Valtteri Saikkonen

NANITE-TEKNIIKAN HYÖDYT 3D-MALLIEN KÄSITTELEMISESSÄ UNREAL ENGINE 5 -PELIMOOTTORILLA

Opinnäytetyö

Liiketalouden ammattikorkeakoulututkinto

Tietojenkäsittelyn koulutus

2023

Kaakkois-Suomen
ammattikorkeakoulu

Kaakkois-Suomen ammattikorkeakoulu

Tutkintonimike	Tradenomi (AMK)
Tekijä	Valtteri Saikkonen
Työn nimi	Nanite-tekniikan hyödyt 3D-mallien käsittelemisessä Unreal Engine 5 -pelimoottorilla
Toimeksiantajat	Xamk Game Studios -hanke ja Xamk ProRak -hanke
Vuosi	2023
Sivut	30 sivua
Työn ohjaaja	Jukka Selin

TIIVISTELMÄ

Tämän opinnäytetyön tavoitteena oli tutkia Unreal Engine 5 -pelimoottorissa olevaa Nanite-renderöintitekniikkaa ja sen tuomia hyötyjä 3D-mallien käsittelemisessä ja niiden pelillistämisessä. Opinnäytetyö on tehty Xamk Game Studios - ja Xamk ProRak -hankkeille, joiden tavoitteina on edistää peliohjelmointia, pelillistämistä ja rakennussuunnittelua, sekä löytää ratkaisuja suurten 3D-mallien optimointiin ja niiden käsittelyyn liittyviin ongelmiin.

Opinnäytetyössä tutustutaan pelillistämiseen, Unreal Engine 5 -pelimoottoriin ja sen ominaisuuksiin, joista tarkemmin perehdytään Nanite-tekniikkaan. Työssä myös käydään läpi ja opastetaan Unreal Engine 5 pelimoottorissa olevien profilointityökalujen käyttöä. Siinä kerrotaan, miten ja mitä tietoa niillä voidaan kerätä ja kuinka sitä tietoa voidaan analysoida ja hyödyntää projektien optimoinnissa. Näitä työkaluja hyödynnetään myös opinnäytetyössä Nanite-tekniikan tuomien hyötyjen selvittämiseen ja niiden toteamiseen.

Työtä varten tehtiin kaksi Unreal Engine 5 -projektia, joissa molemmissa oli kohteena Pieksämäen Hiekanpään yläkoulun 3D-malli. Yhteen projekteista oli otettu Nanite-renderöintitekniikka sekä verkkomalli käyttöön. Toiseen projektiin taas Unreal Engine 5 -pelimoottorin perinteinen renderöintitekniikka ja kolmioverkkomalli. Näitä projekteja analysoitiin ja vertailtiin Profiler ja Unreal Insights -profilointityökaluilla. Lopputuloksena saatiin todennettua, että Nanite-tekniikka parantaa selvästi raskaita 3D-malleja sisältävien ohjelmien suorituskykyä ja kykyä käsitellä suuria kohteita.

Asiasanat: Nanite, Unreal Engine 5, pelillistäminen, 3D-malli, profilointi

South-Eastern Finland University of Applied Sciences

Degree	Bachelor of Business Administration
Author	Valtteri Saikkonen
Thesis title	The benefits of Nanite technology for handling 3D models with the Unreal Engine 5 game engine
Commissioned by	Xamk Game Studios project and Xamk ProRak project
Time	2023
Pages	30 pages
Supervisor	Jukka Selin

ABSTRACT

The objective of this thesis was to research the Nanite rendering technology in the Unreal Engine 5 game engine and its benefits for handling 3D models and gamifying them. The thesis was made for the Xamk Game Studios and Xamk ProRak projects, with goals to advance game programming, gamification and to develop building design, as well as to find solutions to problems related to the optimization of large 3D models and handling them.

The thesis introduced gamification, the Unreal Engine 5 game engine and its features, of which the Nanite technology was introduced in more detail. The thesis also reviewed and guided in the use of the profiling tools in the Unreal Engine 5 game engine. The thesis explained how and what data could be collected with them and how that data could be analysed and utilized in the optimization of projects. These tools were also used in the thesis to find out and verify the benefits of Nanite technology.

Two Unreal Engine 5 projects were made for the thesis, both of which had a 3D model of the Hiekanpää middle school in Pieksämäki. Nanite rendering technology and mesh format were used in one of the projects. For the second project, the default rendering technology and triangle mesh format in the Unreal Engine 5 game engine were used. These projects were analysed and compared using Profiler and Unreal Insights profiling tools. As a result, it was verified that the Nanite technology clearly improves the performance of programs containing heavy 3D models and the ability to handle large objects.

Keywords: Nanite, Unreal Engine 5, gamification, 3D model, profiling

SISÄLLYS

1 JOHDANTO

Opinnäytetyön aiheena on Unreal Engine 5 -pelimoottorissa oleva Nanite-tekniikka ja sen tuomat edut 3D-mallien käsittelemisessä pelimoottorilla. Nanite on virtualisoitu geometriajärjestelmä Unreal Engine 5 -pelimoottorissa. Nanite käyttää uudenlaista verkkomuotoa ja renderöintitekniikkaa objektien renderöintiin. Unreal Engine 5 -pelimoottori julkaistiin huhtikuussa 2022, joten aihe on myös ajankohtainen.

Tämän opinnäytetyön toimeksiantajina toimivat Xamk Game Studios -hanke sekä Xamk ProRak -hanke. Hankkeiden tavoitteena on edistää peliteknologioiden osaamista ja rakennussuunnitelmien havainnollistamista pelimoottoreiden avulla. Opinnäytetyö antaa näille hankkeille ratkaisuja pelimoottoreissa geometrialtaan laajojen ja monimutkaisten 3D-mallien käsittelemiseen ja käsittelyssä esiin nouseviin ongelmiin, sekä ohjeistaa hyödyntämään Nanite-tekniikan tuomia etuja.

Tässä opinnäytetyössä perehdytään pelillistämiseen, Unreal Engine 5 -pelimoottoriin ja tarkemmin siinä olevaan Nanite-tekniikkaan. Työssä käydään myös läpi Unreal Engine 5 -pelimoottorin profilointityökaluja ja niiden käyttötarkoituksia. Työssä myös opastetaan, kuinka näitä profilointityökaluja käytetään ja miten niillä kerättyä tietoa voidaan analysoida ja hyödyntää projektien optimoinnissa.

Nanite-tekniikan hyötyjen selvittämistä varten työssä tehtiin kaksi projektia Unreal Engine 5 -pelimoottorilla. Kummassakin projektissa käytettiin kohteena Pieksämäellä sijaitsevan Hiekanpään yläkoulun 3D-mallia. Toisessa projektissa tähän 3D-malliin laitettiin Nanite-tekniikka ja renderöintimalli päälle. Ja vertailuprojektissa käytettiin Unreal Engine 5 -pelimoottorin tavallista renderöintitekniikkaa ja kolmioverkkomallia. Näiden projektien eri osa-alueista tehtiin analyysejä Unreal Engine 5 -pelimoottorissa olevilla Profiler - ja Unreal Insights -profilointityökaluilla. Naniten tuomia hyötyjä selvitettiin vertaamalla näitä analyysejä keskenään.

2 PELIMOOTTORIT

Pelimoottorit ovat ohjelmistokehyksiä, jotka on pääasiallisesti suunniteltu videopelien tuottamista varten. Yleisesti pelimoottoreissa on valmiina kehitystyökaluja ja ohjelmistokomponentteja, jotka helpottavat pelien kehittämistä tekemällä monimutkaisista tehtävistä yksinkertaisen näköisiä (Game Engine and History of Game Development s.a.)

Unreal Engine 5 -pelimoottorissa nämä kehitystyökalut on viety hyvinkin pitkälle. Yksinkertaisten pelien kehittämisessä ei välttämättä tarvitse kirjoittaa riviäkään koodia, sillä pelin kaiken ohjelmoinnin ja toiminnallisuuden pystyisi tekemään visuaalisella Unreal Enginen Blueprint-työkalulla.

Pelimoottorin komponentit koostuvat yleisesti grafiikanrenderöijästä 2D- tai 3D -grafiikkaa varten. Lisäksi pelimoottorissa on fysiikkamoottori, jolla voidaan simuloida esimerkiksi painovoimaa, nopeutta, törmäyksen tunnistusta ja nesteen dynamiikkaa. Myös äänenhallinta, verkkoyhteys, tekoäly ja erilaiset ohjelmointityökalut sisältyvät pelimoottoreihin. (Game Engine and History of Game Development s.a.)

2.1 Unreal Engine 5

Unreal Engine 5 on Epic Games:n kehittämä pelimoottori, joka julkaistiin 5.4.2022. Se on tarkoitettu 3D-peliprojektien luontiin, mutta siinä on myös työkaluja 2D-projektien tekemistä varten. Unreal Engine 5 on ilmainen ladata ja käyttää, jopa sillä tehtyjä ohjelmia voidaan myydä ilman lisenssimaksuja miljoonan Yhdysvaltain dollarin bruttotuloihin asti (Download s.a., Unreal Engine).

Unreal Engine 5 toi pelimoottoriin uusia ominaisuuksia, joista merkittävimmät ovat Lumen ja Nanite. Lumen on globaali valaistus- ja heijastustekniikka, joka reagoi reaaliajassa valoihin tai muihin skenessä tehtyihin muutoksiin. Unreal Engine 5 -pelimoottoriin on tullut useita päivityksiä, joista viimeisin versio on Unreal Engine 5.3.

Näissä päivityksissä on tullut monia täysin uusia ja hyödyllisiä työkaluja. Yksi näistä työkaluista on Light Mixer, joka on valoeditointityökalu, jonka avulla voidaan muokata skenen kaikkia valokomponentteja yhtäaikaisesti samasta ikkunasta. Äänimaailman luontiin tuli myös useita ominaisuuksia, kuten MetaSounds, joka on korkean suorituskyvyn äänijärjestelmä pelimoottorissa tapahtuvaan äänenluontiin ja äänien ohjaamiseen. Lisäksi äänien käsittelyyn on tarjolla Soundscape, joka luo ympäristönääniä proseduaalisesti, sekä Waveform Editor -työkalu, jolla voidaan muokata ääniaaltoja pelimoottorilla sisäisesti. Pelimoottorin käsittelemän pelimaailman enimmäiskokoa laajennettiin myös valtavasti. Se oli aikaisemmin suunnilleen 21 kuutiokilometriä, josta se nousi noin 88,000,000 kuutiokilometriin.

2.2 Pelillistäminen

Pelimoottoreiden käyttötarkoitus on laajentunut pelkästä pelienkehittämisestä huomattavasti. Nykyään pelimoottoreita hyödynnetään monipuolisesti useilla aloilla pelillistämällä eri asioita, kuten esimerkiksi rakennussuunnitelmia.

Pelillistäminen on peleissä olevien elementtien käyttämistä muissa ympäristöissä, kuten opinnoissa, työskentelyssä, markkinoinnissa ja jopa elämäntapoja voidaan pelillistää. Pelillistämisellä voidaan koittaa tehdä jostain asiasta mielenkiintoisempaa, motivoivampaa tai inspiroivaa. (Chou 2015.)

Pokémon Go -peli on hyvä esimerkki sovelluksesta, joka on pelillistänyt niinkin arkipäiväisen asian kuin kävelemisen. Pelillistäminen on tehty lisäämällä peliin useita pelin sisäisiä palkintoja ja tavoitteita, jotka voidaan saavuttaa vain kävelemällä. Yksinkertaisimmillaan pelillistäminen tapahtuu pelkästään ajatuksen tasolla. Sitä voidaan käyttää vaikka itsensä motivoimiseen asettamalla itselle erinäisiä tavoitteita, määränpäitä ja niitä suorittamalla saatavia palkintoja (Chou 2015). Opiskelussa ja työpaikoilla olevien koulutusten pelillistäminen virtuaaliympäristöihin voi parantaa niiden kiinnostavuutta ja mieleenpainuvuutta tuomalla opittavat asiat konkreettisemmin esille.

Arkkitehtuurissa pelillistämisellä voidaan myös auttaa asioiden havainnollistamista. Rakennuksen 3D-mallin tuominen pelimoottoriin ja sen tutkiminen esimerkiksi virtuaalilasien avulla voi helpottaa mm. siinä mahdollisesti piilevien

käytettävyys- ja turvallisuusongelmien havainnointia (Pelillistäminen herättää motivaation 2022). Myös suunnittelussa pelillistämisestä on hyötyä niin kaupunki- kuin sisustussuunnittelussakin. Esimerkiksi rakennusten, kalusteiden, materiaalien tai äänimelun pelillistämisellä voidaan välttyä erilaisilta suunnitteluvirheiltä (Architecture s.a., Unreal Engine.)

Elokuva- ja tv-tuotannossa Unreal Engine 5 -pelimoottoria käytetään hyvin monipuolisesti. Erikoistehosteiden lisäksi sitä käytetään myös maailmojen luontiin, valaistukseen sekä olentojen ja ihmisten lisäämiseen kohtauksiin. Tämä myös nopeuttaa elokuva- ja tv-tuotantoa, sillä nämä Unreal Enginellä lisätyt asiat renderöidään reaaliajassa ja se mahdollistaa myös niiden nopean muokkaamisen. (Film-television s.a., Unreal Engine.)

Suoratoisto-lähetykset ja live-tapahtumat ovat myös hyödyntäneet näitä ominaisuuksia. Esimerkiksi suoratoisto-lähetyksiin voidaan lisätä Unreal Enginellä luotuja interaktiivisia animaatioita sekä efektejä. Vastaavasti live-esityksissä näitä voidaan näyttää projektoreilla heijastamalla tai näytöillä. (Broadcast-live-events s.a., Unreal Engine.)

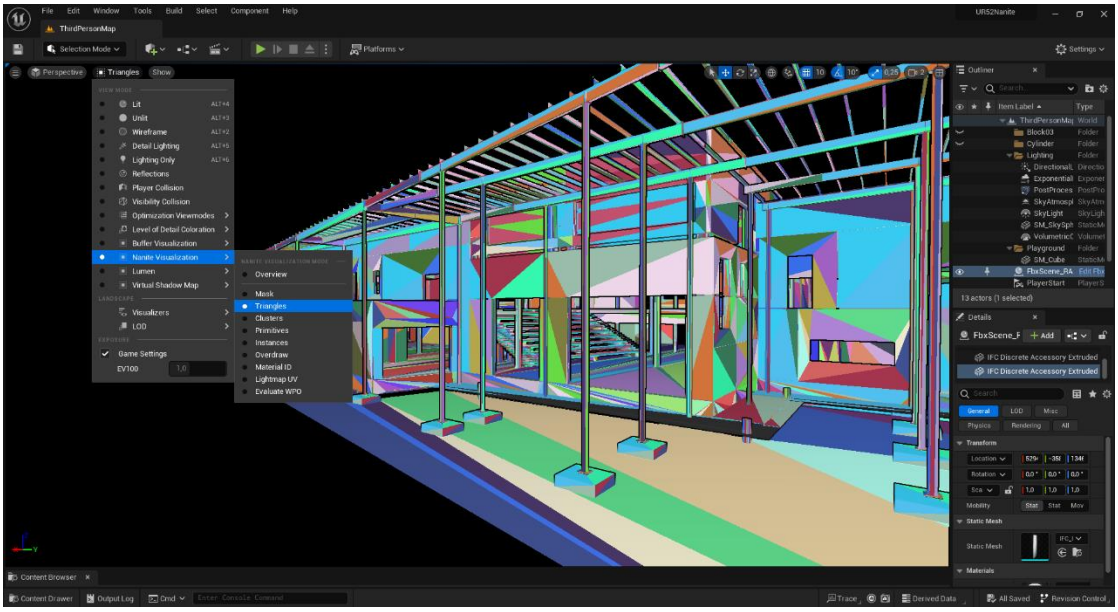
3 NANITE

Nanite on Unreal Engine 5 -pelimoottorissa oleva geometriajärjestelmä, joka käyttää uudenlaista Nanite-verkkoa ja renderöintitekniikkaa pelimoottorissa olevien objektien renderöintiin. Sen avulla voidaan renderöidä älykkäästi pelkästään näkyvillä olevat yksityiskohdat pikselin tarkasti. Sen avulla suurten ja geometrialtaan monimutkaisten objektien tai myös suurten objektimäärien renderöinti ja käyttäminen peleissä on mahdollista. (Nanite virtualized geometry in unreal engine s.a.)

Nanite-verkko on staattinen verkko, joka käyttää Nanite-tekniikkaa. Pohjimmiltaan se on kolmioverkko. Nanite-verkon käyttämä tietoformaatti käyttää pakattuja tiedostoja, jotka voivat olla hyvinkin yksityiskohtaisia. Nanite renderöi Nanite-verkossa olevat tiedostot älykkäällä ja tehokkaalla tavalla aina kulloisenkin kameranäkymän mukaan. Mitään mikä on kameranäkymän ulkopuolella, ei renderöidä. Renderöinnin laatu vaihtelee kameranäkymän ja kohteen etäisyyden mukaan. Kameran ollessa kohteen lähellä, kuva renderöidään jopa sen

alkuperäisen tarkkuuden mukaisesti. Kun taas kauemmaksi siirryttäessä saman kohteen renderöimiseen käytetään huomattavasti vähemmän tehoa ja tarkkuutta. Silti kyseinen kuva näyttää kauempaa katsottunakin hyvä laatuiselta, eikä siitä häviä liikaa yksityiskohtia. (Nanite virtualized geometry in unreal engine s.a.)

Unreal Engine 5 -käyttöliittymässä Nanite voidaan visualisoida. Kuvassa 1 Nanite Triangles -visualisointi on laitettu päälle ja se näyttää Nanite-verkon kolmiot eri väreillä käyttöliittymän näkymässä.



Kuva 1. Nanite Triangles-visualisointi tyylin valinta

Nanite-visualisoinnissa voidaan myös laittaa Overview-asetus päälle, jolloin käyttöliittymässä näkyy samanaikaisesti useampia eri visualisointityylejä (kuva 2).

Kuva 2. Nanite Overview-visualisointi Unreal Engine 5 -käyttöliittymässä

Nämä tyylit visualisoivat näkymän monella tapaa esimerkiksi värjäämällä eri objektit tai materiaalit niiden monimutkaisuuden mukaan. Overdraw-visualisointityyli näyttää ne kohdat, missä geometriaa on kameraan nähden päällekkäin. Tällaiset kohdat voivat aiheuttaa ylikuormitusta tietokoneelle.

Materiaali määrittää 3D-mallien pintojen ominaisuudet ja sitä voidaan ajatella maalina, joka lisätään objektien pinnoille. Nanitea voidaan käyttää vain sellaisten materiaalien kanssa, joiden sekoitustilaksi on valittu joko Opaque tai Masked. Jos Nanitea koitetaan laittaa sellaiseen 3D-malliin, jossa on sellainen materiaaliasetus päällä, jota Nanite ei tue, niin silloin Unreal Engine vaihtaa kyseisen materiaalin Unrealin vakiomateriaaliin, jota Nanite tukee. Opaque-asetusta suositellaan käytettäväksi ennemmin kuin Masked-asetusta, sillä se vie huomattavasti vähemmän tehoja tietokoneelta.

Unreal Engine 5.1 -päivityksessä Nanitelle tuotiin tuki myös lehvistöjen geometrian kanssa. Nanite voidaan nyt myös aktivoida ruohikkoon, lehtiin ja muihin kasveihin. Lehvistöissä oleva hajanainen geometria rikkoo Nanitessa olevan ominaisuuden, joka vähentää yksityiskohtia kaukaa katsottuna ja se samalla ohentaa lehvistön geometriaa. Tähän kuitenkin tuli tämän päivityksen mukana uusi Preserve Area -ominaisuus, joka estää kyseisen ilmiön. Preserve Area -ominaisuutta suositellaan käytettävän aina ja pelkästään lehvistöissä olevissa Nanite-verkoissa. (Nanite virtualized geometry in unreal engine s.a.)

Unreal Engine 5.3 -päivityksen mukana Nanitelle tuli tuki myös maastogeometrian kanssa. Unreal Engine:ssä maailma muodostuu lohkoista. Joissain tapauksissa näiden lohkojen välille voi muodostua näkyviä saumoja maastoon. Tämä 5.3 päivitys toi myös Nanite Skirts -asetuksen, joka saa viereisten lohkojen maastossa olevat Nanite-verkot risteämään keskenään ja estää näiden välisten saumojen muodostumisen. (Using Nanite with landscapes in Unreal Engine s.a.)

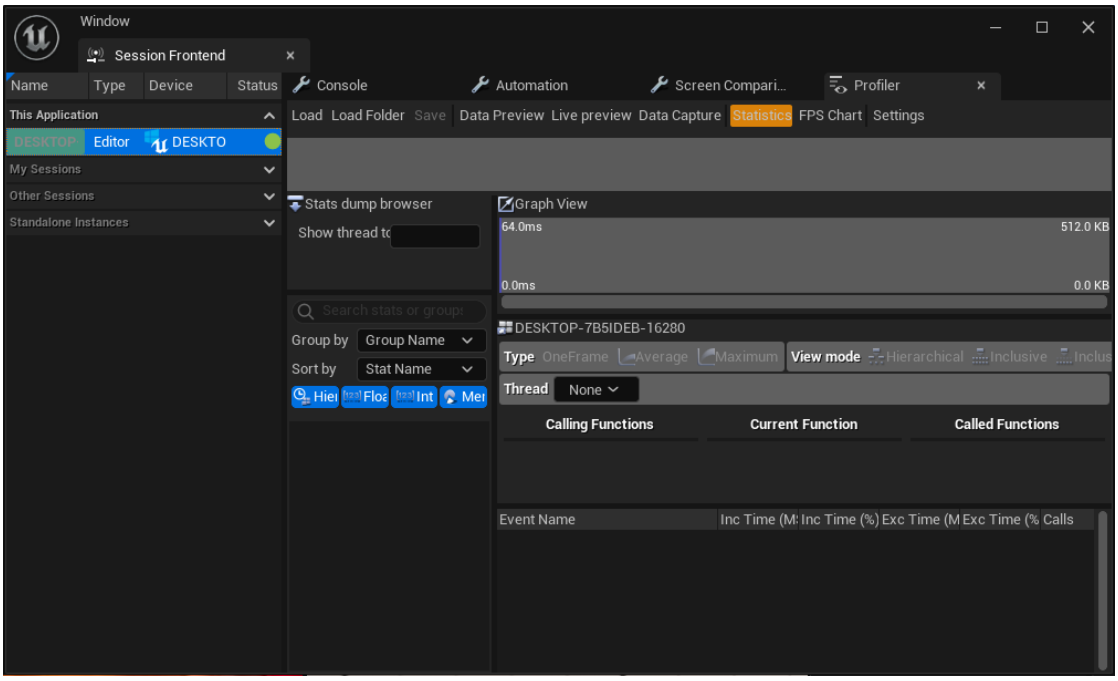
4 UNREAL ENGINE 5 PROFILOINTITYÖKALUT

Unreal Engine 5:ssä on useita profilointityökaluja, joilla voidaan tallentaa ja analysoida Unreal Engine 5 -pelimoottorilla tehtyjä projekteja ja valmiita sovelluksia. Näiden työkalujen avulla voidaan tarkkailla ja tallentaa tietoa suorituskyvystä projektien eri osa-alueilta. Niitä analysoimalla voidaan havaita mahdollisia ongelmakohtia, paikantaa ja korjata virheitä, sekä optimoida niiden avulla projekteja.

Profiler on yksi näistä työkaluista ja se on tarkoitettu suorituskyvyn seuraamiseen. Sillä voidaan tallentaa ja tarkastella projektin suorituskyvyn statistiikkaa sen eri osa-alueilta. Profiler-työkalun avulla voidaan myös saada keskiarvoja monille näistä statistiikoista. Profiler-työkalu löytyy Session Frontend -välilehdeltä, jonka saa auki Unreal Enginen Tools-valikosta (kuva 3).

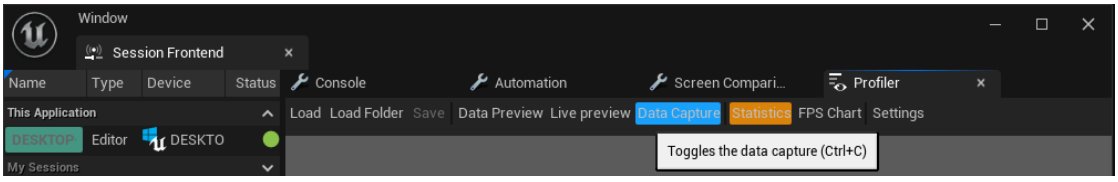
Kuva 3. Tools-valikossa oleva Session Frontend -valinta

Session Frontend -ikkunasta löytyy Profiler-välilehti, josta Profiler-työkalua pääsee käyttämään. Profiler-työkalun käyttäminen aloitetaan valitsemalla Session Frontend -ikkunan vasemmasta reunasta käynnissä oleva projekti, josta tietoja halutaan tallentaa (kuva 4).



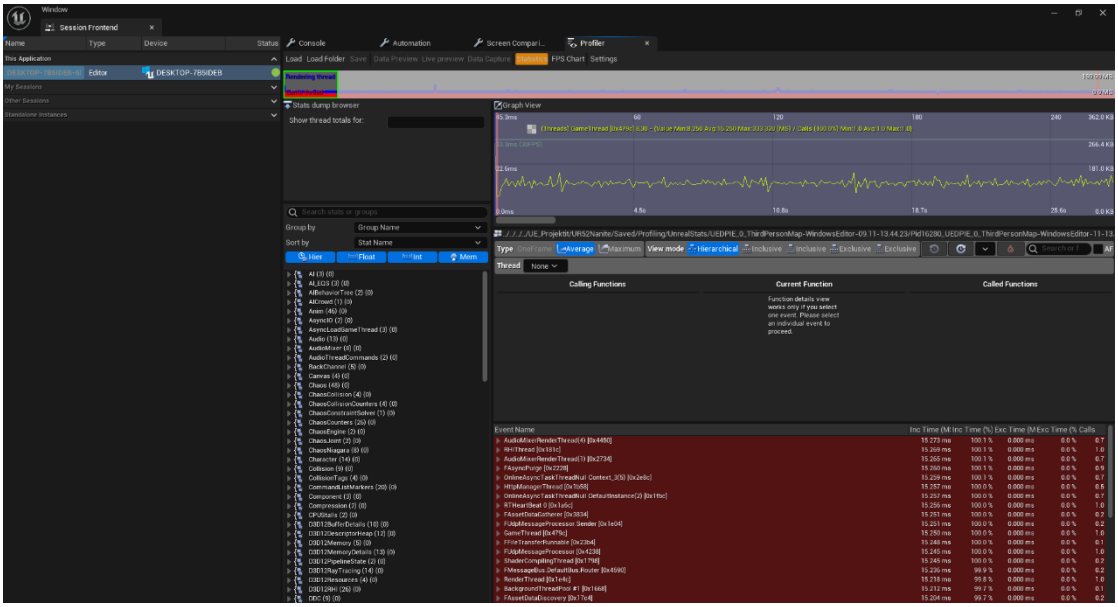
Kuva 4. Session Frontend -ikkunan Profiler-välilehti

Tietojen tallentaminen aloitetaan Session Frontend -ikkunan Profiler-välilehdeltä löytyvästä Data Capture -painikkeesta (kuva 5). Tietojen tallentamisen aikana projekti voi olla joko editointi- tai toistotilassa.



Kuva 5. Profiler-välilehden Data Capture -painike

Data Capture -toiminto kerää tietoa käynnissä olevasta projektista ja tallentaa nämä tiedot tiedostoon, jonka voi avata Profiler-välilehden Load-painikkeesta. Tiedosto on kyseisen projektin kansion ”Saved/Profiling/UnrealStats/Projektin nimi”-polussa. Tiedosto sisältää viivadiagrammin, sekä Event Name -listan prosesseista ja tapahtuma säikeistä (kuva 6).



Kuva 6. Profiler-työkalulla tallennettua tietoa

Viivadiagrammista voidaan valita kuvakohtaisesti yksi tai useampi kuvanpäivitys. Viivadiagrammista valituissa osioissa olevat tapahtumat tulevat Event Name -listaan. Tästä listasta voidaan nähdä, kuinka kauan eri tapahtumien suorittamiseen on kulunut aikaa ja kuinka useasti ne ovat tapahtuneet.

Unreal Insights on erillinen profilointijärjestelmä, joka on telemetrian talteenotto- ja analysointiohjelmisto. Se tulee valmiiksi asennettuna Unreal Engine 5 -pelimoottorin mukana. Se integroituu Unreal Engine 5 -pelimoottoriin ja sillä voidaan tallentaa Unreal Engine 5 -projektien ja valmiiden sovellusten tapahtumia suurilla tiedonsiirtonopeuksilla. Sen avulla voidaan tarkasti havaita ja paikantaa projektin eri osa-alueilta kohtia, jotka vaativat optimointia. Unreal Insights pääkomponentit ovat Trace events, The Unreal Trace Server ja Unreal Insights. Trace events -komponentin tehtävä on määrittää tapahtumat. Se sisältää EventName- ja FieldName- parametrit, jotka määrittävät tapahtuman ja osa-alueen, joka sisältää kyseisen tapahtuman. The Unreal Trace Server taas nauhoittaa ja tallentaa tapahtumat projektista. Unreal Insights visualisoi ja analysoi nämä tapahtumat ja niiden sisältämät tiedot. (Unreal Insights in Unreal Engine s.a.)

Unreal Insights sisältää monipuolisia työkaluja Unreal Engine -projektien eri osa-alueiden tutkimista ja optimointia varten. Timing Insights on yksi näistä työkaluista. Sillä voidaan visualisoida ja tutkia dataa kuvakohtaisesta suoritus-

kyvystä ja ruudunpäivitysnopeuksista projektien eri osa-alueilta. Unreal Insights -työkalun avulla voidaan havaita, kuinka paljon aikaa projekti käyttää eri tehtävien suorittamiseen. Tärkeimmät osa-alueet, joita voidaan tutkia Timing Insights:lla ovat näytönohjaimen ja prosessorin viemät resurssit. (Timing Insights in Unreal Engine 5 s.a.)

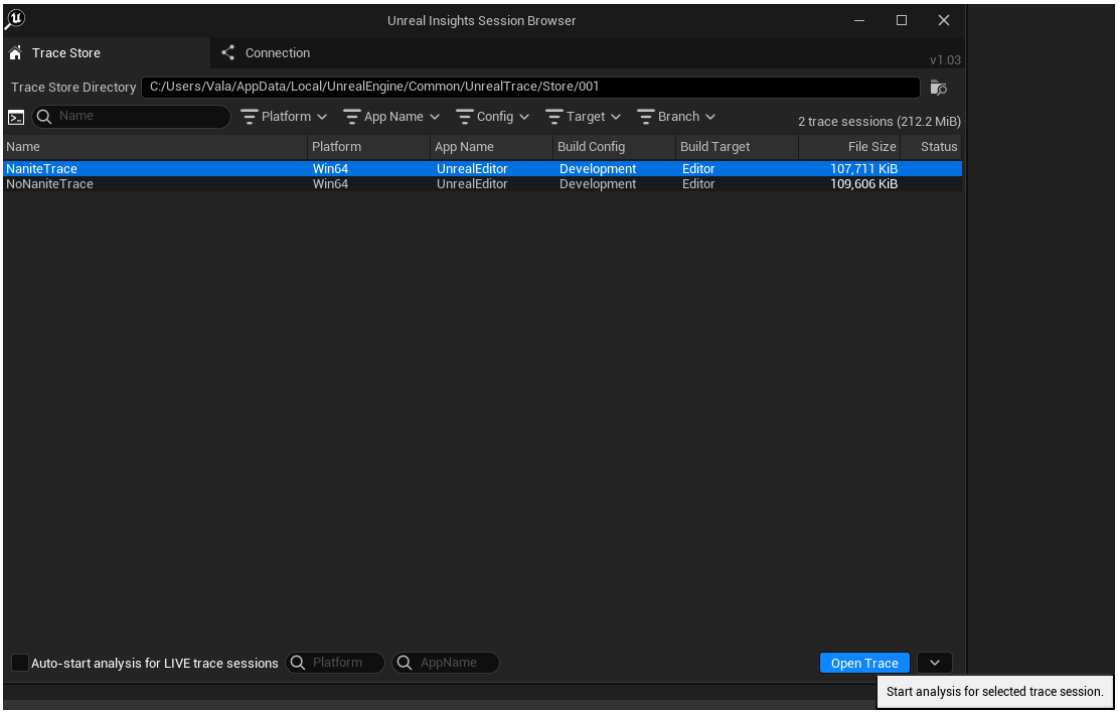
Memory Insights on taas työkalu, jonka avulla voidaan tutkia projektien muistinkäyttöä. Se nauhoittaa projektin jokaisen tapahtuman ja niiden muistinkäytön. Asset Loading Insights -työkalulla voidaan havaita, kuinka paljon aikaa projektin eri toimintoihin, kuten esimerkiksi mallien lataamiseen Unreal Engine 5 -pelimoottoriin kuluu. Unreal Insights sisältää myös työkaluja, joiden avulla voidaan seurata ja optimoida esimerkiksi projektien verkkoliikennettä ja pakettien käsittelyä. (Unreal Insights in Unreal Engine s.a.)

Unreal Insights voidaan käynnistää Unreal Engine 5 -editorin päänäkymästä oikean alareunan Trace-valikon Unreal Insights (Session Browser) -valinnasta (kuva 7). tai Unreal Engine 5 -kansiosta polusta ”Engine/Binaries/Win64/UnrealInsights.exe”, jota kautta sitä voidaan käyttää Unreal Engine 5:llä tehtyjen sovellusten profilointiin.

Kuva 7. Unreal Engine 5 -käyttöliittymän Trace-valikko

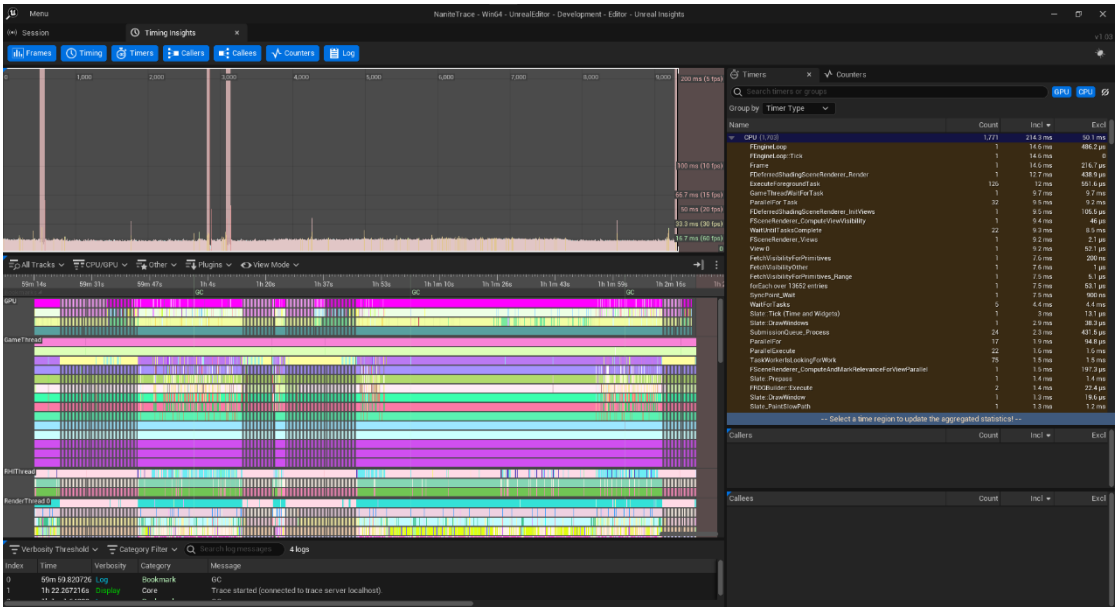
Unreal Insights -profilointi Unreal Engine 5 -editorissa aloitetaan valitsemalla Trace-valikossa olevasta Channels-osiosta, mitä osa-alueita halutaan analysoida (kuva 7). Profilointi voidaan aloittaa kuvassa 7 näkyvästä Trace-valikon

oikeanpuoleisesta Start Trace -valinnasta tai Trace-valikossa olevasta Start Trace -painikkeesta. Toiminto aloittaa analysoimaan ja tallentamaan valittujen osa-alueiden tapahtumia tiedostoon. Tallentamisen voi päättää samasta painikkeesta. Tallennetut tiedostot löytyvät Unreal Insights Session Browser:sta, josta käsin niissä olevia tietoja päästään tarkastelemaan ja analysoimaan (kuva 8).



Kuva 8. Unreal Insights Session Browser -ikkuna

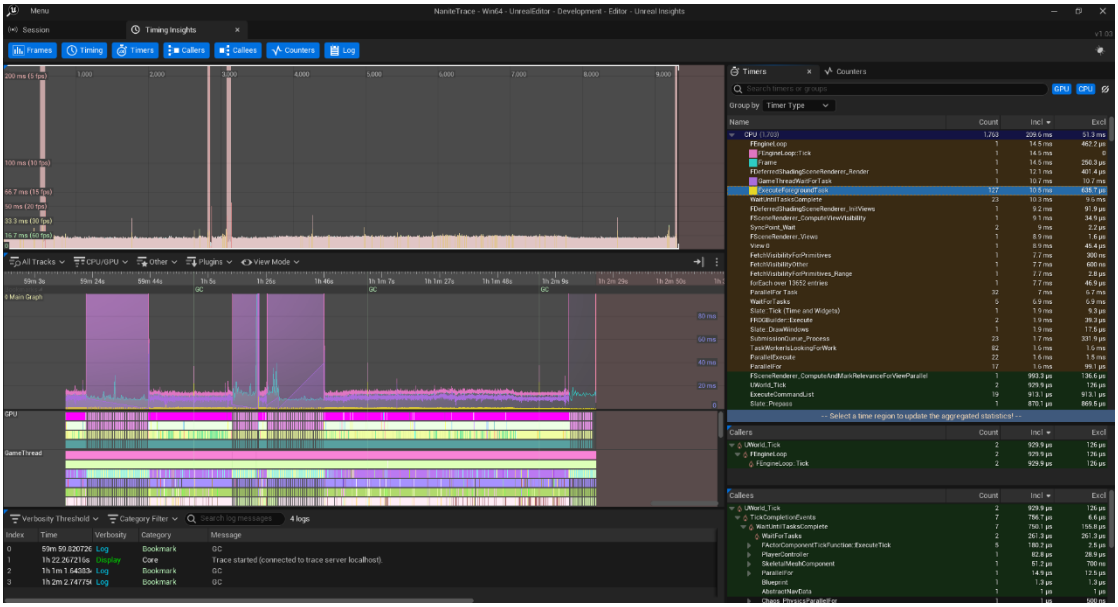
Avatusta tiedostosta Unreal Insights näyttää Session-välilehdellä perustiedot projektista ja siitä tehdystä analyysistä. Timing Insights -välilehdellä on pylväsdiagrammi kuvanpäivitysnopeuksista ja aikajana tapahtumajoukosta, joka sisältää kaikki tallennuksen aikana tapahtuneet tapahtumat (kuva 9).



Kuva 9. Unreal Insigts:n Timing Insights -välilehti

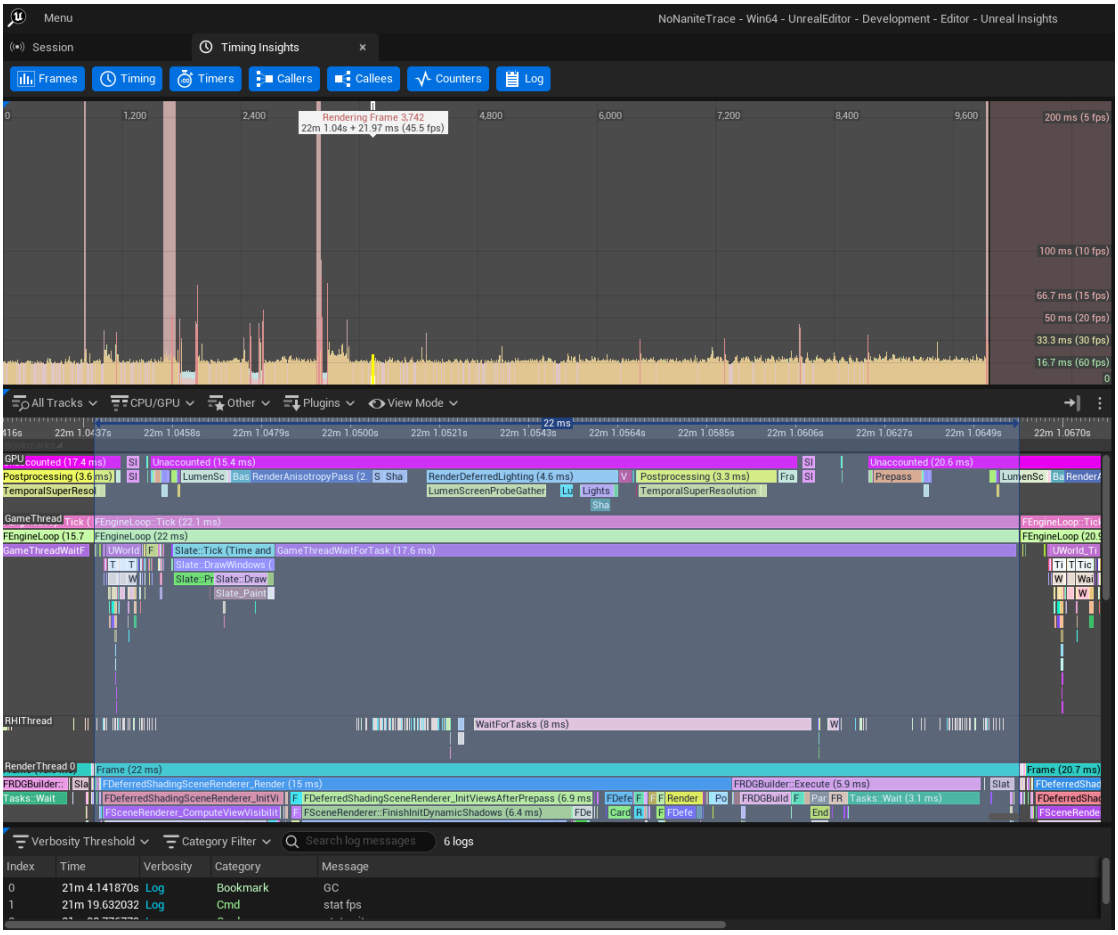
Tapahtumajoukon alapuolella on lokipaneeli, joka näyttää kaikki kutsujen luomat lokit projektin analyysistä. Kuvan 9 oikeassa reunassa on Counters -lista tapahtumista ja niiden määristä, sekä Timers -lista, joka sisältää näytönohjaimen ja prosessorin suorittamat toiminnot ja niihin kuluneen ajan. Niiden alapuolelta löytyy lista funktioiden kutsuista ja kutsutuista funktioista.

Timing Insights -välilehdeltä päästään analysoimaan tietoja tarkasti ja monipuolisesti. Valitsemalla Timers-listassa listattuja funktioita, muodostuu niistä viivadiagrammi aikajanan alapuolelle (kuva 10). Viivadiagrammissa on valmiiksi FEngineLoop-, FEngineloop::Tick- ja Frame-käyrät.



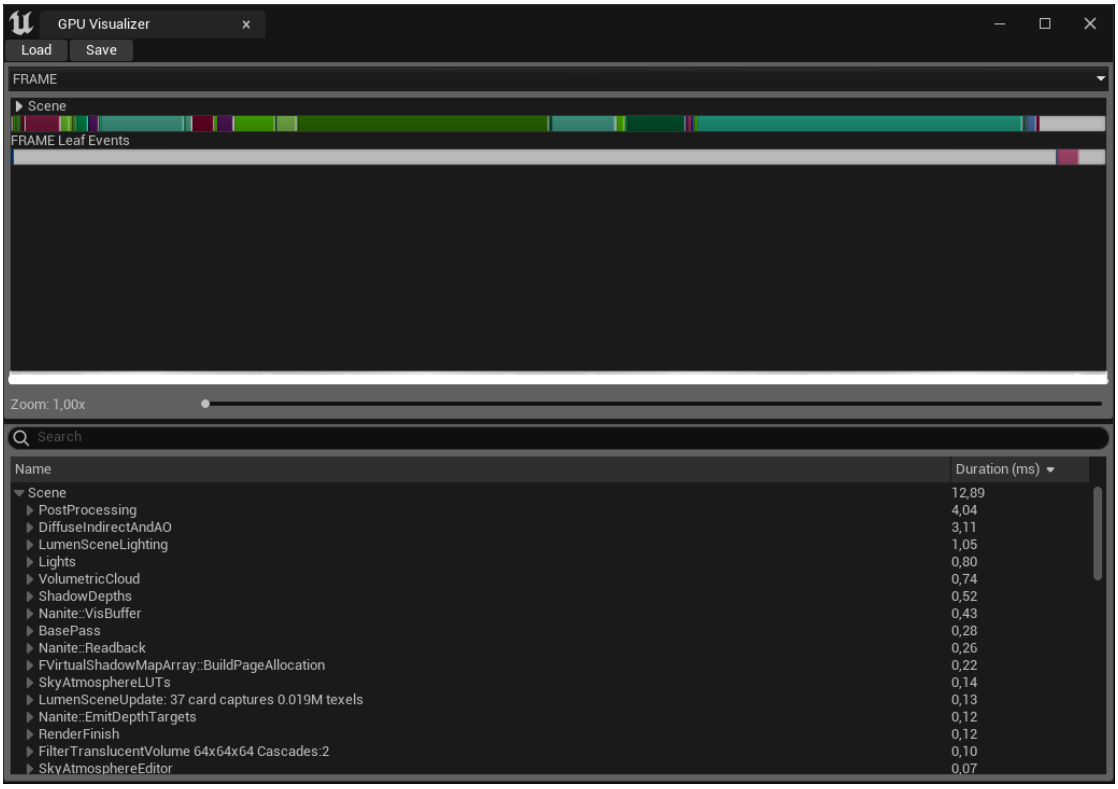
Kuva 10. Timing Insights -viivadiagrammi ja Timers-lista

Erikokoisia tapahtumajoukkoja pääsee analysoimaan tarkemmin valitsemalla aikajanalta sen aikavälin, jota halutaan tarkastella. Yksittäisen kuvanpäivityksen aikaisia tapahtumia voidaan taas analysoida valitsemalla kuvanpäivityspylväsdiagrammista haluttu kohta (kuva 11). Timers- ja Counters-listat, sekä listat funktiokutsuista ja kutsutuista funktioista näyttävät tiedot valitulta aikaväliltä.



Kuva 11. Yhden kuvanpäivityksen aikaiset tapahtumat

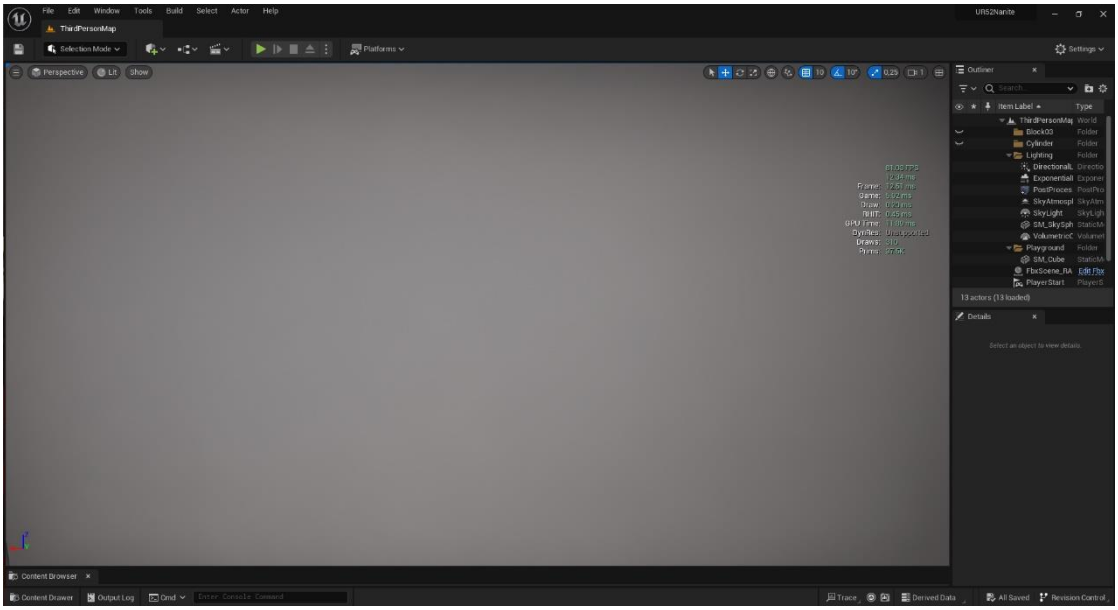
Näytönohjaimen profilointityökalu GPU Visualizer on hyvä työkalu ongelmakohtien paikantamiseen. GPU Visualizer avataan konsolikomennolla ProfileGPU. GPU Visualizer sisältää listan näytönohjaimen käyttämistä toiminto- ja piirtoajoista eri objekteille (kuva 12).



Kuva 12. GPU Visualizer -ikkuna

Nämä toiminto- ja piirtoajat ovat siltä hetkeltä, kun ProfileGPU-komento on annettu, joten komento on hyödyllistä suorittaa useamman kerran tarkemman analyysin kannalta. Listasta näkyy vain sellaiset objektit, jotka vievät näytönohjaimen tehoja tarpeeksi, eli hyvin optimoidut objektit eivät välttämättä näy lainkaan listassa. Kyseinen lista kannattaa laittaa ajankesto-järjestykseen pisimmästä kestosta alkaen, niin optimointia vaativien ongelmakohtien löytäminen on helpompaa. Kuvassa 12 olevasta listasta nähdään näiden objektien nimet, joten ne on helpompi löytää ja optimoida Unreal Engine 5 -käyttöliittymässä.

Nopeaan profilointiin hyödylliset työkalut ovat Stat Unit - ja Stat FPS -komennot. Nämä komennot tuovat Unreal Engine 5 -käyttöliittymän oikeaan laitaan tilaston, joka näyttää hyödyllisiä arvoja reaaliajassa (kuva 13).



Kuva 13. Stat FPS - ja Stat Unit -tilastot Unreal Engine 5 -käyttöliittymässä

Stat FPS -komento näyttää FPS- ja Frame time -arvot. Frame time on aika, joka kuluu yhden kuvan luomiseen pelimoottorissa, kun taas FPS eli Frames Per Second kertoo, kuinka monta kuvaa luodaan sekunnin aikana. Hyödyllisimmät arvot, joita Stat Unit -komento tuo näkyviin ovat Frame time, Game time, Draw time ja GPU time. Game time on pelisäikeen arvo ja Draw time vastaavasti renderöintisäikeen arvo. Nämä arvot kertovat prosessorin käyttämän ajan kyseisten säikeiden ja niiden sisältämien toimintojen suorittamiseen. GPU time kertoo, kuinka kauan näytönohjaimella menee renderöidä pelimoottorin näkymä. Game time ja Draw time ovat ne arvot, joita voidaan verrata Frame time -arvoon. Jos Frame time -arvo on lähellä Game time -arvoa, niin pelisäie aiheuttaa todennäköisesti jonkin pullonkaulan, joka vaikuttaa pelimoottorin suorituskykyyn negatiivisesti. Jos taas Frame time -arvo on lähellä Draw time -arvoa, on silloin todennäköisesti vastaavanlainen pullonkaula renderöintisäikeessä. (Stat commands in Unreal Engine s.a.)

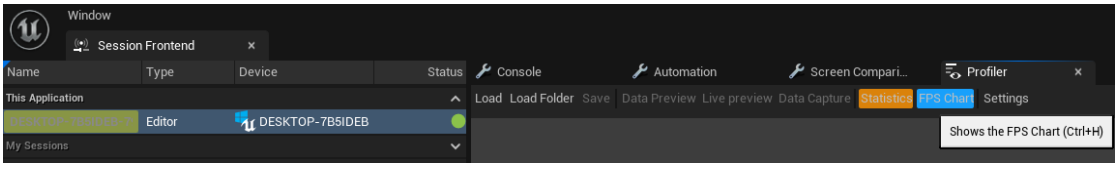
5 NANITEN HYÖTYJEN TUTKIMINEN

Naniten hyötyjen todentamista varten tein kaksi Unreal Engine 5 -projektia, joita vertailin keskenään. Projekteissa on käytetty Pieksämäellä sijaitsevan Hiekanpään yläkoulun rakennuksen 3D-mallia. Nämä projektit ovat muuten täysin identtiset, mutta vain toisessa projektissa oleviin 3D-malleihin on laitettu Nanite-tekniikka päälle. Tulen käyttämään tässä luvussa näistä projekteista ni-

miä Vanilla-projekti ja Nanite-projekti. Vanilla-projektissa käytetään Unreal Enginen perinteistä renderöintijärjestelmää ja yleisesti käytettyä staattista kolmioverkkomallia. Kun taas Nanite-projektissa käytetään Nanite-verkkomallia ja Nanite-renderöintijärjestelmää. Näiden projektien keskinäiseen vertailuun käytin Unreal Engine 5 -profilointityökalut luvussa esiteltyjä Unreal Insights ja Profiler -profilointityökaluja.

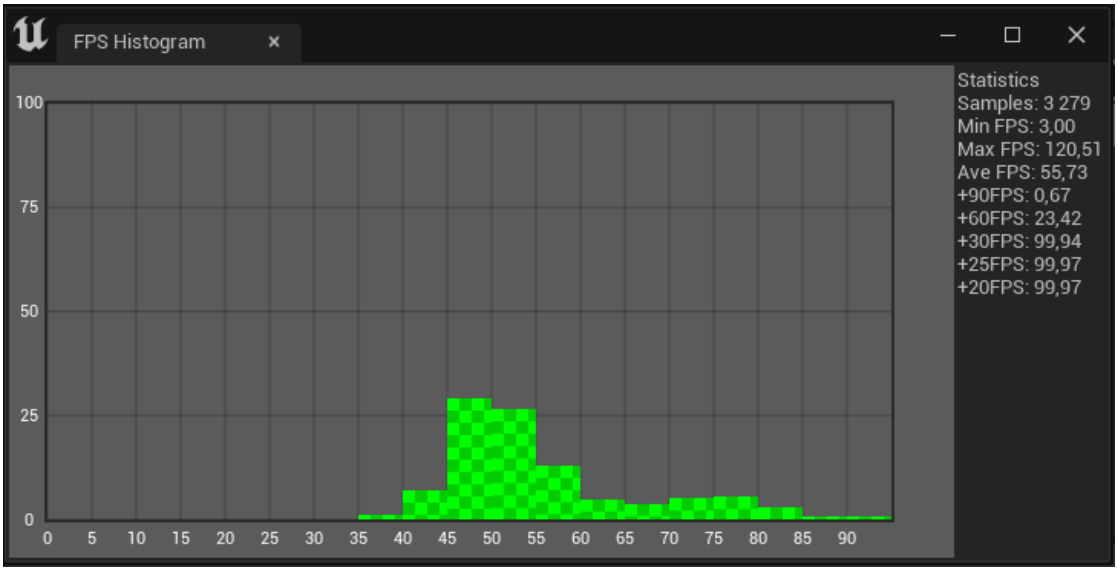
Aloitin vertailun tallentamalla tietoa Profiler-työkalulla kummastakin projektista niiden ollessa pelitilassa. Tallennusten aikana minulla ei ollut ylimääräisiä taustaprosesseja päällä, jotka olisivat voineet vaikuttaa prosessorin tai näytönohjaimen suorituskykyyn ja täten vaikuttaa testituloksiin negatiivisesti. Molemmista projekteista otin noin minuutin pituiset tallennukset. Tallennusten pituudessa eroa oli 1.7 sekuntia, joka täytyy ottaa huomioon vain kuvanpäivitysnopeuksien vertailussa, jossa vertailuajanjaksoa ei pystytä rajaamaan.

Kuvanpäivitysnopeuksien vertailuun Profiler-työkalu on erinomainen ja se havainnollistaa testitulokset selkeästi. Kuvassa 14 olevasta FPS Chart -painikkeesta aukeaa Profiler-työkalussa oleva FPS Histogram -ikkuna, joka näyttää tallennuksen aikaisen kuvanpäivityshistogrammin ja listan kuvanpäivitysnopeuksien statistiikasta.



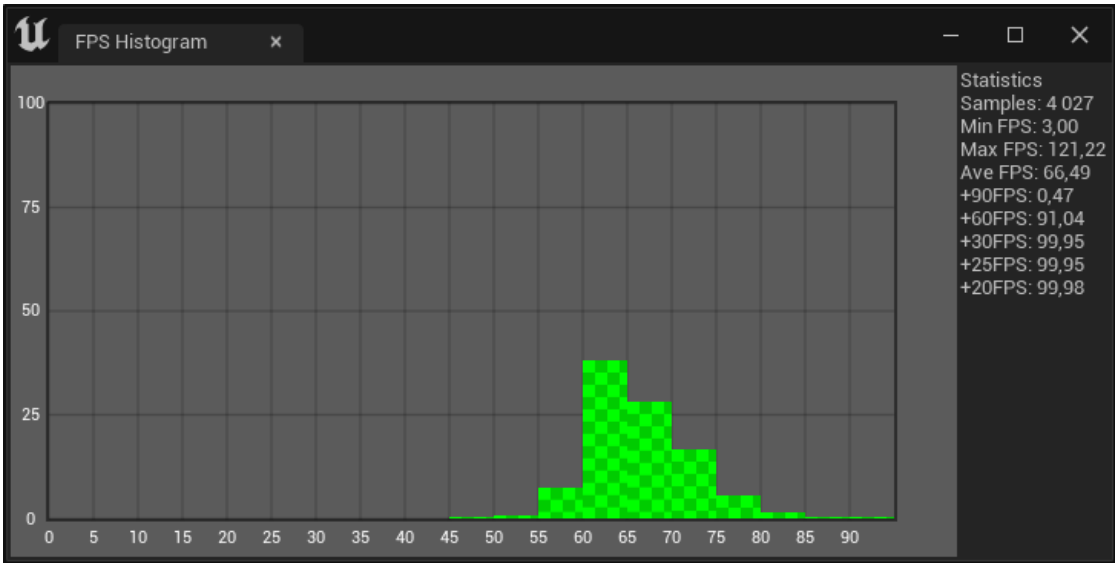
Kuva 14. Profiler-välilehdessä oleva FPS Chart -painike

FPS Histogram -ikkunan Statistics-listassa olevat statistiikat ovat seuraavanlaisia. Samples näyttää, kuinka monta kuvanpäivitystä on tapahtunut koko tallennuksen aikana. Min FPS on pienin ja Max FPS on suurin kuvanpäivitysnopeus tallennuksen ajalta. Ave FPS näyttää kuvanpäivitysnopeuden keskiarvon kokoajalta. +90FPS, +60FPS, +30FPS, +25FPS ja +20FPS näyttävät prosentuaalisen määrän, minkä ajan kuvanpäivitysnopeus on pysynyt kyseisen luvun yläpuolella (kuva 15).



Kuva 15. Vanilla-projektin kuvanpäivitysnopeuden histogrammi ja statistiikka

Kuvassa 15 on kuvanpäivityshistogrammi ja kuvanpäivitysnopeuksien statistiikkaa Vanilla-projektista. Nanite-projektista vastaavat tiedot voidaan nähdä kuvasta 16.

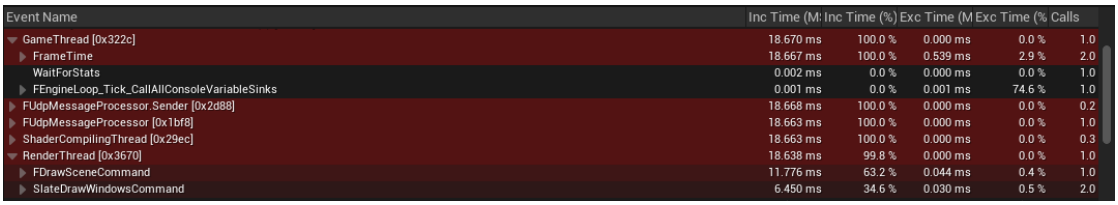


Kuva 16. Nanite-projektin kuvanpäivitysnopeuden histogrammi ja statistiikka

Kuvassa 16 voidaan nähdä, että Nanite-projektissa kuvanpäivitysnopeuden keskiarvo on 66,49 kuvaa sekunnissa. Vastaavasti kuvasta 15 nähdään, että Vanilla-projektissa kuvanpäivitysnopeuden keskiarvo on 55,73 kuvaa sekunnissa. Nanite-projektissa kuvanpäivitysnopeuden keskiarvo on siis 10,76 kuvaa sekunnissa parempi kuin Vanilla-projektissa, joka on noin 19,3% parannus kuvanpäivitysnopeuteen. Kuvassa 15 ja kuvassa 16 olevista kuvanpäivi-

tysnopeushistogrammeista voidaan myös nähdä, että Vanilla-projektissa kuvanpäivitysnopeuksissa on huomattavasti isompi heittely. Kun taas Nanite-projektin kuvanpäivitysnopeus on huomattavasti vakaampi. Kun tallennusten pituuden ero otetaan huomioon, Nanite-projektissa oli noin 645 kuvanpäivitystä minuutissa enemmän kuin Vanilla-projektissa.

Vertailin Profiler-työkalun avulla myös peli- ja renderöintisäikeiden suorittamiseen kuluneiden aikojen keskiarvoja. Kuvassa 17 näemme Profiler-työkalussa olevasta Event Name -listasta Vanilla-projektin tapahtumia. Listassa olevat tapahtumat, joita vertailemme ovat GameThread eli pelisäikeen toimintoihin käyttämä aika ja siihen sisältyvä FrameTime, joka on kuvanluontiin mennyt aika. Näiden lisäksi vertailemme RenderThread -aikaa eli renderöintisäikeen toimintojen suorittamiseen käytettyä aikaa.



Event Name	Inc Time (M	Inc Time (%)	Exc Time (M	Exc Time (%	Calls
GameThread [0x322c]	18.670 ms	100.0 %	0.000 ms	0.0 %	1.0
FrameTime	18.667 ms	100.0 %	0.539 ms	2.9 %	2.0
WaitForStats	0.002 ms	0.0 %	0.000 ms	0.0 %	1.0
FEngineLoop_Tick_CallAllConsoleVariableSinks	0.001 ms	0.0 %	0.001 ms	74.6 %	1.0
FUdpMessageProcessor.Sender [0x2d88]	18.668 ms	100.0 %	0.000 ms	0.0 %	0.2
FUdpMessageProcessor [0x1bf8]	18.663 ms	100.0 %	0.000 ms	0.0 %	1.0
ShaderCompilingThread [0x29ec]	18.663 ms	100.0 %	0.000 ms	0.0 %	0.3
RenderThread [0x3670]	18.638 ms	99.8 %	0.000 ms	0.0 %	1.0
FDrawSceneCommand	11.776 ms	63.2 %	0.044 ms	0.4 %	1.0
SlateDrawWindowsCommand	6.450 ms	34.6 %	0.030 ms	0.5 %	2.0

Kuva 17. Vanilla-projektista Profiler-työkalun tapahtumalista

Kuvassa 18 on Nanite-projektin vastaavat tapahtumat. Nämä tapahtumat ovat rajattu kummastakin projektista otetuista tallennuksista samanpituisiin ajanjaksoihin.

Event Name	Inc Time (M	Inc Time (%)	Exc Time (M	Exc Time (%	Calls
GameThread [0x479c]	14.642 ms	100.0 %	0.000 ms	0.0 %	1.0
FrameTime	14.639 ms	100.0 %	0.000 ms	0.0 %	2.0
WaitForStats	0.003 ms	0.0 %	0.000 ms	0.0 %	1.0
FEngineLoop_Tick_CallAllConsoleVariableSinks	0.001 ms	0.0 %	0.000 ms	0.0 %	1.0
RHIThread [0x181c]	13.593 ms	92.8 %	0.000 ms	0.0 %	1.0
RTHeartBeat 0 [0x1a6c]	13.471 ms	92.0 %	0.000 ms	0.0 %	1.0
Reserve Worker #2 [0x206c]	13.407 ms	91.6 %	0.000 ms	0.0 %	1.0
RenderThread [0x1e4c]	13.367 ms	91.3 %	0.000 ms	0.0 %	1.0
SlateDrawWindowsCommand	6.580 ms	49.2 %	0.000 ms	0.0 %	2.0
FDrawSceneCommand	6.451 ms	48.3 %	0.000 ms	0.0 %	1.0

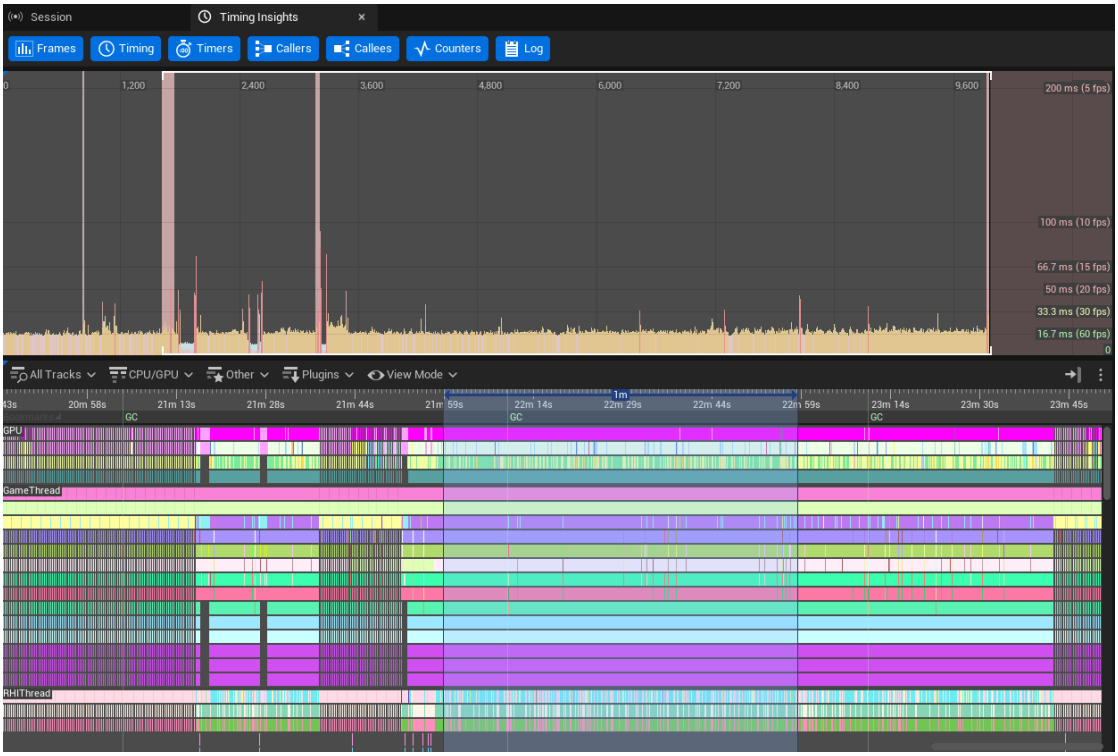
Kuva 18. Nanite-projektista Profiler-työkalun tapahtumalista

Kuvassa 17 näemme, että Vanilla-projektissa pelisäikeen (18.670ms), kuvanluonnin (18.667ms) ja renderöintisäikeiden (18.638ms) tapahtumien suorittamisen ajat ovat hyvin lähellä toisiaan. Nanite-projektissa vastaavasti pelisäikeen (14.642ms) ja kuvanluonnin (14.639ms) ovat hyvin lähellä toisiaan, mutta näiden ja renderöinti säikeen (13.367ms) välillä on huomattava ero (kuva 18). Tässä tapauksessa Nanite nopeuttaa pelisäikeen ja kuvanluonnin

funktioihin pelimoottorin käyttämää aikaa noin 21,6% ja renderöintisäikeen funktioihin käyttämää aikaa noin 28,3%.

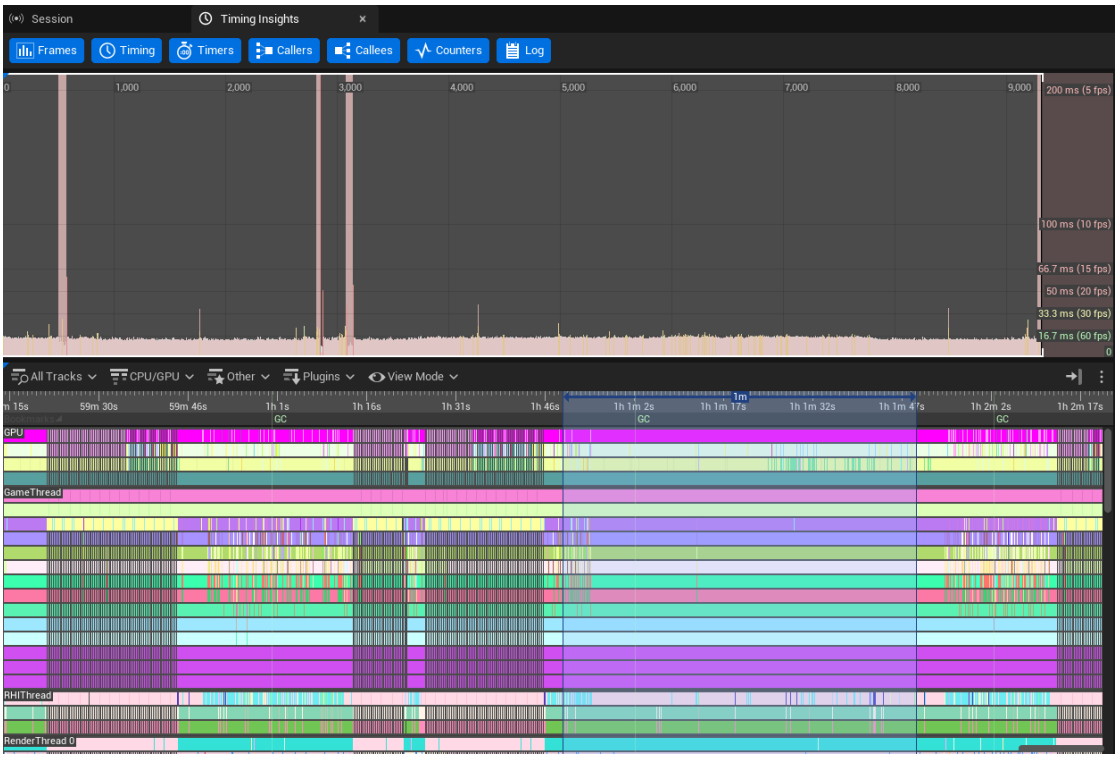
Loput vertailut Vanilla-projektin ja Nanite-projektin välillä tein Unreal Insights -profilointijärjestelmällä ja siinä olevalla Timing Insights -työkalulla. Otin kummastakin projektista Unreal Insights -työkalulla tallennukset, joihin molempiin olin valinnut tallennettaviksi osioiksi näytönohjaimen, prosessorin ja kuvanpäivityksen tapahtumat. Näistä tallennuksista rajasin vertailua varten yhtä pitkät ajanjaksot sellaisesta kohdasta, jonka aikana kummassakaan projektissa ei tapahtunut isompia muutoksia minkään osa-alueen suorituskyvyssä.

Näiden tallennusten analysointiin käytin Timing Insights -työkalua. Kuvassa 19 näemme Vanilla-projektista sinisellä viivalla ja alueella rajatun ajanjakson aikajanalta.



Kuva 19. Timing Insights -tallennus Vanilla-projektista

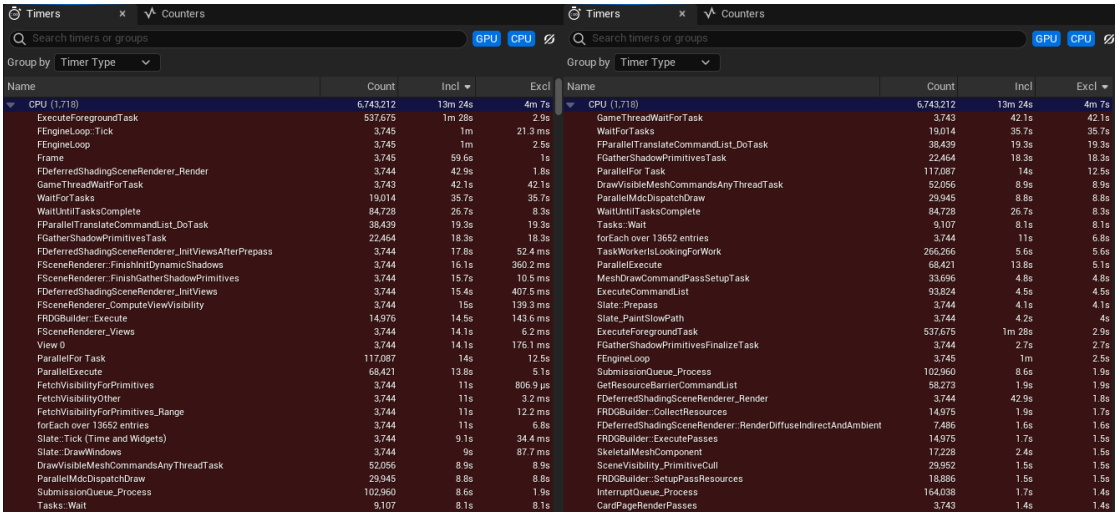
Kuvassa 19 olevan pylväsdiagrammin pylväiden värit vaihtelevat kuvanpäivitysnopeuden mukaan. Alueet, joissa kuvanpäivitysnopeus tippuu alle 20fps pylväät ovat punaisia ja alueet, joissa kuvanpäivitysnopeus tippuu alle 60fps ovat keltaisia. Yli 60fps kuvanpäivitysnopeuden kohdalla pylväät ovat vaaleanpunaisia. Kuvassa 20 on vastaava rajattu ajanjakso Nanite-projektista.



Kuva 20. Timing Insights -tallennus Vanilla-projektista

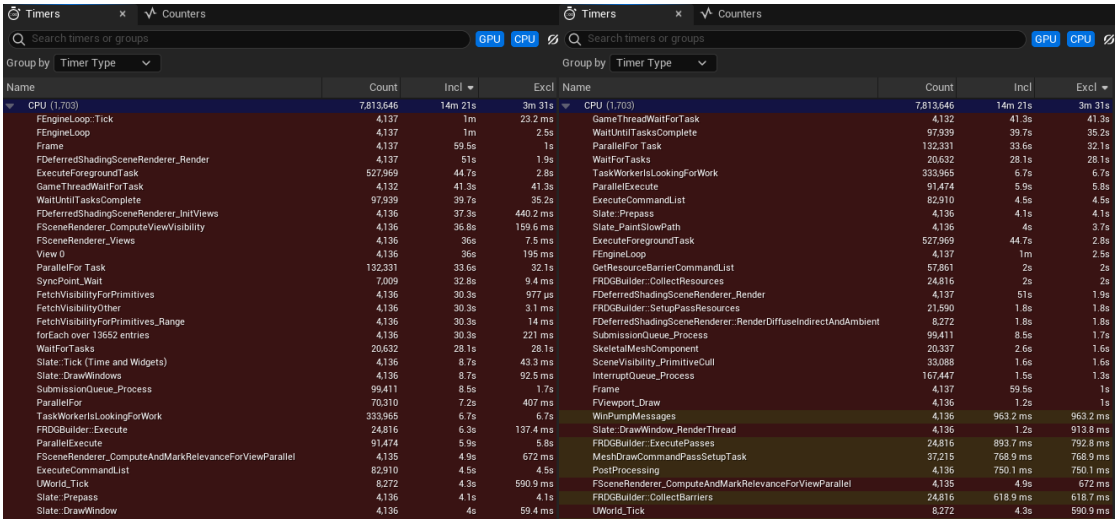
Kuvaa 19 ja kuvaa 20 vertailemalla voidaan nopeasti nähdä, että Vanilla-projektissa kuvanpäivitysnopeus on enimmäkseen pysynyt alle 60fps ja Nanite-projektissa lähes koko ajan yli 60fps.

Molemmista projekteista niistä rajattujen ajanjaksojen aikaiset tapahtumat ovat listattuna Timing Insights -työkalussa olevaan Timers-listaan. Timers-listasta näemme prosessorin ja näytönohjaimen funktioiden suorittamiseen menneet ajat ja määrät. Funktioiden ajat voidaan järjestellä Inclusive Time - tai Exclusive Time -aikajärjestykseen. Inclusive Time näyttää kyseisen funktion ja sen sisäisten funktioiden suorittamiseen menneen ajan ja Exclusive Time pelkästään kyseisen funktion suorittamiseen käytetyn ajan. Kuvassa 21 on Vanilla-projektin prosessorin funktiot Inclusive Time ja Exclusive Time -aikajärjestyksessä.



Kuva 21. Lista Vanilla-projektin prosessorin funktioiden suorittamiseen kuluneesta ajasta

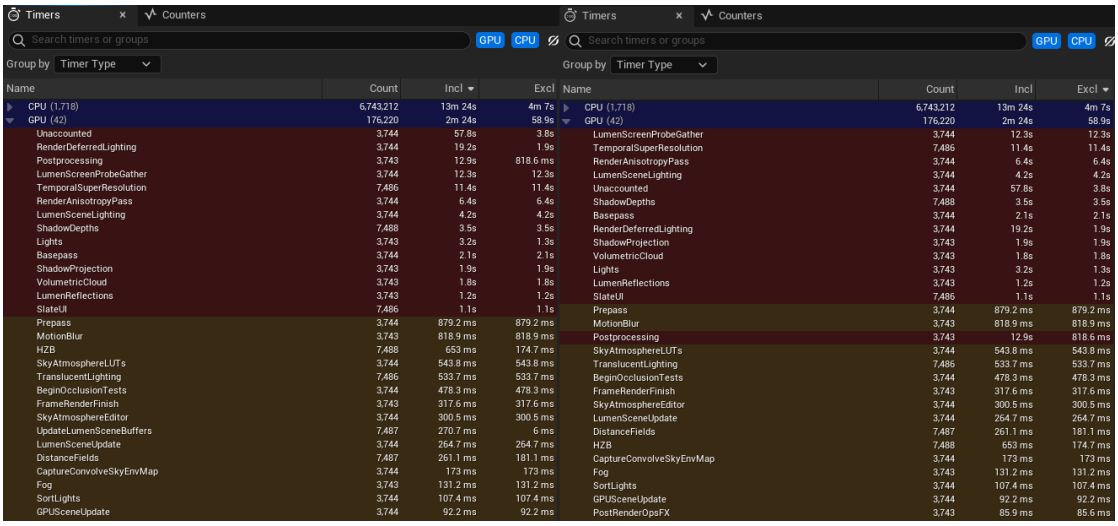
Vastaavasti Nanite-projektin prosessorin funktiot ovat kuvassa 22 Inclusive Time ja Exclusive Time -aikajärjestyksessä.



Kuva 22. Lista Nanite-projektin prosessorin funktioiden suorittamiseen kuluneesta ajasta

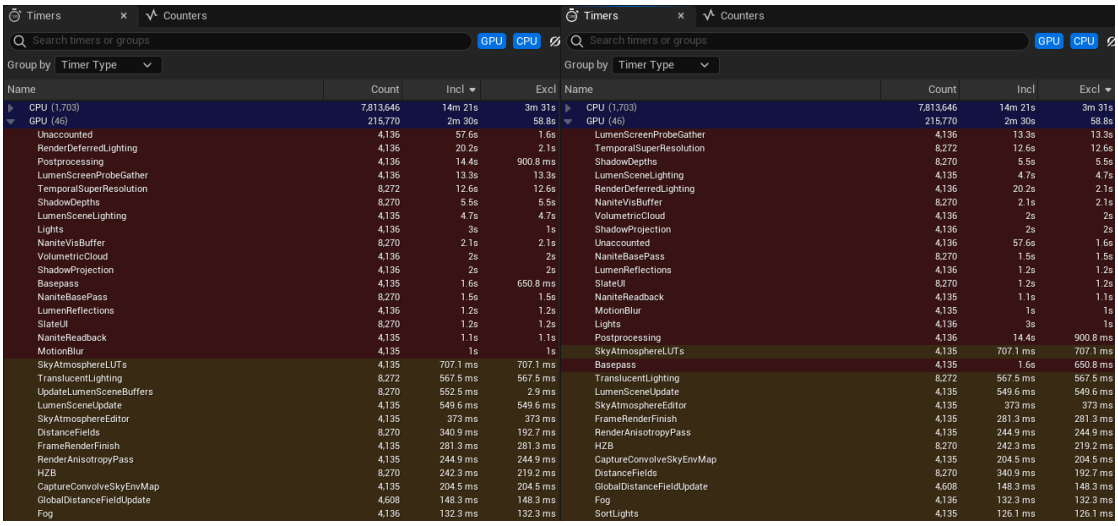
Prosessorin suorittamat ExecuteForegroundTask -funktiot, vievät näissä projekteissa valtaosan prosessorin suorituskyvystä. Vanilla-projektissa ExecuteForegroundTask -funktioiden suorittamiseen meni keskimäärin 0.164ms. Nanite-projektissa näiden funktioiden keskimääräinen suoritusaika oli 0.085ms. Tämä tarkoittaa, että Nanite-projektissa nämä funktiot suoritettiin keskimääräisesti 48,2% nopeammin kuin Vanilla-projektissa.

Vanilla-projektissa näytönohjaimen eri toimintojen määrä ja niihin kulunut aika näkyy kuvassa 23.



Kuva 23. Lista Vanilla-projektin näytönohjaimen funktioiden suorittamiseen kuluneesta ajasta

Nanite-projektista vastaavat tiedot on nähtävissä kuvassa 24.



Kuva 24. Lista Nanite-projektin näytönohjaimen funktioiden suorittamiseen kuluneesta ajasta

Näytönohjaimen yhteenlasketut, eri funktioiden suorittamiseen käytetyt ajat näyttävät hyvin samoilta kuvassa 23 sekä kuvassa 24, kun niitä vertaa keskenään. Suurin eroavaisuus onkin funktioiden määrissä ja niiden suorittamiseen keskimääräisesti kuluneessa ajassa.

Nanite-projektissa näytönohjain on suorittanut noin 10% enemmän funktioita kuin Vanilla-projekissa samassa ajassa. Nanite-projektissa näytönohjain on myös suorittanut eniten sen suorituskykyä vievät funktiot noin 5-10% nopeammin.

6 PÄÄTÄNTÖ

Opinnäytetyössä selvitettiin Nanite-tekniikan hyötyjä 3D-mallien käsittelemisessä Unreal Engine 5 -pelimoottorilla vertailemalla Nanite-renderöintitekniikkaa ja Nanite-verkkomallia Unreal Engine 5:n perinteiseen renderöintitekniikkaan ja kolmioverkkomalliin. Työssä käytiin läpi ja opastettiin Unreal Engine 5 -pelimoottorin eri profilointityökalujen käyttöä. Selvitettiin miten ja minkä tutkimiseen niitä voidaan käyttää, sekä opastettiin tarkemmin Unreal Insights -, Timing Insights - sekä Profiler-työkalujen käyttöä projektien eri osa-alueiden analysointitarkoituksia varten.

Projektien välisten vertailujen testituloksista voidaan todeta, että Nanite-tekniikan käyttäminen 3D-malleissa tuo huomattavia etuja tietokoneen suorituskykyyn. Projektissa, jossa Nanite-tekniikka oli käytössä, prosessorin ja näytönohjaimen funktioiden suorittamiseen meni merkittävästi vähemmän aikaa ja tehoa kuin projektissa, jossa Nanite-tekniikkaa ei käytetty.

Opinnäytetyö hyödyntää molempia toimeksiantajahankkeita. Game Studios -hankkeelle opinnäytetyö antaa uutta tietoa erilaisten laajojen kohteiden pelillistämisestä. ProRak-hankkeessa keskeisenä elementtinä on suurten pelimaalmojen tehokas käsittely. Tähän opinnäytetyössä tutkitut asiat tuovat uutta tietoa ja uusia työkaluja. ProRak-hankkeen tarkoituksena on havainnollistaa pelillistämisen keinoin erilaisia rakennussuunnitelmia. Kohteet ovat hyvin laajoja ja geometrialtaan monimutkaisia, jolloin niiden optimoinnille ja tehokkaalle käsittelylle on tarvetta.

Jatkokehitysehdotuksina Naniten hyötyjen tutkimisessa voitaisiin hyödyntää Unreal Engine 5:n uudempia ominaisuuksia ja käyttää projekteissa myös kasvillisuutta ja maastogeometriaa. Pelillistämällä näitä 3D-malleja ja lisäämällä skeneen eri funktioita ja käyttämällä niiden ohella myös Lumen vailaistus- ja heijastustekniikkaa. Myös Unreal Engine 5 -pelimoottorin eri profilointityökaluja voitaisiin hyödyntää monipuolisemmin näissä jatkotutkimuksissa. Unreal Insigths -järjestelmässä on paljon mahdollisia analysoitavia osa-alueita. Näitä kaikkia ei tässä opinnäytetyössä päästy havainnollistamaan. Pelillistämisessä Unreal Insights olisi todella hyödyllinen, koska sen avulla päästään analysoimaan funktioiden toimintaa hyvinkin yksityiskohtaisesti.

Projektissa suurimmat haasteet tulivat geometrialtaan monimutkaisten 3D-mallien tuonnissa Unreal Engine 5 -pelimoottoriin. Olisin halunnut lisätä ja

käyttää projekteissani vieläkin monimutkaisempia 3D-malleja. Minulla olisi ollut Hiekanpään yläkoulun kalusteiden, vesiverkoston, ilmastoinnin ja lämmitysputkien 3D-mallit käytössäni, mutta omalla tietokoneellani näiden mallien pelimoottoriin tuonti toi jo liikaa ongelmia koneen suorituskyvyn kanssa. Toinen ongelma oli saatavilla olevan tiedon vähäinen määrä aiheesta sillä, Unreal Engine 5:n omien dokumentaatioiden lisäksi aiheesta ei juurikaan ollut muualla tietoa.

Itselleni opinnäytetyöstä oli hyötyä, kun en ennen opinnäytetyön aloittamista ollut käyttänyt Unreal Engine -pelimoottorin mitään versiota. Opinnäytetyön ohella opin käyttämään Unreal Engine 5 -pelimoottoria ja sen profilointityökaluja monipuolisesti. Unreal Engine 5 -pelimoottoria tulen käyttämään tulevaisuudessakin eri projektien parissa ja toivottavasti pääsen hyödyntämään oppimiani tietoja myös työelämässä.

LÄHTEET

Architecture s.a. Unreal Engine. WWW-dokumentti. Saatavissa: https://www.unrealengine.com/en-US/feed/spotlights/architecture [viitattu 25.8.2023].

Broadcast-live-events s.a. Unreal Engine. WWW-dokumentti. Saatavissa: https://www.unrealengine.com/en-US/solutions/broadcast-live-events [viitattu 25.8.2023].

Chou Y.-K. 2015. Actionable Gamification: Beyond Points, Badges, and Leaderboards. Etelä-Carolina: CreateSpace.

Download s.a. WWW-dokumentti. Unreal Engine. Saatavissa: https://www.unrealengine.com/en-US/download [viitattu 20.11.2023].

Film-television s.a. Unreal Engine. WWW-dokumentti. Saatavissa: https://www.unrealengine.com/en-US/solutions/film-television [viitattu 25.8.2023].

Nanite virtualized geometry in unreal engine s.a. Unreal Engine. WWW-dokumentti. Saatavissa: https://docs.unrealengine.com/5.3/en-US/nanite-virtualized-geometry-in-unreal-engine/ [viitattu 15.10.2023].

Pelillistäminen herättää motivaation. 2022. Rakennuslehti 5.5.2022. Verkkolehti. Saatavissa: https://www.rakennuslehti.fi/2022/05/pelillistaminen-herattaa-motivaation [viitattu 20.8.2023].

Stat commands in Unreal Engine s.a. Unreal Engine. WWW-dokumentti. Saatavissa: https://docs.unrealengine.com/5.0/en-US/stat-commands-in-unreal-engine/ [viitattu 17.9.2023].

Studytonight s.a. Game Engine and History of Game Development. WWW-dokumentti. Saatavissa: https://www.studytonight.com/3d-game-engineering-with-unity/game-engine [viitattu 13.10.2023].

Timing Insights in Unreal Engine 5 s.a. Unreal Engine. WWW-dokumentti. Saatavissa: https://docs.unrealengine.com/5.1/en-US/timing-insights-in-unreal-engine-5/ [viitattu 20.9.2023].

Unreal Insights in Unreal Engine s.a. Unreal Engine. WWW-dokumentti. Saatavissa: https://docs.unrealengine.com/5.1/en-US/unreal-insights-in-unreal-engine/ [viitattu 20.11.2023].

Using Nanite with landscapes in Unreal Engine s.a. Unreal Engine. WWW-dokumentti. Saatavissa: https://docs.unrealengine.com/5.3/en-US/using-nanite-with-landscapes-in-unreal-engine/ [viitattu 22.11.2023]