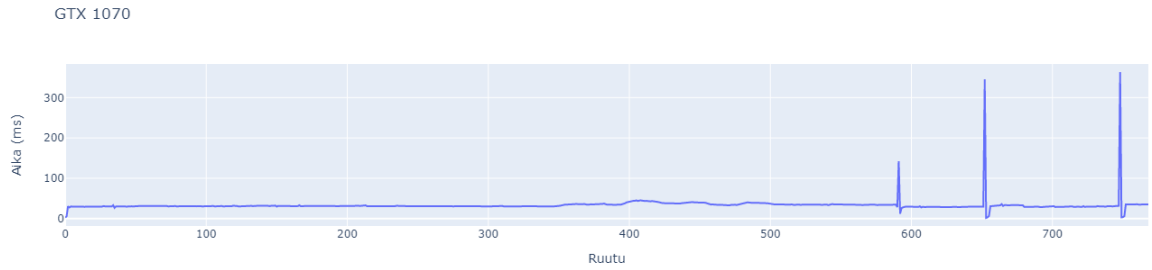
Ääni

Pelissä kuuluu muutamia erilaisia ääniä, jotka ovat peräisin Unreal Marketplacessa saatavilla olevasta paketista “Retro 8Bit Sounds”, joka oli osana Unreal Marketplacen sponsoroitua sisältöpakettia vuoden 2019 toukokuussa.

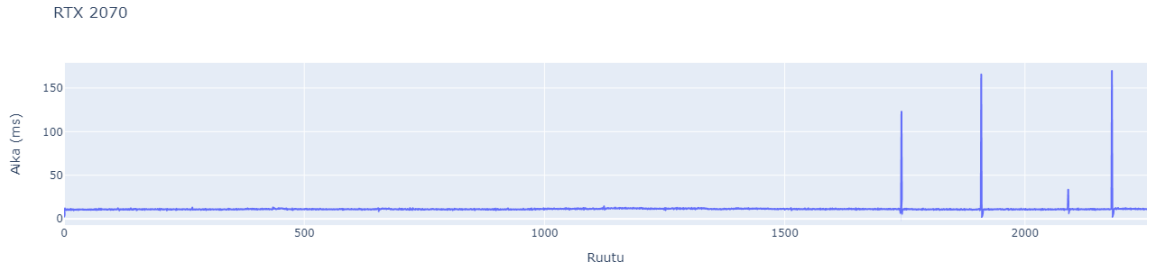
keskiarvo frametime: 28.31 ms keskiarvo fps: 35.32 korkein piikki: 283.07 ms



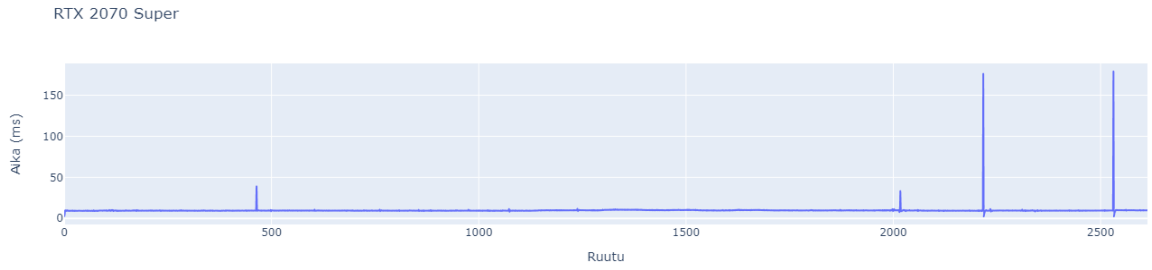
keskiarvo frametime: 33.17 ms keskiarvo fps: 30.14 korkein piikki: 363.18 ms



keskiarvo frametime: 11.33 ms keskiarvo fps: 88.24 korkein piikki: 169.49 ms



keskiarvo frametime: 9.78 ms keskiarvo fps: 102.26 korkein piikki: 178.85 ms



keskiarvo frametime: 8.82 ms keskiarvo fps: 113.44 korkein piikki: 134.12 ms