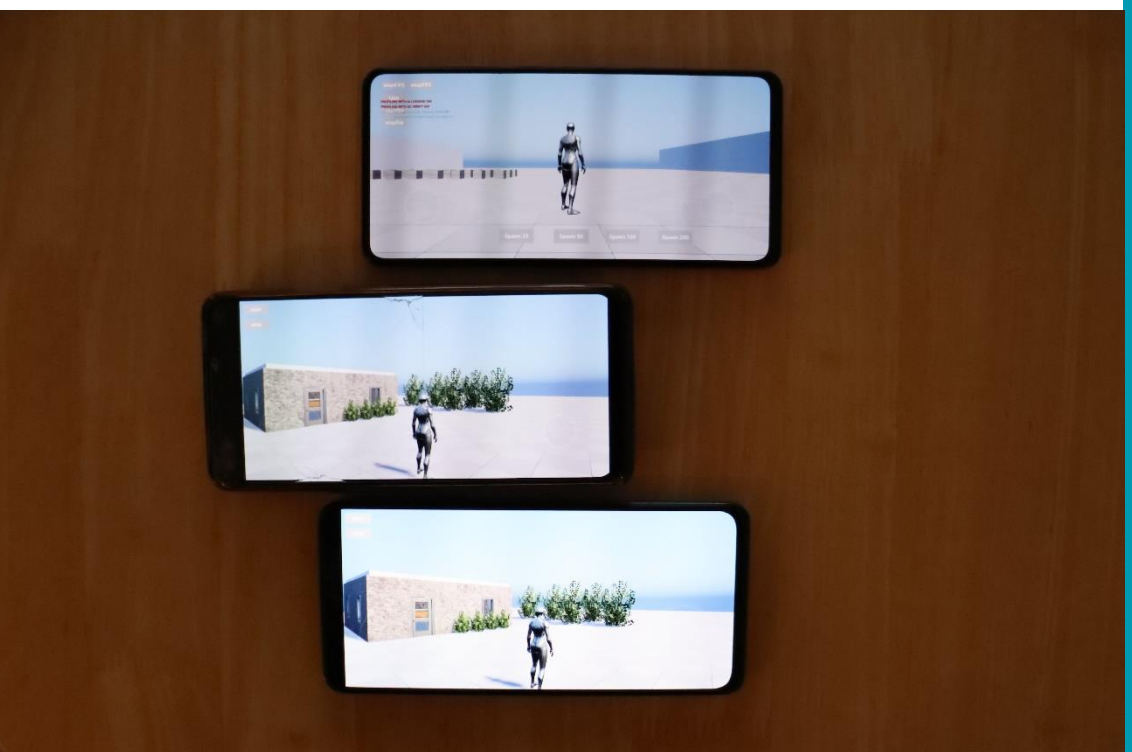
Maiju Alavuotunki

Unreal Engine ja mobiilipelin kehittäminen



Insinööri

Tieto- ja viestintätekniikka

Kevät 2024

KAMK • University of Applied Sciences

Tiivistelmä

Tekijä: Alavuotunki Maiju

Työn nimi: Unreal Engine ja mobiilipelin kehittäminen

Tutkintonimike: Insinööri (AMK), tieto- ja viestintätekniikka

Asiasanat: unreal engine, mobiilipelit, profilointi, optimointi, videopeli

Opinnäytetyön aiheena oli selvittää Unreal Engine 5 -pelimoottorilla kehitettyjen mobiilipelien profilointia. Tämä opinnäytetyö toteutettiin Critical Forcen toimeksiantona. Critical Force on kajaanilainen pelistudio, joka perustettiin vuonna 2012. Critical Force on kehittänyt Critical Ops -mobiilipelin, joka kehitettiin Unity-pelimoottoria käyttäen. Toimeksiantajaa kiinnosti, millaisia ovat laitteiden suorituskykyeroavaisuudet ja kuinka peleissä käytettävät eri toiminnot vaikuttavat suorituskykyyn.

Aineistona käytettiin Unreal Enginen dokumentaatiota ja webinaareja sekä muita internetistä löytyviä aineistoja. Opinnäytetyön teoriaosuudessa tutustuttiin mobiilipeleihin ja Unreal Engine 5 -pelimoottoriin. Työssä käytiin myös läpi joitain mobiilipelikehitykseen liittyviä keskeisiä huomioita sekä joitain yleisiä optimointisuosituksia. Käytännön osuudessa tehtiin erilaisia testejä, joita profiloimalla kartoitettiin suorituskykyä sekä ruutuajan muutosta. Opinnäytetyön käytännön osuudessa saadut tulokset osoittivat eri laitteiden suorituskykyeroavaisuuksia. Tuloksista voitiin myös nähdä eri toimintojen vaikutus pelin ruutuaikaan.

Testeissä saadut tulokset osoittivat eri toimintojen vaikutusta ruutuaikaan. Objekteja lisättäessä spawn-funktion aika kasvoi testilaitteilla melko tasaisesti. Kolmioiden ja piirtokutsujen testeissä todennettiin liiallisten kolmioiden ja piirtokutsujen hidastavan ruutuaikaa huomattavasti. Testilaitteiden välillä oli eroja ruutuajan vaikutuksessa sekä piirtämien kolmioiden ja piirtokutsujen määrässä. Jälkikäsittelyefektitesteissä ei havaittu mainittavia suorituskykyeroja. Forward- ja deferred-renderöintien testauksessa voitiin havaita deferred-renderöinnillä olevan alempi ruutuaika kuin forward-renderöinnillä joillain testilaitteilla.

Työn tuloksista voitiin todeta eri laitteiden välillä olevia suorituskykyeroavaisuuksia. Testeissä voitiin havaita optimoinnin tärkeys, jotta mobiilipeli toimisi sulavasti laajemmalla valikoimalla mobiililaitteita. Työssä käytetyt testilaitteet käyttivät Adreno-grafiikkaprosessoria. Testejä ei saatu suoritettua eri grafiikkaprosessoreita käyttävillä testilaitteilla vastaan tulleiden ongelmien vuoksi, joita ei ollut ajan puitteissa mahdollista selvittää tähän opinnäytetyöhön.

KAMK • University of Applied Sciences

Abstract

Author(s): Alavuotunki Maiju

Title of the Publication: Unreal Engine and mobile game development

Degree Title: Bachelor of Engineering, Information and Communication Technologies

Keywords: unreal engine, mobile games, profiling, optimization, video game

The subject of this thesis was to research the profiling of games developed using the Unreal Engine 5 game engine. Critical Force was the commissioner for this thesis. Critical Force is a game studio from Kajaani founded in 2012. Critical Ops is a mobile game developed by Critical Force made with Unity game engine. The commissioner was interested in how different functions have an influence on the performance and the difference between various mobile devices.

The material used for this thesis was Unreal Engine's official documentation, webinars and other material found on the internet. The theorical part of the thesis introduces mobile games and Unreal Engine 5 game engine as well as some general mobile game development specific remarks and optimization recommendations. The practical part of the thesis was accomplished by building different types of tests and profiling them to inspect the performance and the frame time. The result of the profiled data showed the performance difference between various devices as well as how different functions affect the frame time.

The results obtained in the tests showed the effect of different functions on screen time. When adding objects, the time of the spawn function increased evenly on the test devices. In the triangles and draw call tests, it was verified that excessive number of triangles and draw calls significantly slow down the screen time. There were differences between the test devices in the effect of screen time and the number of drawn triangles and draw calls. No significant performance differences were observed in the post-processing effects tests. In the testing of forward and deferred rendering, it was observed that on some test devices deferred rendering had a lower frame time than forward rendering.

The results of the tests showed performance differences between various test devices. The importance of optimization was verified in the tests so that the mobile game would work smoothly on a wider range of mobile devices. The test devices used in the practical part of the thesis used the Adreno graphics processor. Tests could not be performed with test devices using different graphics processors due to the problems encountered, which were not possible to solve in the time frame for this thesis.

Sisällys

Symboliluettelo

CPU	Prosessori
GPU	Grafiikkaprosessori
Ruutuaika	Ilmoitetaan yleensä millisekunteina, kertoo, kuinka kauan ruudun piirtämiseen kuluu aikaa.
FPS	Frames per seconds eli ruutuja sekunnissa, ilmaisee kuinka monta ruutua peli näyttää sekunnissa.
LOD	Level of detail eli yksityiskohtien tarkkuus.
Forward-renderöinti	Renderöi objektien valaistuksen valonlähteiden perusteella. Käsittelee kentän geometrian ja valaistuksen piirtoajassa.
Deferred-renderöinti	Renderöi kentän geometrian ja valaistuksen erillisinä.

1 Johdanto

Mobiilipelimarkkinat ovat nousujohteisia ja useat pelistudiot kehittävät mobiililaitteille koko ajan uusia pelejä. Menestyksekkäät mobiilipelit tarjoavat pelaajilleen sujuvan pelikokemuksen ja niitä voidaan pelata laajalla laitevalikoimalla. Mobiililaitteiden laitteistoissa voi olla huomattavia eroja, minkä takia mobiilipelejä tulisi testata eri mobiililaitteilla ja ne tulisi optimoida. Huonosti optimoitu peli voi pienentää pelaajakuntaa, jos alhaisemman tason laiteiden omistajat eivät saa hyvää pelikokemusta tai pahimmassa tapauksessa eivät pysty pelaamaan peliä lainkaan omalla mobiililaitteellaan.

Opinnäytetyössä keskityttiin Android-laitteisiin eikä iOS-laitteiden eroja otettu huomioon. Opinnäytetyön käytännönosuudessa toteutettiin testit Adreno-grafiikkaprosessoreita käyttävillä laitteilla. Testeiksi valittiin mahdollisia suorituskykyyn vaikuttavia tekijöitä, kuten piirrettyjen kolmioiden määrä sekä usean objektin samanaikainen lisäys kenttään. Testeissä kartoitettiin, kuinka testitapaukset vaikuttavat suorituskykyyn. Samat testit suoritettiin eri laitteilla, jotta voitiin verrata myös laitteiden välisiä suorituskykyeroja.

Tässä opinnäytetyössä kartoitettiin mobiilipelien optimoinnin teoriaa ja keskitytään pelien profilointiin Unreal Engine 5 -pelimoottoria käyttäen sekä selvitettiin eri laitteiden suorituskykyeroja peliprojekteissa. Unreal Engine 5 -pelimoottorilla profiloinnista löytyy jonkin verran tietoa ja dokumentaatiota, mutta huomattavasti vähemmän kuin kilpailevista pelimoottoreista. Mobiilipelien kehittämisestä Unreal Engine 5 -pelimoottorilla löytyy vielä vähäisesti materiaalia internetistä, mistä syystä aihe kiinnostaa mobiilipelien kehittäjiä.

Opinnäytetyö tehtiin Critical Forcen toimeksiantona. Critical Force on kajaanilainen pelistudio, joka on kehittänyt Critical Ops -mobiilipelin Unity-pelimoottorilla. Toimeksiantajaa kiinnostaa Unreal Enginen tarjoamat mahdollisuudet mobiilipelien kehittämisessä sekä eri laitteiden eroavaisuudet suorituskyvyssä.

Opinnäytetyön tavoitteena on nostaa esiin niitä haasteita, joita käyttäjien laitevalikoima asettaa pelaamisen sujumiselle ja pelaamiskokemukselle. Käytännön osuuden testien avulla halutaan selvittää, miten kyseiset toiminnot vaikuttavat suorituskykyyn sekä millaisia ovat projektin suorituskykyerot eri laitteilla.

2 Mobiilipelit

Älylaitteiden tullessa edelleen yleisemmäksi ja mobiilipelaamisen suosion kasvaessa mobiilipelit ovat yksi suuri pelaamisen muoto. Mobiilipelejä on saatavilla jo tuhansia ja maailmanlaajuisesti mobiilisovellusten latausmäärät ovat olleet nousussa viime vuosina. Vuoden 2023 heinäkuussa mobiilipelituotot olivat melkein puolet globaaleista pelimarkkinoista vuosikasvun ollessa +0,8 %. [1.] Mobiilipelien suosiota selittää osin älypuhelimien laaja saatavuus. Älypuhelin löytyy nykypäivänä yli puolelta maailman väestöstä [2] ja mahdollinen pelaajakunta on näin ollen laajempi kuin tietokone- tai konsolipeleillä. Mobiilipelit ovat helposti ladattavissa laitteelle ja niiden pelaaminen on kätevää, vaikka aikaa olisi vain muutamia minuutteja. Monet mobiilipeleistä ovat ilmaiseksi pelattavissa, mikä alentaa pelaajan kynnystä hankkia pelejä.

Monet mobiilipelit sisältävät vuodenajan tai kausittaisten juhlapyhien teemaisia live-tapahtumia, jotka pitävät pelaajan mielenkiinnon yllä ja saavat heidät palaamaan päivittäin pelin pariin. Mobiilipelit usein sisältävät myös sosiaalisia toimintoja, kuten tulostaulun tai niin kutsuttujen lisäelämien pyytämisen pelikavereilta.

Teknologian kehittyessä mobiililaitteiden laitteistot paranevat ja niille voidaan rakentaa monimutkaisempia ja upeamman näköisiä pelejä. Uudet teknologiat mahdollistavat myös uusien ja erilaisten pelityyppien syntymistä. Mobiililaitteilla voidaan käyttää jo hyödyksi pelaajan sijaintia, puhelimen kameraa sekä laitteen kallistusta erilaisen pelikokemuksen saavuttamiseksi.

2.1 Mobiilipelien historia

Ensimmäiset mobiilipelit olivat grafiikoiltaan ja pelimekaanikoiltaan yksinkertaisia. Tetris oli ensimmäinen mobiilipeli, kun se julkaistiin vuonna 1994 Hagenuk MT-2000 -laitteelle [3]. Mobiilipelit alkoivat saada tuulta allensa vuonna 1997, kun Nokia julkaisi Snake-pelin valmiiksi asennettuna Nokia 6110 -puhelimelle. Muutamaa vuotta myöhemmin Nokia lisäsi Space Impact - ja Bounce-pelit puhelimiinsa. [4.]

Mobiilipelit muuttuivat taas, kun kosketusnäytöt alkoivat yleistyä mobiililaitteissa. Myös puhelimien suorituskyky parani laitteiston kehittyessä. Vuonna 2007 julkaistu ensimmäinen iPhone oli kehittyneempi kuin muut markkinoilla olevat laitteet. AppStoren julkaisun myötä kehittäjät saivat alustan, jossa julkaista ja myydä tai jakaa ilmaiseksi omia pelejään. Vuonna 2009 kauppa-alusta

lisäsi mahdollisuudet pelinsisäisiin ostoihin. Vuonna 2023 suurimmat sovelluskaupat ovat Googlen Play Store ja Applen App Store [5].

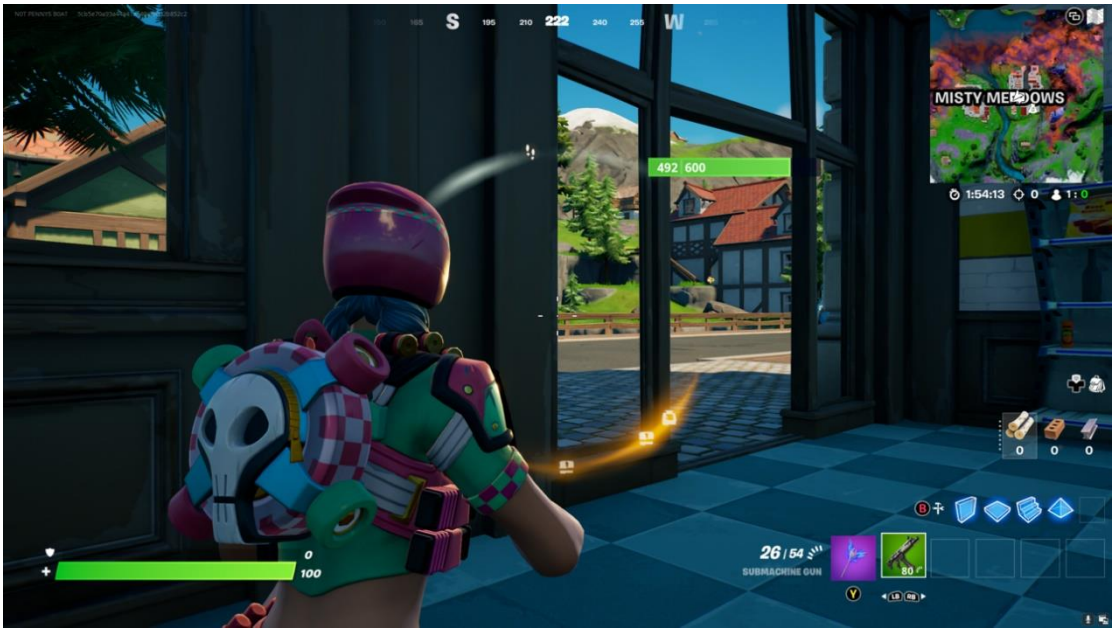
Sosiaalinen pelaaminen kasvoi Facebookin myötä. Vuonna 2007 Facebook antoi kolmansien osapuolien kehittää ohjelmia sekä pelejä alustalleen. Zynga julkaisi vuonna 2009 Farmville-pelin, joka oli Facebookin ensimmäinen hittipeli. Helmikuussa 2010 Zyngan Farmville-pelillä oli yli 80 miljoonaa pelaajaa. [4.] Seuraava Facebook-pelien hitti oli Candy Crush Saga, jonka julkaisi King. Candy Crush Saga julkaistiin myös sovelluskaupoissa ja se oli ladatuin ilmainen peli App Storessa vuonna 2013. [6.]

Mobiilipeleissä alkoi lisääntyä lisätty todellisuus eli AR Pokémon Go:n myötä vuonna 2016. Peli nousi heti suureen suosioon ja siitä kasvoi kulttuurillinen ilmiö. Pokémon Go vaikutti peliteollisuuteen popularisoimalla sijaintiin perustuvan pelaamisen. [7.]

2.2 Huomioitavaa mobiilipelin kehityksessä

Mobiililaitteissa on niiden koon puolesta rajoituksia muistissa ja suorituskyvyssä. Teknologia on kuitenkin mennyt harppauksia eteenpäin parin viime vuosikymmenen aikana, ja mobiililaitteille kehitetyt sovellukset voivatkin olla jo melko monimutkaisia. Mobiililaitteissa näytön pieni koko alentaa tarvittavaa grafiikan yksityiskohtaisuutta, mutta sitä voidaan hyödyntää pelinkehityksessä, sillä peligrafiikan ei tarvitse olla yhtä tarkkaa kuin tietokone- tai konsolipeleissä. Huonosti optimoitu peli voi kuluttaa huomattavan määrän laitteen virtaa tai toimia hitaasti ja tökkivästi. Mobiilipelien kehityksen jokaisessa vaiheessa tulisi huomioida mahdollinen optimointi ja projektia tulisi säännöllisesti testata kohdelaitteella, jotta ei tulisi kehitetyksi ohjelmaa, jota mobiililaite ei jaksa pyörittää.

Mobiililaitteella pelatessaan pelaajat eivät välttämättä aina pidä peliääniä päällä. Tämä olisi hyvä ottaa huomioon kokeilemalla, onko peliin tarpeellista lisätä visuaalisia efektejä, joiden avulla pelaajan olisi helpompi ymmärtää pelin tapahtumia myös ilman peliääniä. Esimerkiksi useisiin taistelupeleihin on lisätty nuoli, joka indikoi, mistä suunnasta pelaajaan yritetään tehdä vahinkoa. Unreal Engine -pelimoottorilla tehtyyn Fornite-peliin on lisätty visuaaliset efektit äänille, kuten askelille ja laatikoiden avaamiselle. Äänien visualisointi pitää laittaa päälle pelin asetuksista. [8.] Kuvassa 1 on nähtävissä kyseiset visualisoinnit pelissä.



Kuva 1. Fortnite-pelissä on mahdollista laittaa päälle äänet visualisoiva efekti. [8.]

Kosketuseleet ovat mobiililaitteiden pääsyötetapa. Kosketussyöte ottaa vastaan useita eri eleitä, kuten napautus, tuplanapautus sekä pyyhkäisy. Mobiililaitteet pystyvät vastaanottamaan tietyn määrän kosketussyötteitä yhtäaikaisesti. Näiden eleiden ohjelmointi tulisi tehdä huolellisesti, jotta eri eleet eivät sekoittuisi keskenään. Esimerkiksi sivulle pyyhkäistessä syötettä ei pitäisi lukea ylöspäin pyyhkäisyksi, vaikka liike olisikin lievästi yläviistoon, ellei tätä nimenomaan haluta.

Älypuhelimilla näyttökoko on rajoitettu, jolloin ohjausnäppäimet saattavat olla hyvin lähekkäin. Mobiililaite ei anna palautetta napautuksesta, joten jos napin painamisen ja toiminnon toteuttamisen välillä on viivettä, pelaaja ei välttämättä tiedä, painoiko hän oikeaa painiketta tai rekisteröityikö kosketus ollenkaan. [9.]

Mobiilipeliä kehittäessä tulisi kiinnittää huomiota käyttöliittymän suunnitteluun. Mobiililaitteiden näytön koko voi vaihdella melko laajasti, jolloin pelin skaalautuvuus eri resoluutiolle on yksi tärkeä osa kehityksessä. Käyttöliittymän tulisi toimia erilaisilla näytöillä ja kehittäjien tulisi varmistaa, etteivät painikkeet jää näytön reunan ulkopuolelle ja etteivät ne mene päällekkäin.

3 Mobiilipelien suorituskyvyn parantaminen

Pelin suorituskyky vaikuttaa siihen, millaisen pelikokemuksen pelaaja saa. Jotta peli näyttäisi sujuvalta, täytyy sen pystyä näyttämään 30 ruutua sekunnissa. Kuvataajuuteen vaikuttavat käyttäjän osalta käytetty laite sekä käytetyt peliasetukset ja kehittäjien osalta pelin koodin sekä assettien eli pelin käyttämien tiedostojen, kuten 3D-mallien, optimointi. Verkkopeleissä halutaan varmistaa, että peli ei aiheuta viivettä samassa pelissä olevien pelaajien välillä. Varsinkin kilpailullisissa PvP (player versus player) eli pelaaja vastaan pelaaja -peleissä on tärkeää pitää viive pienenä, jotta toinen pelaaja ei saisi epäreilua etua.

Pelin tiedostokokoon tulisi kiinnittää huomiota kehitysvaiheessa. Eräässä tutkimuksessa tehtiin käyttäjäkysely, jonka perusteella todettiin ongelmakohdiksi pelin suuri koko, pelin asennusaika sekä kenttien välinen pitkä latausaika [10]. Tällaiset ongelmat voivat olla huonon optimoinnin merkkejä ja ne vaikuttavat negatiivisesti pelaajan saamaan kokemukseen. Pelin tiedostokoko ja muistin hallinta vaikuttavat pelistä saatuun kokemukseen, joten näitä resursseja tulisi käyttää tehokkaasti.

3.1 Mobiilipelien optimointi

Optimoinnilla tarkoitetaan pelin parantamista korjaamalla ongelmakohdat, jotka kuluttavat paljon laitteen resursseja. Optimoinnin tavoitteena on käyttää saatavilla olevia resursseja tehokkaasti, jolloin pelin suorituskyky pysyy hyvänä. Optimoinnin avulla voidaan saavuttaa nopeampi pelin kuvataajuus ja vähentää latausaikoja, jolloin pelikokemus tuntuu sujuvammalta. [11.] Pelaajat usein seuraavat ruutujen määrää sekunnissa eli FPS-lukemaa, kun kehittäjien tulisi seurata kuinka kauan ruudun piirtämiseen kuluu aikaa millisekunteina eli ruutuaikaa. Taulukko 1 havainnollistaa, kuinka ruutuaika nousee tasaisesti viisi millisekuntia, mutta FPS-lukema ei. Kun ruutuaika kasvaa 21 millisekunnista 26 millisekuntiin, FPS-lukemien ero on noin 9 FPS, kun taas ruutuajan noustessa 11 millisekunnista 16 millisekuntiin, FPS-lukemien ero on noin 28 FPS. Ruutuaika nousi molemmissa tapauksissa 5 millisekuntia, mutta FPS-lukemissa on suuri ero. Tästä syystä pelin kehittämisen sekä sen profiloinnin kannalta tulisi tarkastella ruutuaikaa ja sen muutoksia.

Taulukko 1. Tasainen viiden millisekunnin kasvu ruutuajassa ei muuta FPS-lukemaa tasaisesti.

Ruutuaika (ms)	FPS
11	90,91
16	62,50
21	47,62
26	38,46

Mobiililaitteille julkaistavat pelit ovat ensisijaisen tärkeä optimoida, sillä huonosti optimoitu peli voi kuumentaa puhelimen sekä kuluttaa paljon laitteen akkua. Hyvin optimoitua peliä voidaan pelata laajemmalla valikoimalla eritasoisia laitteita ja saada pelaajille hyvä pelikokemus.

Jo pelin alkuvaiheessa on hyvä miettiä, mille alustoille peli julkaistaan sekä esimerkiksi millainen visuaalinen tyyli pelissä on. Kehittämisen aikana jokaisen tiimin jäsenen tulisi miettiä, miten optimoida peliin lisättyä omaa tuotostaan. Optimointia tulisi tehdä aina tarvittaessa, eli muun muassa silloin, kun jokin toiminto kuluttaa suorituskykyä huomattavasti. [12.]

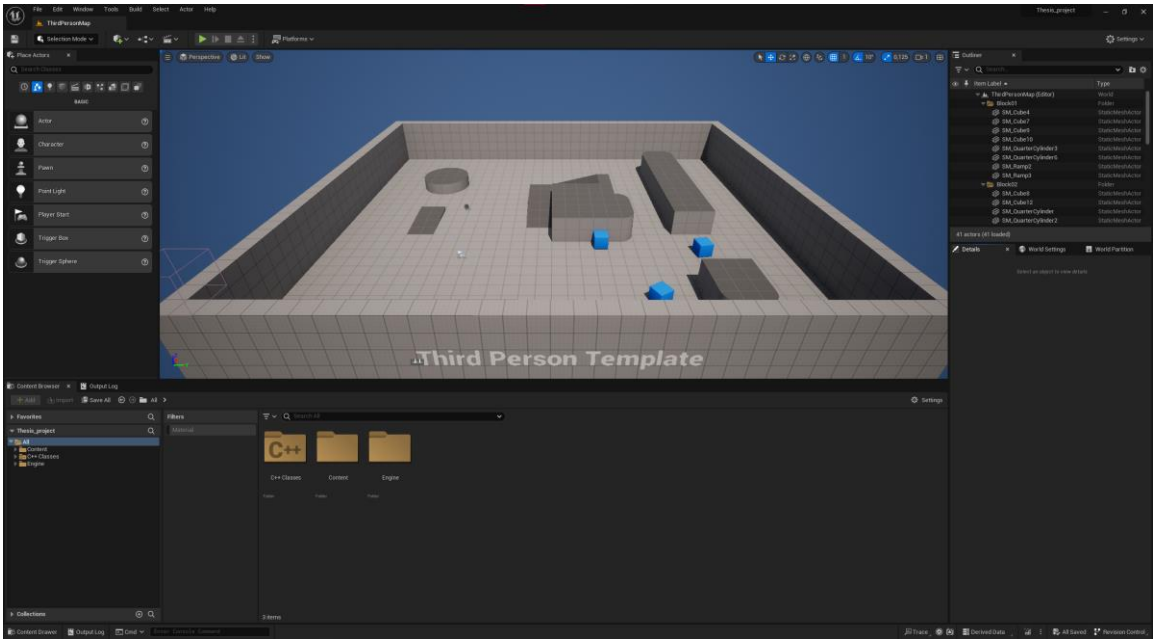
3.2 Mobiilipelien profilointi

Jotta peli voidaan optimoida, pitää tietää, mitkä ovat suorituskykyä alentavia kohtia. Profilointia tehdään näiden ongelmakohtien havaitsemiseksi sekä löytämiseksi. Profilointi on tärkeä osa kehitystä, sillä kaikkia ongelmakohtia ei välttämättä löydetä ennen kuin niitä on useita, jolloin joudutaan käyttämään paljon aikaa niiden korjaamiseen. Kun profilointia tehdään jatkuvasti kehityksen edetessä, ongelmakohdat voidaan korjata heti. [12.] Profilointi kannattaa suorittaa kohdelaitteella ja pakattuna versiona, jotta profiloitu data olisi mahdollisimman puhdasta.

Profiloinnin avulla voidaan selvittää, kuinka kauan jonkin asian suorittaminen kestää tai kuinka monta kertaa jokin asia suoritetaan. Profiloinnilla voidaan myös seurata resurssien käyttöä, kuten muistin ja verkon käyttöä. Profiloinnin avulla voidaan tarkastella yksityiskohtaisemmin, mitkä toiminnot kuluttavat prosessoria ja grafiikkaprosessoria eli CPU:ta ja GPU:ta.

4 Unreal Engine

Unreal Engine 5, kuva 2, on Epic Gamesin kehittämä 3D-luontityökalu ja se julkaistiin alkuvuodesta 2022. Unreal Engine on ilmaiseksi ladattavissa Epic Games launcherin kautta. Useat pelistudiot käyttävät Unreal Engine -moottoria ja monet viime vuosina julkaistut isot nimikkeet ovat tehty kyseisellä moottorilla. Unreal Enginen monipuolinen työkaluvalikoima mahdollistaa moottorin käyttämisen pelien kehittämisen lisäksi myös esimerkiksi viihdeteollisuudessa, kuten elokuvissa, arkkitehtuurissa ja live-tapahtumissa. [13.]



Kuva 2. Unreal Engine 5 käyttöliittymä.

Unreal Engine sopii myös mobiilipelien kehitykseen, sillä pelimoottorilla voi luoda pelejä kaikille pelialustoille. Unreal Enginellä voidaan luoda niin 2D- kuin 3D-pelejä. Unreal Engine on helposti käytettävissä ja kehittäjät voivat käyttää moottoria ilmaiseksi, kunnes sillä tehdyllä pelillä tienataan miljoonan dollarin tulot, jolloin yli menevästä osuudesta maksetaan 5 % rojalteja [14]. Käyttäjällä on myös pääsy moottorin lähdekoodiin, joten itse moottoria voi tarvittaessa muokata juuri omaan tarpeeseen.

Unreal Engine on tehokas työkalu ja se tarjoaa käyttäjilleen useita eri toimintoja pelien kehittämiseen. Jotkin moottorin toiminnoista vaativat laitteilta paljon suorituskykyä, minkä vuoksi moottoriin integroidut profilointityökalut ovat moottoria käyttäville ensisijaisen tärkeitä sujuvan ja tehokkaan lopputuloksen saamiseksi.

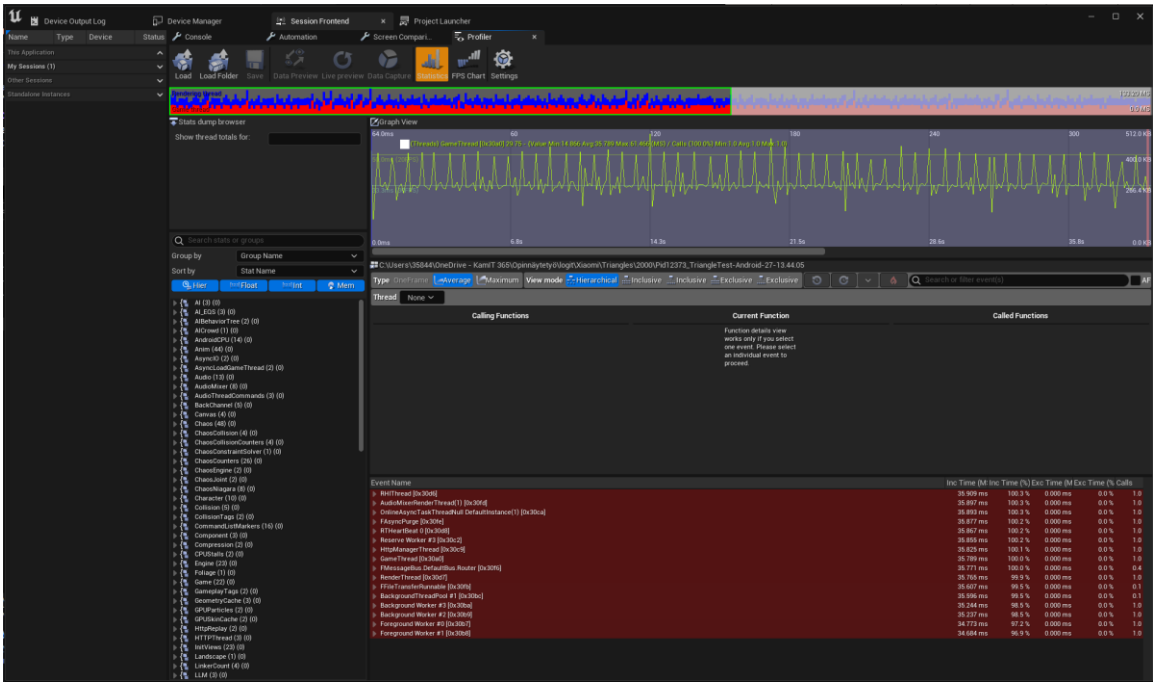
4.1 Profilointityökalut

Pelien profilointiin on tarjolla useita erilaisia työkaluja eri valmistajilta. Unreal Engine sisältää siihen integroituja profilointityökaluja, joiden avulla käyttäjä voi helposti havainnoida omaa sisältöään tai koodiaan sekä löytää siitä mahdolliset suorituskykyä kuluttavat ongelmakohdat.

Tässä osassa käydään läpi opinnäytetyössä eniten käytettyjä Unreal Enginen sisäisiä profilointityökaluja. Kolmannen osapuolen profilointityökaluja ei opinnäytetyötä tehdessä käytetty.

4.1.1 Unreal Frontend

Unreal Frontend -työkalulla voidaan analysoida yksityiskohtaisesti, mitkä toiminnot vaikuttavat suorituskykyyn ja kuinka kauan toiminnon suorittaminen kestää. Frontend Profiler -käyttöliittymä on nähtävillä kuvassa 3. Työkalun avulla voidaan CPU- ja GPU-profiloinnin lisäksi kartoittaa muistin käyttöä.



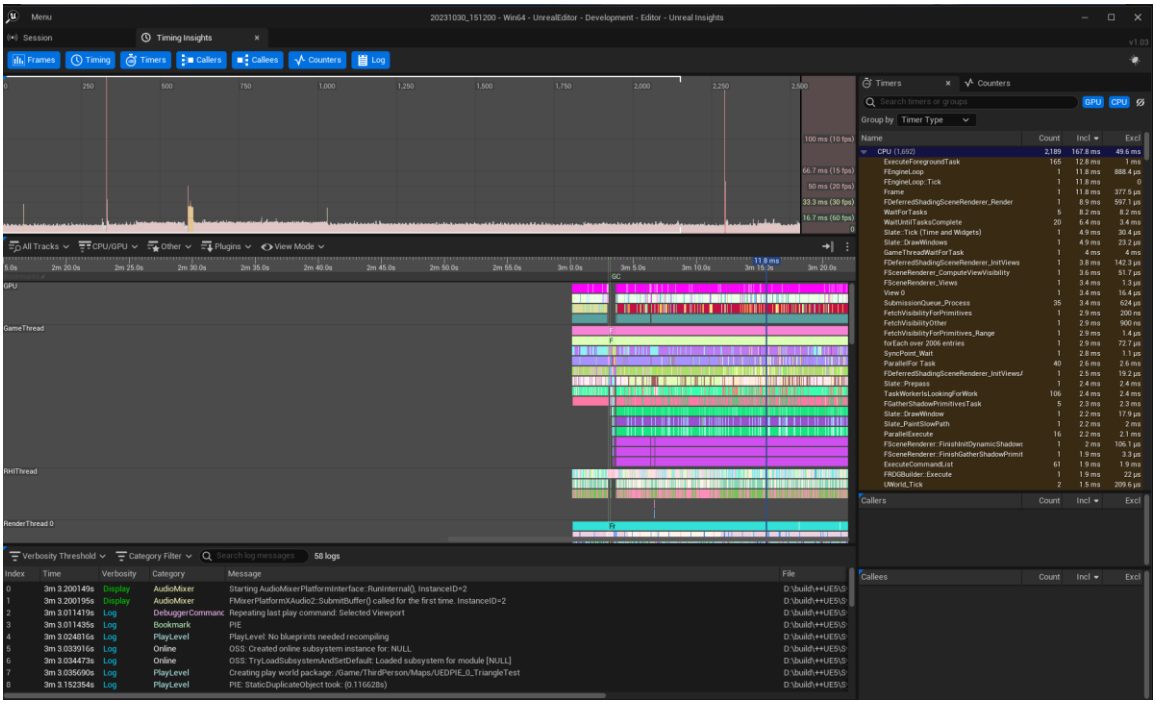
Kuva 3. Unreal Frontend -käyttöliittymä.

Frontend on työkalu, jonka avulla kehittäjät voivat tarkastella pelinsä kaikkia verkossa käynnissä olevia sessioita eli erillisiä peli-instansseja. Työkalun avulla voidaan analysoida peliä etänä ja paikantaa eri instansseissa esiintyviä suorituskykyä alentavia kohtia. Konsoliin näytetään jokaisen

instanssin tietoja ja niille voidaan syöttää konsolikomentoja. Datan kerääminen mobiililaitteella voidaan aloittaa kirjoittamalla konsolikomento stat StartFile. [15.]

4.1.2 Unreal Insights

Unreal Insights on erillinen profilointityökalu, joka lisättiin moottorin 4.23-versioon, jonka avulla voidaan analysoida ja jäljittää tietoa. Työkalun avulla voidaan kerätä sekä visualisoida moottorin lähettämää dataa hyvinkin tarkasti. Työkalu näyttää hierarkkisesti CPU- ja GPU-suorittimien tapahtumat sekä siitä voidaan lukea lokitulosteita ja tarkastella kuvataajuutta. [16.] Unreal Insights -työkalun käyttöliittymä näkyy kuvassa 4.



Kuva 4. Unreal Insights -käyttöliittymä.

Unreal Insightin avulla voidaan tutkia myös tarkemmin esimerkiksi muistin ja netin käyttöä. Tarkasteluun voi valita aikaisemmin tallennetun tiedoston tai sitä voi käyttää reaaliajassa.

4.2 Unreal Enginen suosittelemat asetukset pelin kehitykseen

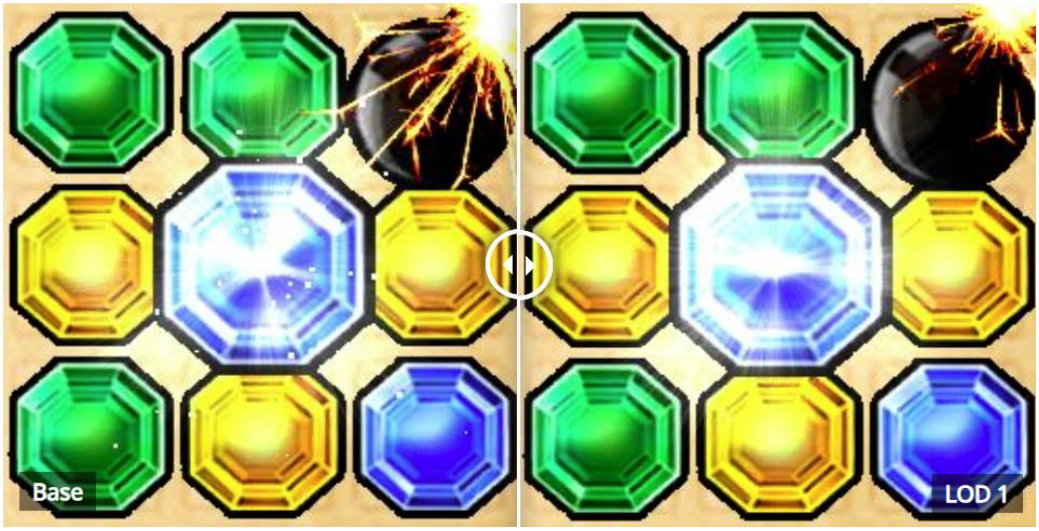
Unreal Engine -moottorin dokumentaatiosta löytyy suosituksia, joilla voidaan parantaa pelin suorituskykyä. Jotkin suosituksista on projektin asetuksista laitettavia asetuksia ja osa suosituksista on vinkkejä pelin optimointiin. Unreal Engine YouTube-kanavalta löytyy erilaisia webinaareja, joista löytyy vinkkejä pelin suorituskyvyn parantamiseksi.

4.2.1 Yleisiä suosituksia

Usein kun puhutaan Unreal Engine -moottorilla tehtyjen projektien optimoinnista, tulee esille tick-funktio. Tick-funktio on oletusarvoisesti joka ruudulla suoritettava funktio, joten on hyödyllistä miettiä, tarvitseeko kyseisen objektin logiikkaa suorittaa joka ruudulla. Usean objektin käyttäessä tick-funktiota, peli saattaa hidastua huomattavasti. [17.] Tämän vuoksi olisi kannattavaa miettiä, onko mahdollista suorittaa logiikka vain jonkin tapahtuman sattuessa tai käyttämällä ajastinta. Tick-funktion aikaa, jolloin se suoritetaan, voi myös muuttaa, jolloin funktiokutsuja saadaan vähennettyä.

Pelilogiikka voi toisinaan olla hyvinkin monimutkaista ja käyttää kalliita matemaattisia laskuja sekä fysiikkatoimintoja. Näiden toiminta olisi hyvä toteuttaa C++-koodissa, sillä C++-logiikka on blueprinttejä eli Unreal Enginen visuaalista skriptausmenetelmää huomattavasti nopeampi. C++ tarjoaa monimutkaisten järjestelmien suunnitteluun ja toteuttamiseen enemmän mahdollisuuksia kuin blueprintit. [17.] Unreal Enginen ydinominaisuuksien muokkaaminen ja joidenkin algoritmien käyttö voi olla C++:n avulla helpompaa.

Yksityiskohtien tasot eli LOD:t (level of detail) kannattaa ottaa käyttöön jokaiseen objektiin, jolloin piirrettävien kolmioiden määrää saadaan laskettua. Näin ollen ei käytetä suorituskykyä turhaan kaukana pelaajasta olevien objektien yksityiskohtien piirtämiseen. LOD:ja voi käyttää myös eri tavalla hyödyksi kolmioiden määrän vähentämisen sijaan. Unreal Enginen Unreal Match 3 -esimerkkiprojektissa, yksityiskohtien tasoja käytettiin partikkeliefekteihin siten, että alemman tason laitteilla partikkelien määrä puolittui [18]. Unreal Match 3 -projektin LOD-profiilien efekti on nähtävillä kuvassa 5. Kuvassa vasemmalla on alkuperäinen efekti ja oikealla näkyy LOD 1 -profiilin vaikutus partikkelien määrään.



Kuva 5. Unreal Match 3 -esimerkkiprojektin partikkeliefektit. [18.]

4.2.2 Mobiilipeleille ominaiset suositukset

Tässä osassa käydään läpi joitain mobiilipeleille ominaisia suosituksia Unreal Engine 5 -pelimoottorilla. Unreal Enginen dokumentaatiosta löytyy mobiilipelikohtaisia suosituksia, joiden avulla voidaan parantaa pelin suorituskykyä. Useimmat näistä suosituksista ovat asetuksia, jotka on hyvä ottaa käyttöön projektissa.

Eritasoisille laitteille voidaan määrittää asetuksia, joita peli käyttää laitteella, kuten materiaalien laadun taso ja ruudun resoluutio. Alemman tason laitteille voidaan määrittää laatuasetukset oletuksena alemmaksi. Tällä tavoin peli voi toimia lähtökohtaisesti paremmin käyttäjän laitteella. Kuvassa 6 on Cropout-esimerkkiprojektin DefaultDeviceProfiles.ini-tiedoston kohta, jossa on määritelty alemman tason Android-laitteiden oletusasetukset. Tiedosto löytyy projektin config-kansiosta.

```
[Android_Low DeviceProfile]
DeviceType=Android
BaseProfileName=Android
+CVars=r.MobileContentScaleFactor=0.8
; Scalability groups, see AndroidScalability.ini
+CVars=sg.ViewDistanceQuality=0
+CVars=sg.AntiAliasingQuality=0
+CVars=sg.ShadowQuality=0
+CVars=sg.GlobalIlluminationQuality=0
+CVars=sg.ReflectionQuality=0
+CVars=sg.PostProcessQuality=0
+CVars=sg.TextureQuality=0
+CVars=sg.EffectsQuality=0
+CVars=sg.FoliageQuality=0
```

Kuva 6. Cropout-esimerkkiprojektin DefaultDeviceProfiles.ini-tiedosto.

Riippuen kehitettävästä pelistä ja sen tarpeista, Mobile HDR (High-Definition Rendering) on mahdollista ottaa pois käytöstä. Jos Mobile HDR-asetus on poistettu käytöstä, jälkikäsittelyefektit ja valaistukset eivät toimi. Unreal Engine tukee eritasoisia valaistustapoja mobiililaitteelle, jotka vaativat Mobile HDR -asetuksen olevan päällä. [19.] Mobile HDR -asetus löytyy projektin asetuksista renderöintiasetuksista.

Unreal Enginen dokumentaatiossa mainitaan, että mobiilipelien maksimi kolmiomäärä tulisi olla maksimissaan 500 tuhatta mihin tahansa kentässä katsottuna. Dokumentaatiossa mainitaan myös, että piirtokutsuja tulisi olla enintään 700 mihin tahansa kentässä katsottuna. [19.]

5 Testikentän rakentaminen ja profilointi

Opinnäytetyössä ei ollut käytössä valmista peliprojektia, joten testit toteutettiin kolmannen persoonan valmista projektipohjaa käyttämällä. Testit toteutettiin Android-laitteilla. Käytettävät laitteet olivat Xiaomi Mi 9T Pro, POCO X4 Pro 5G ja OnePlus Nord. Kaikissa laitteissa on Snapdragon-suoritin sekä Adreno-grafiikkaprosessori. Taulukkoon 2 on lueteltuna käytössä olleet testilaitteet sekä niiden prosessori ja grafiikkaprosessori. Tarkoitus oli tehdä testit myös eri grafiikkaprosessoreita käyttävillä mobiililaitteilla. Testilaitteiden kanssa vastaan tulleiden ongelmien vuoksi näitä testejä ei voitu suoritettua.

Taulukko 2. Käytetyt laitteet sekä niiden prosessorit ja grafiikkasuorittimet. [20.] [21.] [22.]

Laite	Prosessori	Grafiikkasuoritin
Xiaomi Mi 9T Pro	Snapdragon 855	Adreno 640
OnePlus Nord	Snapdragon 765G	Adreno 620
POCO X4 Pro 5G	Snapdragon 695	Adreno 619

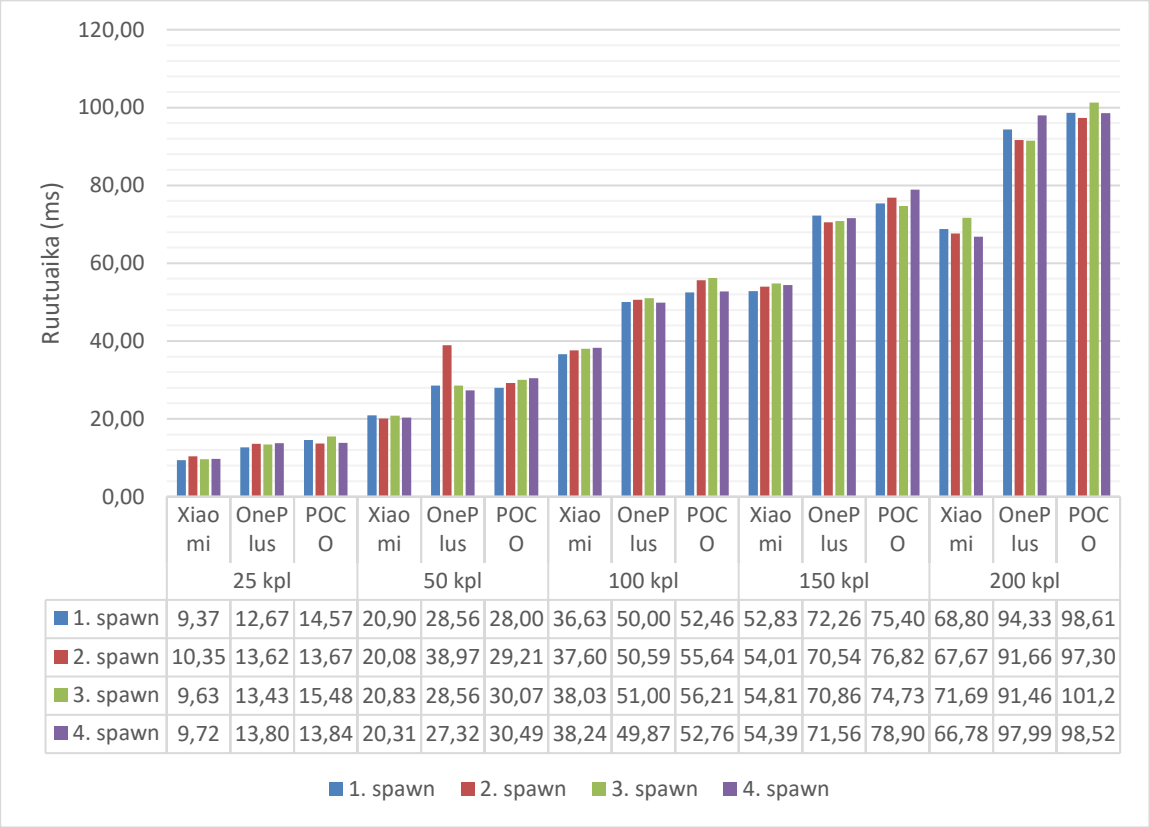
Testeiksi valittiin kolmioiden määrän, piirtokutsujen, jälkikäsittelyefektien sekä forward -ja deferred-renderöintien vaikutus ruutuaikaan. Lisäksi testattiin objektien lisäämistä eri määrillä ja tutkittiin spawn-funktion kestoa objekteja lisättäessä. Näillä testeillä voidaan todentaa eri laitteiden suorituskykyeroja sekä tehdä havaintoja, miten yleisesti käytössä olevat toiminnot vaikuttavat pelin suorituskykyyn. Kolmioiden määrä ja piirtokutsut ovat yleisiä optimointikohteita, jonka vuoksi ne valittiin opinnäytetyöhön testeiksi. Jälkikäsittelyefektejä käytetään useissa peleissä, jonka vuoksi haluttiin kartoittaa jälkikäsittelyefektien vaikutusta ruutuaikaan ja suorituskykyeroja eri laitteilla. Eri renderöintitapojen suorituskykyeroja kartoitettiin mielenkiinnon vuoksi.

5.1 Objektien lisääminen spawn-funktiolla

Objektien lisäämistesti toteutettiin kirjoittamalla koodi, joka lisäsi eli spawnasi tietyn määrän objekteja kahden sekunnin välein neljä kertaa. Tämän jälkeen testeistä kerätty data käytiin läpi ja niistä laskettiin keskiarvo. Testeissä lisättävät määrät olivat 25, 50, 100, 150 ja 200 kappaletta objekteja. Jokaiselle määrälle testit tehtiin neljä kertaa paremman keskiarvon saamiseksi. Tauluk-

koon 3 on kerätty testikertojen keskiarvot niin, että testien ensimmäisistä spawnauksista laskettiin keskiarvo, toisista spawnauksista laskettiin keskiarvo, ja niin edelleen. Kaikkien testien tarkemmat tulokset ovat nähtävillä liitteessä 1.

Taulukko 3. Objektien lisäämistestien tulokset.



	Xiaomi	OnePlus	POCO	Xiaomi	OnePlus	POCO	Xiaomi	OnePlus	POCO	Xiaomi	OnePlus	POCO	Xiaomi	OnePlus	POCO
	25 kpl			50 kpl			100 kpl			150 kpl			200 kpl		
1. spawn	9,37	12,67	14,57	20,90	28,56	28,00	36,63	50,00	52,46	52,83	72,26	75,40	68,80	94,33	98,61
2. spawn	10,35	13,62	13,67	20,08	38,97	29,21	37,60	50,59	55,64	54,01	70,54	76,82	67,67	91,66	97,30
3. spawn	9,63	13,43	15,48	20,83	28,56	30,07	38,03	51,00	56,21	54,81	70,86	74,73	71,69	91,46	101,2
4. spawn	9,72	13,80	13,84	20,31	27,32	30,49	38,24	49,87	52,76	54,39	71,56	78,90	66,78	97,99	98,52

Saadut arvot ovat spawn-funktion aikoja. Testeissä parhaiten suoriutui Xiaomi 9T Pro, jonka spawn-funktion ajat olivat alhaisimmat. Xiaomilla tehdyissä testeissä spawnausaika lisääntyi objektien lisääntyessä keskiarvollisesti 14,8 millisekuntia ja OnePlussalla keskiarvollisesti 20,4 millisekuntia. POCO-laitteella spawnausaika lisääntyi keskiarvollisesti 21,1 millisekuntia. Xiaomi-laiteella viimeinen testi suoriutui yli 25 millisekuntia nopeammin kuin OnePlus- tai POCO-laitteella.

5.2 Kolmioiden määrä ruudulla ja yksityiskohtien määrä

Testi toteutettiin lisäämällä objekteja karttaan ja profiloimalla siitä kerättyä dataa. Objekteina käytettiin aloituspakkauksesta löytyvää patsasobjektia, josta otettiin yksityiskohtien tasot eli LOD:t pois käytöstä. Testiä varten tyhjään karttaan lisättiin aluksi viisisataa objektia. Jokaisen testin jälkeen karttaan lisättiin 250 objektia ja profilointi suoritettiin uudelleen. Jokaisesta testistä

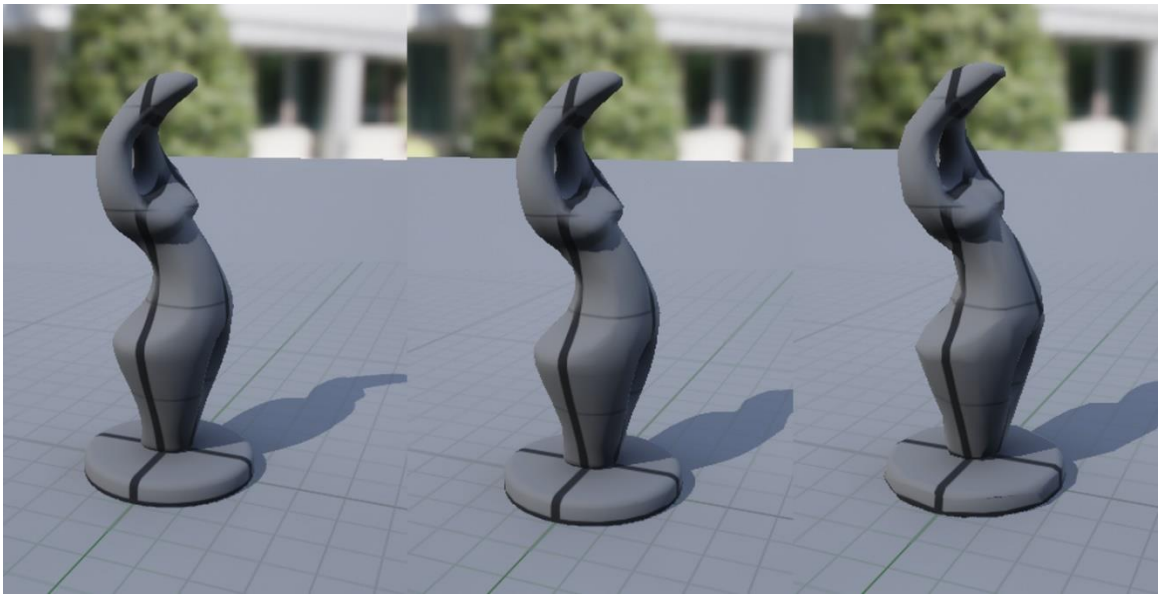
kerätystä datasta otettiin keskiarvoinen ruutuaika sekä piirretyt kolmiot. Saadut arvot näkyvät taulukossa 4. Taulukossa näkyy vasemmalla, kuinka monta objektia testikentällä oli. Mitattuja arvoja olivat ruutuaika millisekunteina sekä piirrettyjen kolmioiden määrä. Pelissä ei ole pelilogiikkaa tai muuta toimintaa, joka kuluttaa suorituskykyä merkittävästi. Testeissä mitatuissa tuloksissa on siis vain kolmioiden määrä vaikuttamassa suorituskykyyn.

Taulukko 4. Kolmioiden määrä -testistä saadut tulokset.

Objektien määrä	Mitattu arvo	Xiaomi Mi 9T Pro	OnePlus Nord	POCO X4 Pro 5G
500 kpl	Ruutuaika (ms)	16,485	20,115	22,632
	Kolmiot	2 051 681,43	1 782 330,99	2 047 926,88
750 kpl	Ruutuaika (ms)	16,452	24,724	26,702
	Kolmiot	3 059 447,99	2 489 189,50	3 059 453,59
1000 kpl	Ruutuaika (ms)	20,128	29,698	31,532
	Kolmiot	4 031 689,23	3 372 477,31	4 064 369,38
1250 kpl	Ruutuaika (ms)	24,449	33,684	40,371
	Kolmiot	4 792 986,16	4 683 413,48	5 035 297,2
1500 kpl	Ruutuaika (ms)	27,26	39,782	44,873
	Kolmiot	5 634 727,87	5 077 866,72	6 041 356,67
1750 kpl	Ruutuaika (ms)	33,566	43,275	49,324
	Kolmiot	6 984 059,08	5 961 275,42	7 047 549,72
2000 kpl	Ruutuaika (ms)	43,108	47,568	50,911
	Kolmiot	8 036 783,42	7 131 653,51	8 032 369,34

Testeissä parhaiten pärjäsi Xiaomi Mi 9T Pro. OnePlus Nord piirsi vähemmän kolmioita, kuin Xiaomi, mutta suoriutui silti heikommin testeissä. POCO piirsi testeissä lähes yhtä paljon kolmioita kuin Xiaomi tai enemmän. POCOn ruutuaika oli kaikissa testeissä suurin. Kolmioiden lisääntyminen ruudulla ei muuttanut tasaisesti keskiarvollista ruutuaikaa.

LOD-profiilien käyttöönottoa suositellaan Unreal Enginen dokumentaatiossa ja tästä syystä tehtiin vielä yksi testi niin, että patsasobjektille lisättiin LOD:t käyttöön. Kuvassa 7 näkyy testeissä käytetyn patsasobjektin alkuperäinen versio, eli LOD-profiili 0, vasemmassa reunassa. LOD-profiili 1 on kuvassa keskellä ja sen kolmiomäärä on 50 prosenttia alkuperäisestä. LOD-profiili 2, jonka kolmiomäärä on 15 prosenttia alkuperäisestä, on kuvassa oikeassa reunassa. LOD-profiili määräytyy sen mukaan, kuinka monta prosenttia ruudusta objekti on. Nämä luvut voidaan määrittää objektin LOD-profiilien asetuksista.



Kuva 7. Patsasobjekti eri LOD-profiileilla.

LOD-profiilien ollessa käytössä viimeinen testi suoritettiin uudelleen. Testissä patsasobjekteja oli 2 000 kappaletta. Taulukossa 5 on nähtävillä uuden testin tulokset sekä vertailun vuoksi aiempi testitulos, jossa LOD-profiilit eivät olleet käytössä.

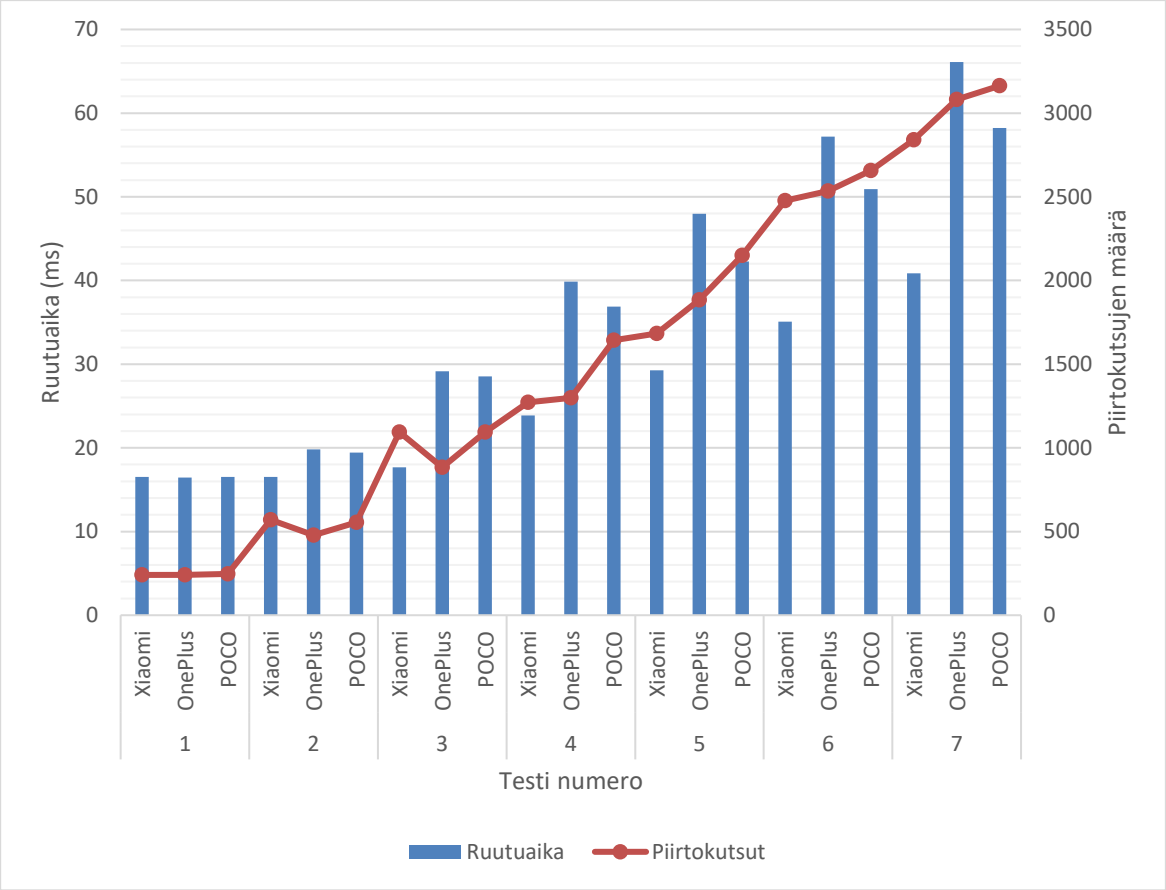
Taulukko 5. 2 000 kappaleen kolmiotestin tulokset ilman LOD-profiileja sekä niiden kanssa.

Objektien määrä	Mitattu arvo	Xiaomi Mi 9T Pro	OnePlus Nord	POCO X4 Pro 5G
2 000 kpl LOD:t ei käytössä	Ruutuaika (ms)	43,108	47,568	50,911
	Kolmiot	8 036 783,42	7 131 653,51	8 032 369,34
2 000 kpl LOD:t käytössä	Ruutuaika (ms)	21,867	30,198	28,973
	Kolmiot	864 732,88	919 887,57	1 185 353,28

Yksityiskohtien tason käyttöön ottaminen vähensi ruutuaikaa huomattavasti kaikilla laitteilla. Xiaomi-laitteen ruutuaika pieneni 49,2 prosenttia, OnePlus-laitteen ruutuaika pieneni 36,5 prosentilla ja POCO-laitteen ruutuaika pieneni 43,1 prosentilla. Kolmioiden määrä putosi Xiaomi-laitteella 10,8 prosenttiin, OnePlus-laitteella 12,9 prosenttiin ja POCO-laitteella 14,8 prosenttiin, kun ylempänä mainitut LOD-profiilit otettiin käyttöön.

5.3 Piirtokutsut

Piirtokutsujen vaikutusta ruutuaikaan testattiin lisäämällä kenttään useita objekteja. Kaikille objekteille lisättiin eri materiaalit. Objekteina käytettiin dynaamisia objekteja ja eri materiaaleja oli käytössä noin 80 erilaista. Tarkat tulokset näkyvät liitteessä 2 ja kuvassa 8 on tuloksista kerätty kuvaaja. Testejä tehtiin seitsemän ja jokaiseen testiin lisättiin objekteja.



Kuva 8. Piirtokutsu-testien tulokset.

Piirtokutsujen määrät vaihtelivat eri laitteilla. Jokaisella laitteella ruutuaika hidastui piirtokutsujen noustessa. OnePlus suoriutui testeissä heikoimmin ja Xiaomi suoriutui huomattavasti parhaiten. Ensimmäisessä testissä piirtokutsujen määrät olivat lähes samat jokaisella laitteella. Toisessa ja kolmannessa testissä Xiaomi-laitteen piirtokutsut olivat hieman muita korkeammat, mutta ruutuaika oli alhaisin. Lopuissa testeissä Xiaomin piirtokutsut olivat kaikkein alhaisimmat, OnePlus-laitteella toiseksi alhaisimmat ja POCO-laitteen piirtokutsut olivat kaikista korkeimmat. Vaikka OnePlus-laitteella piirtokutsuja oli vähemmän kuin POCO-laitteella, OnePlus-laitteen ruutuaika oli pidempi kuin POCO-laitteella.

5.4 Jälkikäsittelyefektit

Testit toteutettiin lisäämällä kenttään muutamia objekteja efektien vaikutuksen näkemiseksi ja lisäämällä jälkikäsittelyefektejä. Testistä kerättiin dataa ja verrattiin efektien vaikutusta ruutuaikaan. Jotta voitiin havainnoida jälkikäsittelyefektien vaikutusta suoritusaikaan, testiin lisättiin useita eri jälkikäsittelyefektejä. Lisättyjä efektejä ovat syväterävyys eli depth of field, bloom-efekti, linssinheijastus, kromaattinen aberraatio, vinjetti sekä ympäristön okkluusio. Linssinheijastusefekti oli päällä, mutta sen efektiä ei näkynyt testeissä testilaitteilla. Kuva 9 havainnollistaa testissä käytössä olevat efektit.

Kuva 9. Jälkikäsittelyefektit-testi.

Jälkikäsittelyefektien arvot laitettiin korkeiksi, jotta efektit olisivat voimakkaita. Taulukko 6 esittelee testeistä saadut tulokset. Testejä tehtiin kaksi, joissa ensimmäisessä oli aiemmin mainitut efektit käytössä ja toisessa efektejä ei ollut laitettu päälle.

Taulukko 6. Jälkikäsittely-testitulokset.

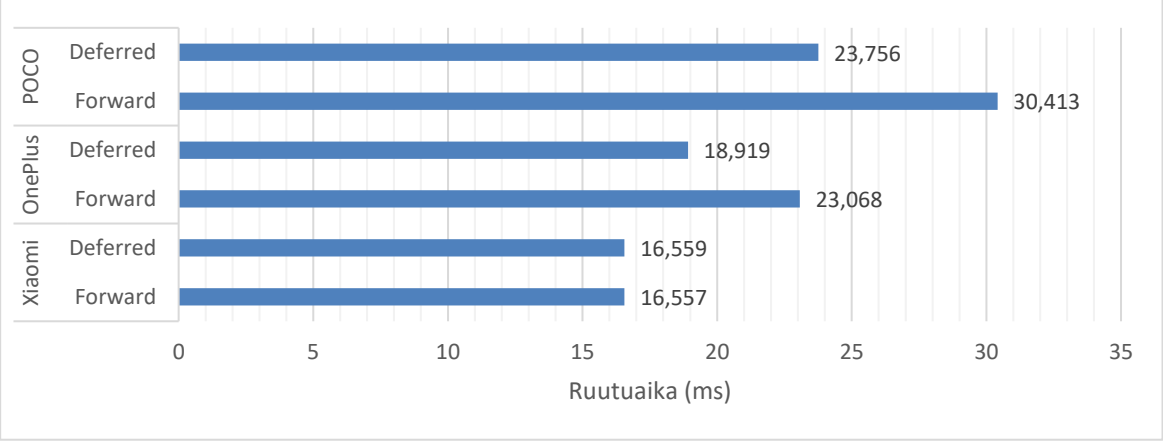
Testin kuvaus	Mitattu arvo	OnePlus	Xiaomi	POCO
Efektit käytössä	Ruutuaika (ms)	16,429	16,546	16,539
	Jälkikäsittelyefekti prosessi (ms)	0,38	0,267	0,437
Ei efektejä käytössä	Ruutuaika (ms)	16,35	16,554	16,55
	Jälkikäsittelyefekti prosessi (ms)	0,321	0,211	0,318

Näissä testeissä ei havaittu jälkikäsittelyefektien kuluttavan suorituskykyä mainittavasti, mutta testit suoritettiin hyvin pienessä skaalassa ja pienellä projektilla. Kun efektejä ei lisätty, jälkikäsittelyefekti prosessin kesto lyheni OnePlus-laitteella 0,09 ms ja POCO-laitteella 0,039 ms. Xiaomi-laitteella prosessi hidastui tässä testissä 0,007 ms.

5.5 Forward- vs. deferred-renderöinti

Unreal Engine tarjoaa erillisen renderöintipolun mobiilille. Forward-renderöinti on yleisesti ottaen parempi vaihtoehto, mutta deferred-renderöinti voi tarjota osalle peleistä paremman vaihtoehdon. Tyypillisesti pelit, joissa on suuria ulkoilmaympäristöjä ja voimakas riippuvuus dynaamisesta valaistuksesta, ovat sopivia deferred-renderöinnille. Kohdelaitteille, jotka käyttävät ruutupohjaista eli tile-based grafiikkasuorittimia, deferred-renderöinti sopii myös paremmin. Monimutkaista valaistusta vaativissa peleissä deferred-renderöinti voi säästää muistia ja parantaa pelin suorituskykyä. [23.]

Tätä testiä varten luotiin kartta, johon lisättiin jonkin verran objekteja sekä useita valoja. Projektissa oli aluksi käytössä forward-renderöinti ja toiseen testiin vaihdettiin deferred-renderöinti. Testit suoritettiin jokaisella laitteella ja kuvassa 10 on testien tuloksista kerätty kuvaaja.



Kuva 10. Renderöintitapojen testitulokset.

Deferred-renderöinti alensi ruutuaikaa OnePlus- sekä POCO-laitteilla. Xiaomi-laitteella ruutuajassa ei ollut nähtävissä mainittavaa muutosta. OnePlus-laitteella deferred-renderöinti oli noin 4,1 millisekuntia nopeampi ja POCO-laitteella noin 6,6 millisekuntia nopeampi. Pelin tiedostokoko oli deferred-renderöinnin ollessa käytössä noin 5 000 KB pienempi kuin forward-renderöinnin ollessa käytössä.

6 Yhteenveto

Testeissä voitiin todentaa, että eri laitteissa on jo yksinkertaisissa testeissä huomattavia eroja projektin ruutuajassa. Eri laitteiden suorituskyvyssä on siis eroja, minkä vuoksi optimoinnin tärkeys voitiin testituloksien perusteella todeta tarpeelliseksi, jotta peli toimisi paremmin laajemmalla laitevalikoimalla.

Objektien lisäämistesteissä havaittiin eroja spawn-funktion kestossa. Lisättävien objektien määrän kasvaessa, myös spawn-funktion aika kasvoi jokaisella laitteella. Spawn-funktion suoritusaika kasvoi melko tasaisesti jokaisella laitteella.

Kolmioiden ja piirtokutsujen testeistä saaduista tuloksista pystytään huomaamaan, että liialliset määrät kolmioita ja piirtokutsuja hidastavat ruutuaikaa huomattavasti. Näitä resursseja tulisi profiloida pelissä, kun peliin lisätään uusia objekteja tai rakennetaan uutta pelikenttää, jotta objektit tai kenttä voidaan tarvittaessa optimoida. Eri laitteiden välillä oli huomattavia eroja kolmioiden piirtämisessä ja piirtokutsuissa sekä niiden vaikutuksessa ruutuaikaan.

Jälkikäsittelyefekteistä ei saatu testeissä mainittavia tuloksia aikaan, joiden perusteella olisi voitu päätellä niiden olevan niin raskaita prosesseja, kuin Unreal Enginen dokumentaatiossa kerrotaan. Testeissä saadut tulokset olisivat voineet olla huomattavampia, jos ne olisi tehty valmiiseen peliprojektiin.

Forward- ja deferred-renderöinneissä huomattiin joillakin laitteilla ruutuajan olevan pienempi deferred-renderöinnin ollessa käytössä. Myös projektin tiedostokoko oli pienempi. Se, valitseeko kehittäjä renderöintitavaksi forward- vai deferred-renderöinnin, riippuu pelin tarpeista, mutta on hyvä testata molempia renderöintitapoja, jos kumpikin tavoista sopii pelin tarkoituksiin.

7 Pohdinta

Opinnäytetyötä tehdessä huomattiin joidenkin testilaitteiden kohdalla haasteita, jotta projekti saataisiin niillä toimimaan. Osa testeistä yritettiin tehdä Nokia G20- sekä Honor 8 -laitteilla. Nokia-laitteella testit eivät renderöityneet oikein, eikä projektia saatu asennettua Honor-laitteelle. Opinnäytetyötä olisi voinut laajentaa selvittämällä syyt, miksei joillakin testilaitteilla saatu suoritettua testejä ja miten projektin saisi toimimaan eri laitteilla. Käytettävissä olevan ajan puitteissa näitä syitä ei ollut mahdollista selvittää tässä opinnäytetyössä. Lopulta muita kuin Adreno-grafiikkaprosessoreita käyttäviä laitteita ei voitu ottaa mukaan testeihin.

Testausta tehdessä kohdattiin ongelmia myös käytetyn tietokoneen sekä Unreal Enginen kanssa. Näytönohjain kaatui muutaman kerran käytännön osuutta tehdessä, joka osaltaan vaikeutti ja hidasti projektin tekemistä. Unreal Engine kaatui näytönohjaimen kaatumisen seurauksena. Käytetyssä tietokoneessa on AMD Radeon RX 5700 XT -näytönohjain. Projekti saatiin myös rikkoutumaan, kun siihen yritettiin laittaa mobiiliesikatselu käyttöön. Esikatselun käyttöönoton jälkeen projekti ei enää auennut toimivana, vaan editorin valikot olivat päällekkäin, eikä mikään näppäin tai valikko ollut painettavissa.

Opinnäytetyön teoriaosuudessa tekijä laajensi tietämystään mobiilipelin optimoinnista ja mahdollisista huomioon otettavista seikoista mobiilipeliä kehittäessä. Käytännönosuus opetti tekijälle Unreal Engine -pelimoottorin profilointityökaluista ja niiden käytöstä optimoinnin tukena. Vastaan tulleet ongelmat eri testilaitteiden kanssa laajensi tekijälle mobiilipelien kehittämisessä esiintyviä mahdollisia ongelmia.

Opinnäytetyön tavoitteena oli nostaa esiin käyttäjien laitevalikoiman aiheuttamia haasteita sekä selvittää joidenkin yleisesti käytössä olevien toimintojen vaikutusta suorituskykyyn. Toimeksiantaja olisi toivonut testeihin vielä hieman lisää syvyyttä joiltain osin, mutta oli kuitenkin tyytyväinen opinnäytetyöhön.

Lähteet

[1] Wijman T. Newzoo's games market revenue estimates and forecasts by region and segment for 2023. Newzoo. [Internet]. 2023. [viitattu 03.09.2023]. Saatavilla: https://newzoo.com/resources/blog/games-market-estimates-and-forecasts-2023

[2] Kemp S. Digital 2023 october global statshot report. Data Reportal. [Internet]. 2023. [viitattu 03.09.2023. Saatavilla: https://datareportal.com/reports/digital-2023-october-global-statshot

[3] Microsoft Devices Blog [Internet]. [viitattu 05.09.2023]. Saatavilla: https://blogs.windows.com/devices/2013/01/16/10-things-you-didnt-know-about-mobile-gaming-2/

[4] Karthikeyan K. The History, Evolution, and Future of Mobile Gaming. Gameopedia [Internet]. 2023. [viitattu 05.09.2023]. Saatavilla: https://www.gameopedia.com/the-history-evolution-and-future-of-mobile-gaming/

[5] Laura Ceci. Number of apps available in leading app stores of 3rd quarter 2022. Statista [Internet]. 2023. [viitattu 05.09.2023]. Saatavilla: https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/

[6] Candy Crush Saga Is Apple's Most Downloaded App of 2013. ABC News [Internet]. 2013. [viitattu 13.10.2023]. Saatavilla: https://abcnews.go.com/Technology/candy-crush-saga-apples-downloaded-app-2013/story?id=21250929

[7] Mathilde. Pokémon Go: A successful gamification case study. LoQuiz [Internet]. 2023. [viitattu 05.09.2023]. Saatavilla: https://loquiz.com/2023/02/27/pokemon-go-gamification-case-study/

[8] Delaney M. How To Use Visualize Audio Effects In Fortnite. GameSpot. [Internet]. 2021. [viitattu 10.12.2023]. Saatavilla: https://www.gamespot.com/articles/how-to-use-visualize-audio-effects-in-fortnite/1100-6498314/

[9] Velin Tchalakov. Building Mobile Games with UE5 | Unreal Fest 2023 [Video]. Unreal Engine. Youtube. 13.10.2023 [viitattu 10.12.2023]. Saatavilla: https://www.youtube.com/watch?v=kfZ9BxrmYBU&ab_channel=UnrealEngine

[10] Trisnadoli A, Kreshna J. Optimization of Educational Mobile Game Design 'Ayo Wisata ke Riau' based on User's Perspective. IT Journal Research and Development. [Internet]. 2021. [viitattu 15.09.2023]. Saatavilla: https://journal.uir.ac.id/index.php/ITJRD/article/view/5766/3375

[11] Michael Klappenbach. Undestanding and Optimizing Video Game Frame Rates. Lifewire. [Internet] 05.11.2022 [viitattu 15.09.2023]. Saatavilla: https://www.lifewire.com/optimizing-video-game-frame-rates-811784

[12] Ari Arnbjörnsson. Maximizing Your Game's Performance in Unreal Engine | Unreal Fest 2022 [Video]. Unreal Engine. Youtube. 03.11.2022 [viitattu 15.09.2023]. Saatavilla: https://www.youtube.com/watch?v=GuIav71867E&ab_channel=UnrealEngine

[13] Unreal Engine. Wikipedia. [Internet]. [viitattu 16.12.2023]. Saatavilla: https://en.wikipedia.org/wiki/Unreal_Engine

[14] Unreal® Engine End User License Agreement. Unreal Engine. [Internet]. [viitattu 15.12.2023] Saatavilla: https://www.unrealengine.com/en-US/eula/unreal

[15] Unreal Frontend. Unreal Engine. [Internet]. [viitattu 26.09.2023] Saatavilla: https://docs.unrealengine.com/5.1/en-US/using-the-unreal-frontend-tool/

[16] Ionut Matasaru. Collect, Analyze, and Visualize Your Data with Unreal Insights | Unreal Fest Online 2020 [Video]. Unreal Engine. Youtube. 07.08.2020 [viitattu 26.09.2023]. Saatavilla: https://www.youtube.com/watch?v=Rf6oNkcGmX4&ab_channel=UnrealEngine

[17] Balancing Blueprint and C++. Unreal Engine. [Internet]. [viitattu 14.12.2023] Saatavilla: https://docs.unrealengine.com/4.27/en-US/Resources/SampleGames/ARPG/BalancingBlueprintAndCPP/

[18] Mobile Performance Tips and Tricks. Unreal Engine. [Internet]. [viitattu 12.10.2023]. Saatavilla: https://docs.unrealengine.com/4.27/en-US/SharingAndReleasing/Mobile/Performance/TipsAndTricks/

[19] Performance guidelines for mobile devices in Unreal Engine. Unreal Engine. [Internet]. [viitattu 09.08.2023]. Saatavilla: https://docs.unrealengine.com/5.1/en-US/performance-guidelines-for-mobile-devices-in-unreal-engine/

[20] Xiaomi Mi 9T Pro specs. Mi. [Internet]. [viitattu 14.12.2023]. Saatavilla: https://www.mi.com/uk/mi-9-t-pro/specs

[21] OnePlus Nord Tekniikka. Oneplus. [Internet]. [viitattu 14.12.2023]. Saatavilla: https://www.oneplus.com/fi/nord-specs

[22] POCO X4 Pro 5G Specs. POCO. [Internet]. [viitattu 14.12.2023]. Saatavilla: https://www.po.co/global/product/poco-x4-pro-5g/specs

[23] Mobile Deferred Shading Mode. Unreal Engine. [Internet]. [viitattu 14.12.2023]. Saatavilla: https://docs.unrealengine.com/5.2/en-US/using-the-mobile-deferred-shading-mode-in-unreal-engine/

Xiaomi Mi 9T Pro:lla tehdyn spawnaustestin tulokset:

25 kpl	1. testi	2. testi	3. testi	4.testi	KA
1. spawn	10,036	9,404	9,639	8,392	9,36775
2. spawn	16,01	8,832	8,314	8,227	10,34575
3. spawn	10,125	10,837	9,242	8,306	9,6275
4. spawn	8,654	14,322	8,463	7,45	9,72225

50 kpl	1. testi	2. testi	3. testi	4.testi	KA
1. spawn	26,618	18,963	20,551	17,456	20,897
2. spawn	19,494	24,081	18,699	18,026	20,075
3. spawn	22,619	23,168	17,676	19,858	20,83025
4. spawn	20,565	18,509	21,341	20,806	20,30525

100 kpl	1. testi	2. testi	3. testi	4.testi	KA
1. spawn	39,391	36,324	35,638	35,16	36,62825
2. spawn	38,276	38,722	35,626	37,762	37,5965
3. spawn	38,715	36,394	35,628	41,397	38,0335
4. spawn	39,105	36,214	38,583	39,054	38,239

150 kpl	1. testi	2. testi	3. testi	4.testi	KA
1. spawn	55,579	53,009	51,442	51,29	52,83
2. spawn	51,835	55,66	54,056	54,473	54,006
3. spawn	56,725	52,266	55,066	55,172	54,80725
4. spawn	56,107	51,711	55,293	54,431	54,3855

200 kpl	1. testi	2. testi	3. testi	4.testi	KA
1. spawn	71,609	67,86	67,715	68,013	68,79925
2. spawn	72,617	69,526	64,702	63,84	67,67125
3. spawn	69,902	77,689	66,703	72,475	71,69225
4. spawn	64,493	67,674	66,865	68,1	66,783

OnePlus Nord-laitteella tehdyn spawnaustestin tulokset:

25 kpl	1. testi	2. testi	3. testi	4.testi	KA
1. spawn	11,875	12,13	10,907	15,783	12,67375
2. spawn	10,235	11,554	15,096	17,606	13,62275
3. spawn	17,538	10,739	13,972	11,471	13,43
4. spawn	14,638	14,121	14,009	12,427	13,79875

50 kpl	1. testi	2. testi	3. testi	4.testi	KA
1. spawn	26,314	25,35	30,954	31,615	28,55825
2. spawn	26,701	35,854	65,872	27,438	38,96625
3. spawn	26,479	25,486	36,271	26,006	28,5605
4. spawn	26,898	25,343	27,071	29,978	27,3225

100 kpl	1. testi	2. testi	3. testi	4.testi	KA
1. spawn	52,178	51,772	46,632	49,403	49,99625
2. spawn	48,467	53,532	48,699	51,672	50,5925
3. spawn	55,111	49,34	49,343	50,204	50,9995
4. spawn	51,226	50,94	52,488	44,839	49,87325

150 kpl	1. testi	2. testi	3. testi	4.testi	KA
1. spawn	77,663	70,441	69,253	71,671	72,257
2. spawn	70,989	67,096	71,139	72,924	70,537
3. spawn	73,979	70,217	71,012	68,214	70,8555
4. spawn	70,36	71,98	73,14	70,771	71,56275

200 kpl	1. testi	2. testi	3. testi	4.testi	KA
1. spawn	89,225	111,446	89,177	87,457	94,32625
2. spawn	99,702	88,359	89,616	88,973	91,6625
3. spawn	91,725	90,469	89,843	93,804	91,46025
4. spawn	103,777	94,027	98,005	96,169	97,9945

POCO-laitteella tehdyn spawnaustestin tulokset:

25 kpl	1. testi	2. testi	3. testi	4.testi	KA
1. spawn	16,063	15,133	14,464	12,626	14,5715
2. spawn	14,28	14,705	12,939	12,77	13,6735
3. spawn	14,686	17,871	14,585	14,78	15,4805
4. spawn	12,96	12,342	14,812	15,261	13,84375

50 kpl	1. testi	2. testi	3. testi	4.testi	KA
1. spawn	28,379	28,237	26,731	28,664	28,00275
2. spawn	29,23	31,993	27,886	27,712	29,20525
3. spawn	30,52	34,082	26,446	29,246	30,0735
4. spawn	32,206	31,367	27,562	30,842	30,49425

100 kpl	1. testi	2. testi	3. testi	4.testi	KA
1. spawn	51,476	52,287	50,254	55,838	52,46375
2. spawn	53,951	53,9	54,42	60,299	55,6425
3. spawn	60,128	57,684	53,894	53,141	56,21175
4. spawn	53,384	51,298	52,753	53,595	52,7575

150 kpl	1. testi	2. testi	3. testi	4.testi	KA
1. spawn	75,048	75,463	76,37	74,714	75,39875
2. spawn	81,299	77,947	73,917	74,099	76,8155
3. spawn	73,27	76,593	75,282	73,756	74,72525
4. spawn	80,285	82,409	75,345	77,545	78,896

200 kpl	1. testi	2. testi	3. testi	4.testi	KA
1. spawn	99,396	98,405	95,389	101,234	98,606
2. spawn	96,622	98,453	95,585	98,535	97,29875
3. spawn	110,59	102,067	96,135	96,229	101,2553
4. spawn	94,908	101,577	96,373	101,235	98,52325

Piirtokutsu-testien tulokset:

Testi	Laite	Ruutu-aika	Piirtokut-sut
1	Xiaomi	16,508	240,61
	OnePlus	16,436	240,81
	POCO	16,537	246,59
2	Xiaomi	16,531	569,21
	OnePlus	19,807	478,26
	POCO	19,441	555,23
3	Xiaomi	17,688	1 094,96
	OnePlus	29,161	884,35
	POCO	28,556	1 094,27
4	Xiaomi	23,888	1 271,24
	OnePlus	39,861	1 298,13
	POCO	36,865	1 642,22
5	Xiaomi	29,276	1 682,56
	OnePlus	47,973	1 883,25
	POCO	42,266	2 149,23
6	Xiaomi	35,075	2 477,01
	OnePlus	57,18	2 533,81
	POCO	50,923	2 656,21
7	Xiaomi	40,841	2 840,44
	OnePlus	66,115	3 082,35
	POCO	58,24	3 163,25