OULUN AMMATTIKORKEAKOULU

Johannes Tasala

UNITY- JA GODOT-PELIMOOTTOREIDEN VERTAILU 3D-PELIKEHITYKSESSÄ

UNITY- JA GODOT-PELIMOOTTOREIDEN VERTAILU 3D-PELIKEHITYKSESSÄ

Johannes Tasala
Opinnäytetyö
Kevät 2024
Tieto- ja viestintätekniikka
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tieto- ja viestintätekniikka, Ohjelmointi

Tekijä(t): Johannes Tasala
Opinnäytetyön nimi: Unity- ja Godot-pelimoottoreiden vertailu 3D-pelikehityksessä
Työn ohjaaja(t): Manne Hannula
Työn valmistumislukukausi ja -vuosi: Kevät 2025 Sivumäärä: 26

Tässä opinnäytetyössä vertaillaan Unity ja Godot pelimoottoreita 3D-pelikehityksessä. Työn lähtökohtana on vähäinen kokemus kummastakin pelimoottorista sekä niissä käytettävistä ohjelmointikielistä. Opinnäytetyössä käytetyt ohjelmointikielet ovat C# ja GDScript.

Opinnäytetyön tavoitteena oli selvittää mitä eroja pelimoottoreiden välillä on suorituskyvyllisesti, ominaisuuksissa, mukavuuksissa ja hinnassa. Suorituskykyä vertailtiin ohjelmoimalla molemmilla pelimoottoreilla samanlaiset stressitestit. Stressitestien aikana seurattiin ruudunpäivitysnopeutta sekä prosessorin ja näytönohjaimen suoritusnopeuksia. Ominaisuuksia ja mukavuuksia vertailtiin stressitestejä ohjelmoitaessa. Hintavertailussa huomioon otettiin käyttöönottokustannus sekä mahdolliset lisäkustannukset.

Tuloksien perusteella ei päästy yhteen selkeään johtopäätökseen siitä kumpi pelimoottori on sopivampi. Stressitestien perusteella oli selkeää että Unity on suorituskyvyllisesti paljon Godotia tehokkaampi. Hintavertailussa täysin ilmainen Godot taas loisti Unityn maksusuunnitelmien rinnalla.

Asiasanat: Pelimoottori, Videopeli, Godot, Unity

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Information Technology, Option of Software Development

---

Author(s): Johannes Tasala
Title of thesis: Comparison of Unity and Godot game engines in 3D-game development
Supervisor(s): Manne Hannula
Term and year when the thesis was submitted: Spring 2025
Number of pages: 26

In this thesis I compare Unity and Godot game engines in 3D-game development. The reason I chose to compare these two engines was that I had some development experience with them beforehand and the programming languages they use. The programming languages used for the project part of the thesis were C# and GDScript.

The goal of the thesis was to see what differences there are between Unity and Godot in performance, features and pricing. To measure the performance of the game engines I programmed stress tests that put heavy load on the computer. During the stress test I monitored the frames per second and how fast the CPU and GPU can handle their tasks. During development I compared the features of both game engines. The price comparison consisted of the implementation price and possible additional costs.

The result of the thesis was not one clear winner. Unity performed way stronger with the stress tests and had a way better built in profiler. Godot won the price contest by being completely free while Unity has many different subscription plans and additional fees.

Keywords: Game engine, Videogame, Godot, Unity

SISÄLLYS

1 JOHDANTO

Pelimoottori on videopelien kehityksessä käytettävä työkalu, jonka tarkoitus on helpottaa pelinkehittäjän työtä tarjoamalla käyttöliittymä sekä valmiita ominaisuuksia, joita peleissä yleisimmin tarvitaan. Pelimoottori on yleensä kohdennettu tietynlaisten pelien tekemiseen ja yleisimmin pelimoottorit jaetaan kolmeen eri genreen, 2D-pelit, 3D-pelit ja puhelinpelit. Pelimoottoreiden kustannukset käyttäjälle voi vaihdella täysin ilmaisesta todella kalliisiin lisensseihin tai vuosimaksuihin. Neogamesin (2023) teettämän tutkimuksen mukaan Suomen pelialalla oli vuonna 2022 3,2 miljardin liikevaihto, aktiivisia pelistudioita oli 232 ja työntekijöitä 4100. Globaalissa mittakaavassa Suomen osuus on pieni, sillä Newzoon raportti kertoo että globaali liikevaihto pelialalla oli 187,7 biljoonaa vuonna 2023. (Crichton-Stuart 2023.)

Videopelaaminen on harrastus, joka kasvattaa suosiotaan joka vuosi ja joillekkin pelaajille siitä on tullut e-urheilun kautta myös ammatti. Exploding Topics blogiin kirjoitetun artikkelin mukaan vuonna 2024 koko maailmassa oli yhteensä 3,32 miljardia pelaajaa. Näistä pelaajista 80 % on iältään yli 18-vuotiaita. E-urheilijoita eli ammatikseen kilpaa pelaavia on maailmassa yli kaksitoistatuhatta. (Duarte 24.4.2025.) Suomessa elantonsa pelaamalla tienaavia e-urheilijoita arvioidaan olevan noin 50–100 (Hakaniemi 26.3.2023).

Tässä opinnäytetyössä vertaillaan Godot ja Unity pelimoottoreita 3D-pelikehityksessä ja pyritään selvittämään kumpi vastaa paremmin yksittäisen peliohjelmoijan tai pienen peliyrityksen tarpeisiin. Tavoitteena on selvittää minkälaisia eroja näiden pelimoottoreiden välillä on suorituskyvyllisesti, ominaisuuksissa sekä kustannuksissa. Tuloksia käytetään hyödyksi, kun valitaan sopivampi pelimoottori tulevaisuuden projekteja varten. Tulokset voivat myös auttaa muita peliohjelmoijia sekä pieniä peliyrityksiä.

Opinnäytetyön tarkoituksena ei ole vastata kysymykseen kumpi pelimoottori, Godot vai Unity, on parempi, vaan tarkastella ja selvittää aloittelevan peliohjelmoijan näkökulmasta kumpi vastaa paremmin sen hetkisiin tarpeisiin ja kumman pelimoottorin käyttö tuntuu paremmalta. Suorituskykyä vertaillaan käyttämällä pelimoottoreiden perusasetuksia sekä samaa tietokonetta. Tuloksia tarkastellessa on hyvä tiedostaa, että minulla on vain vähän kokemusta Unityn ja Godotin käytöstä sekä C#– ja GDScript–ohjelmointikielistä, eikä ollenkaan kokemusta videopelien optimoinnista.

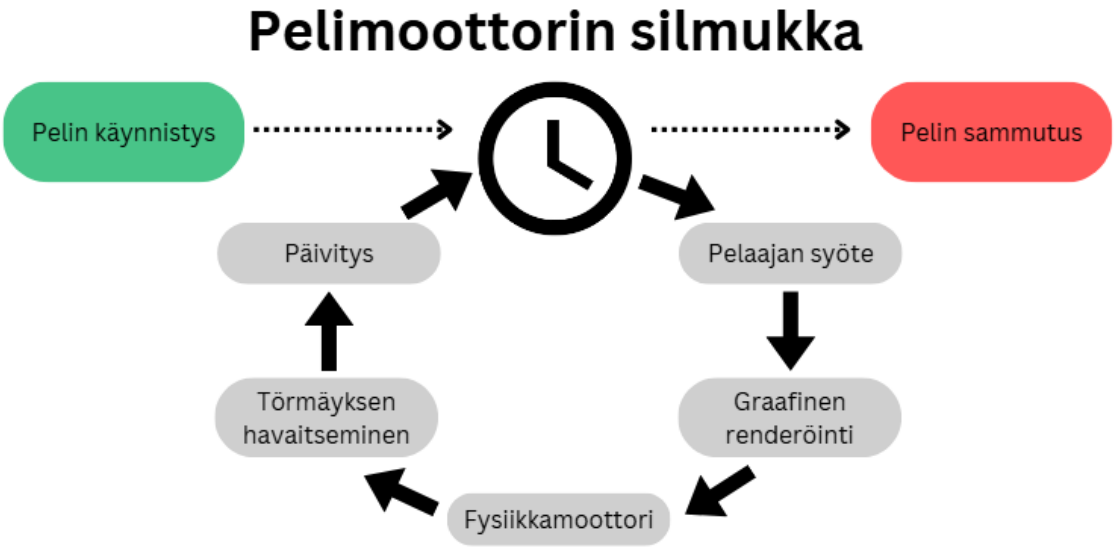
2 PELIMOOTTORIT

Pelimoottori-termiä käytetään siitä osasta videopeli ohjelmistoa, joka on vastuussa ohjelmiston ydinkomponenteista. Esimerkiksi grafiikkojen renderöinti, fysiikkamoottori sekä äänijärjestelmä ovat osa pelimoottoria. Pelinkehittäjän on valittava ohjelmoiko hän oman pelimoottorinsa, vai käyttääkö jotain valmista pelimoottoria. Varsinkin pienet peliyritykset tai yksittäiset peliohjelmoijat käyttävät usein valmiita pelimoottoreita, sillä oman pelimoottorin ohjelmointi voi olla taloudellisesti ja ajallisesti raskasta. Pelimoottoria valittaessa yleisesti otetaan huomioon pelin genre, mille alustalle peliä ollaan tekemässä sekä mahdolliset kustannukset. (Gregory 2018.) Tällä hetkellä suosituimpia tietokonepelien pelimoottoreita ovat esimerkiksi Unreal Engine, Unity, Godot Engine sekä Amazon Lumberyard. (Sibony 2024.)

Pelimoottoreita käytetään nykyään myös muiden projektien, kuin pelkkien videopelien kehitykseen. Pelimoottori tarjoaa käyttöliittymän, jolla voi yhdistää graafista renderöintiä, ääntä sekä käyttäjän syötteet, joten se soveltuu mainiosti esimerkiksi interaktiivisten- tai multimediasovellusten kehitykseen. (Dornhege.)

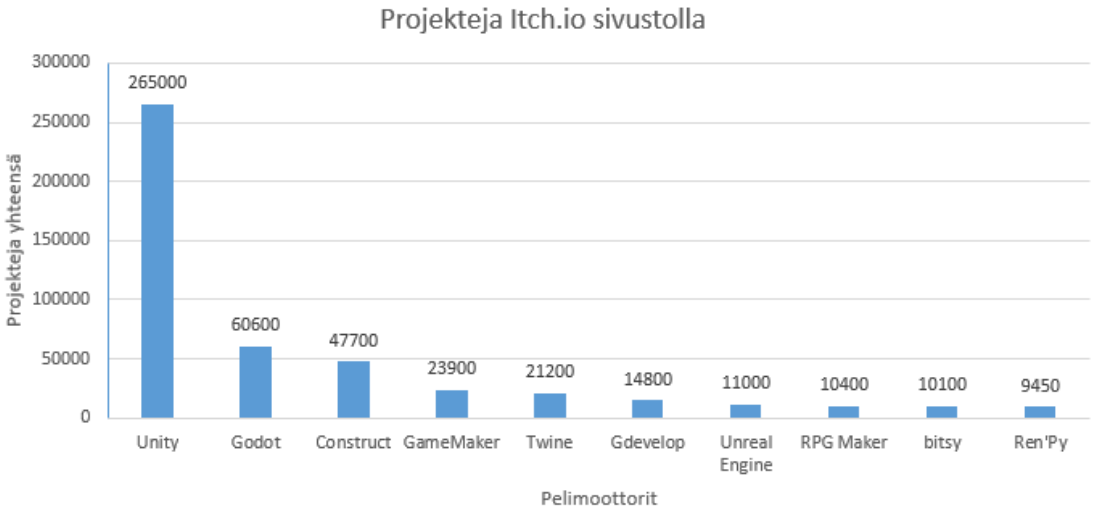
Yksinkertaistettuna pelimoottorin perus toimintaperiaate on käynnistää videopeli ja sen jälkeen suorittaa pelimoottorin silmukkaa jatkuvasti niin kauan, kunnes peli sammutetaan (kuva 1). Pelimoottorin silmukassa käsitellään pelaajan antama syöte, joka voi olla esimerkiksi näppäimen painaminen tai hiiren liikuttaminen. Sen jälkeen silmukassa käsitellään graafinen renderöinti eli näytönohjaimelle lähetetään tarvittavat tiedot kaikista pelin 2D- ja 3D-elementeistä, jotta näytönohjain osaa tuottaa näytölle kuvaa, jossa nämä elementit näyttävät oikeilta ja ovat oikeilla paikoillaan. Fysiikkamoottorin tehtävä on luoda peliin halutut fysiikat. Fysiikkamoottorilla voidaan esimerkiksi toteuttaa peliin alaspäin työntävä voima jolla jäljitellään maan painovoimaa tai kohdistaa pelihahmoon voimaa tietystä suunnasta perustuen pelaajan antamiin syötteisiin, jolloin pelihahmo liikkuu haluttuun suuntaan. Törmäyksen havaitseminen on yhteydessä fysiikkamoottoriin ja sen tehtävänä on tarkistaa, törmäsivätkö jotkin pelin elementit toisiinsa kyseisen pelimoottorin silmukan aikana. Jos esimerkiksi pallon muotoiseen 3D-elementtiin kohdistetaan painovoimaa ja se törmää pelimaailman lattiaan, lähetetään törmäyksestä tieto fysiikkamoottorille, joka laskee pallon nopeudesta riippuen paljonko palloon tulee kohdistaa vastavoimaa, jolloin pelaajalle näyttää kuin pallo pomppaisi maahan törmätessään. Törmäyksen havaitsemisen ansiosta pelin elementit eivät pääse lävistämään toisiaan. Silmukan viimeinen osio

on päivitys, se tarkoittaa että pelimoottori suorittaa kaikki pelinkehittäjän tekemät päivitysmenetelmät ennen seuraavaan silmukkaan siirtymistä. (Serrano 2019.)



KUVA 1. Pelimoottorin silmukan toimintaperiaate

Itch.io on videopeleihin keskittynyt nettikauppa, jossa pelien kehittäjät voivat jakaa ja myydä tekemiään pelejä vapaasti. Sivusto on suosittu varsinkin yksittäisten sekä aloittelevien pelinkehittäjien keskuudessa. Sivustolta löytyy projektien määrät jokaista pelimoottoria kohden ja Unity on selvästi muita moottoreita suositumpi (kuva 2).



KUVA 2. Kaavio projektien määrästä Itch.io sivustolla

2.1 Godot Engine

Godot Engine on 2D- ja 3D-pelien kehitykseen kohdistuva avoimen lähdekoodin pelimoottori, joka julkaistiin GitHubiin helmikuussa 2014 MIT lisenssin alla. Ennen julkaisua pelimoottoria kehittivät Juan Linietsky ja Ariel Manzur, julkaisun jälkeen kehitys on yhteisövetoista ja osallistujia onkin jo yli 2400. Pelimoottori tukee pelien vientiä usealle eri alustalle, kuten tietokoneelle, mobiili-laitteille, selaimille sekä konsoleille. (GitHub 2024.) Tässä opinnäytetyössä käytetään marraskuussa 2023 julkaistua Godotin versiota 4,2 GDScript osuuden tekemiseen. Godotin C# osuus toteutetaan uudemmalla versiolla 4,4, koska vanha versio ei tue tarvittavan dotnet ohjelmistokehyksen käyttöä. Versio 1,0 Godotista julkaistiin joulukuussa 2014. (Godot Docs 2024.)

2.2 Unity

Unity on vuonna 2005 julkaistu pelimoottori, jonka on kehittänyt Unity Technologies. Unity pelimoottoria voi käyttää 2D- ja 3D-pelien sekä simulaatioiden kehitykseen ja se tukee vientiä useille eri alustoille kuten tietokoneelle, mobiili-laitteille, konsoleille, selaimille sekä virtuaalitodellisuus-alustoille. Unity Technologies julkaisee pelimoottorista pitkäaikainen tuki (Long Term Support) version kerran vuodessa, tämä versio saa säännöllisiä päivityksiä kolmen vuoden ajan. Viimeisin tällainen versio jota tässäkin projektissa käytetään on 2022.3 ja se julkaistiin 30.5.2023. (endoflife.date 2024.)

3 TAVOITE JA TARKOITUS

Unity ja Godot ovat hyvin samanlaisia pelimoottoreita ja niitä vertaillaankin keskenään usein nettikeskusteluissa. Halusin kuitenkin itse selvittää, mitä eroavaisuuksia huomaan ja kumpi sopii paremmin omiin tarpeisiini. Tutkin pelimoottoreiden suorituskykyä seuraamalla prosessorin ja näytönohjaimen suoritusaikoja sekä ruudunpäivitysnopeutta stressitestin aikana. Hintavertailussa tutkin pelimoottoreiden käyttöönotto hintaa sekä mahdollisia lisäkustannuksia.

3.1 Pelimoottorin valinta

Pelimoottoria valittaessa kolme tärkeintä huomioon otettavaa seikkaa ovat tulevan projektin vaatimukset, aiempi osaaminen liittyen pelimoottoriin ja käytettävään ohjelmointikieleen sekä budjetti. Valitsin vertailun kohteeksi Unityn sekä Godotin, koska olen käyttänyt molempia aikaisemmin peliprojekteja tehdessä. Unity on suosittu pelimoottori 3D-pelien kehityksessä, koska sen suorituskyky kykenee käsittelemään visuaalisesti raskaita pelejä. Godot ei ole yhtä tehokas raskaiden 3D-pelien kehityksessä, mutta 2D-pelikehitykseen mainio valinta. Ialtään Unity on Godottia vanhempi, jonka takia Unityn käyttöön löytyy enemmän opetusmateriaalia ja esimerkkejä. Molemmilla pelimoottoreilla on kuitenkin käyttäjäyhteisöt, joista voi tarvittaessa pyytää apua. (Wayline 2024.)

3.2 Suorituskyky

Videopelejä pelatessa mikään ei ole niin raivostuttavaa, kuin pelin pätkiminen tai heikko suorituskyky. Pelin pätkimisen syy voi olla esimerkiksi se, ettei pelaajan oma laite, jolla peliä pelataan ole tarpeeksi tehokas pelin vaatimuksiin nähden tai peli on ohjelmoitu tai optimoitu huonosti. Videopeleissä yleisin suorituskykyä mittaava tekijä on ruudunpäivitysnopeus eli FPS (Frames Per Second). Ruudunpäivitysnopeus kertoo nimensä mukaisesti, kuinka monta kertaa ruutu päivitetään sekunnin aikana. Mitä korkeampi ruudunpäivitysnopeus on sitä tasaisemmalta pelaaminen tuntuu, kun taas liian alhainen ruudunpäivitysnopeus aiheuttaa pelin pätkimistä ja hankaloittaa pelaamista. Tietokoneella pelattaessa tärkeimmät tietokoneen komponentit ovat näytönohjain sekä prosessori. Näytönohjaimen tehtävä on huolehtia pelin graafisista ominaisuuksista ja niiden näyttämisestä ruudulla. Mitä graafisesti raskaampi ja yksityiskohtaisempi

peli on sitä tehokkaampi näytönohjain tasaiseen ja korkeaan ruudunpäivitysnopeuteen tarvitaan. Prosessorin tehtävä on hoitaa pelin logiikka ja laskelmoinnit, fysiikan simulaatiot, pelaajan syötteet, resurssienhallinta sekä kommunikoida muiden tietokoneen komponenttien kanssa. Tehokasta prosessoria tarvitaan varsinkin verkkoyhteyttä vaativissa moninpeleissä ja laajoissa simulaatiopeleissä. (Dobbin 2024.)

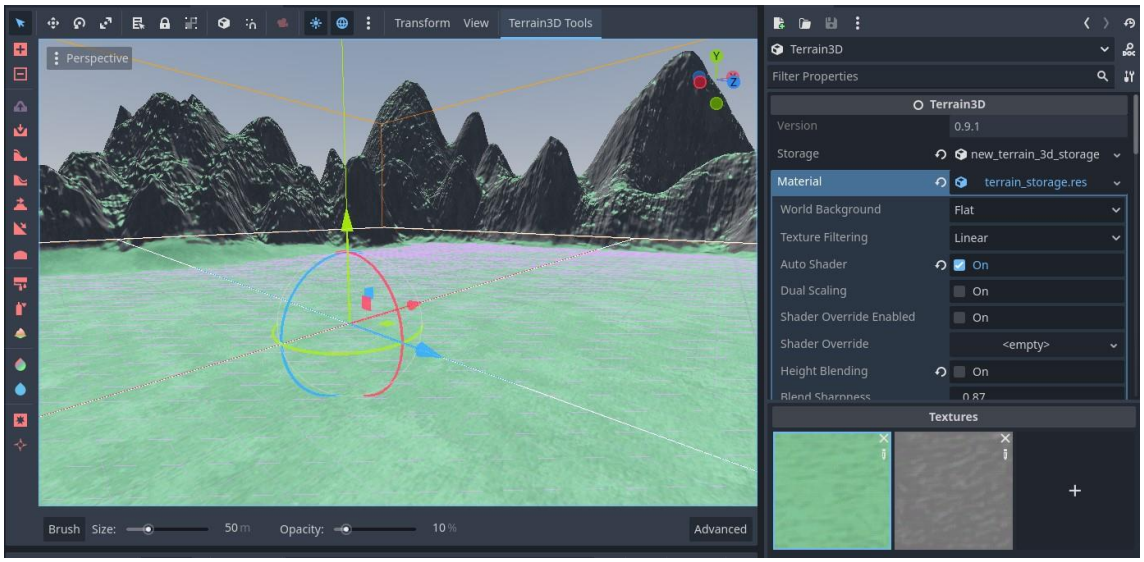
Tässä opinnäytetyössä käytetään samaa tietokonetta kaikkien stressitestien toteuttamiseen, jotta testituloksien vertailut ovat mahdollisimman reiluja. Kuten jo aiemmin manitsin minulla ei ole kokemusta videopelien optimoinnista eikä tämän opinnäytetyön tarkoituksena ole saavuttaa stressitesteillä optimaalisinta lopputulosta. Tarkoitus on nähdä miten pelimoottoreiden suorituskyvyt eroavat toisistaan niiden perusasetuksilla, eri ohjelmointikielillä sekä aloittelevan peliohjelmoijan ohjelmointitaidoilla.

4 TOTEUTUS

4.1 Pelimaailman kehitys

Pelimaailma on se ympäristö, johon videopeli sijoittuu. Pelimaailma on todella tärkeässä roolissa videopeleissä ja pelinkehityksessä, sillä se tukee pelin muita elementtejä, kuten mekaniikkoja sekä tarinaa. Kaunis ja toimiva pelimaailma auttaa pelaajaa uppoutumaan peliin, sekä sen avulla peliä voi markkinoida ja myydä. (Bradley 2024.) Tämän projektin pelimaailman suunnitelmana oli luoda yksinkertainen vuoristokenttä, jossa vuoristot toimisivat pelimaailman rajoina joita pelaaja ei voi ylittää. Godot pelimoottorissa ei ole valmiina mitään työkalua, jolla tällaisen pelimaailman luonti onnistuisi, joten käytin TokisanGames:in tekemää Terrain3D maastonmuokkaus työkalua kuvassa 3 näkyvän vuoristokentän tekoon. Terrain3D on Godot pelimoottorille tehty ilmainen lisätyökalu, jota oli helppo käyttää ja jonka avulla vuoristoisen kentän luonti onnistui hyvin.

Kentän luonti aloitettiin tekemällä suuri tasainen levy, joka toimii kentän pohjana ja jonka päällä pelihahmo voi kävellä. Tätä tasaista levyä muokattiin vuoriston malliseksi käyttämällä Terrain3D:n nosto-ominaisuutta. Tämän jälkeen maalattiin ruohikko ja kivikko tekstuurit Microsoftin Paint 3D sovelluksella ja vietiin ne Godottiin png-muodossa. Kenttä teksturoitiin automaattisesti määrittämällä Terrain3D työkaluun haluttu nousukulma, joka erottaa kivikko ja ruohikko tekstuurit. Kuvan 3 vasemmassa reunassa näkyy työkalun eri maastonmuokkaus ominaisuudet ja vasemmassa alareunassa näkyy Paint 3D:llä tehdyt tekstuurit.



KUVA 3. Terrain3D lisäosa Godot Enginen käyttöliittymässä

Halusin luoda Unityyn täysin identtisen maailman, joten käytin Terrain3D:n korkeuskartan vienti työkalua. Unityn oma maastotyökalu pystyy käyttämään tätä korkeuskarttaa hyödyksi luoden samanlaisen vuoristomaiseman. Korkeuskartta ei kuitenkaan sisältänyt Godotissa maalattuja ruoho ja kivi tekstuureja, joten käytin Unityn lisäosa kaupasta löytyvää Procedural Terrain Painter lisäosaa tekstuurien maalamiseen. Valitsin kyseisen lisäosan sen automaattisen maaston nousukulmaan perustuvan tekstuurimaalaus ominaisuuden takia. Säädin työkalun niin että maaston tasaiset osuudet maalataan ruohotekstuurilla ja kun maaston nousukulma muuttuu tarpeeksi jyrkäksi vaihtuu tekstuuri kallioksi. Samanlaista ominaisuutta käytin Godotin maaston teksturointiin Terrain3D työkalulla.

4.2 Ohjelmointikielet

Tässä opinnäytetyössä käytettiin kahta eri ohjelmointikieltä. Unity toteutus ohjelmoitiin C#–ohjelmointikielellä ja Godot toteutukset C#– ja GDScript–ohjelmointikielillä. Vertailuja tehdessä täytyy ottaa huomioon, ettei vertailtavien toteutuksien koodi ole täysin samanlaista joka varmasti vaikuttaa tuloksiin jollain tavalla. Päätin toteuttaan Godotin kahdella eri kielellä, sillä minua kiinnosti nähdä kuinka paljon ohjelmointikieli vaikuttaa pelin suorituskykyyn.

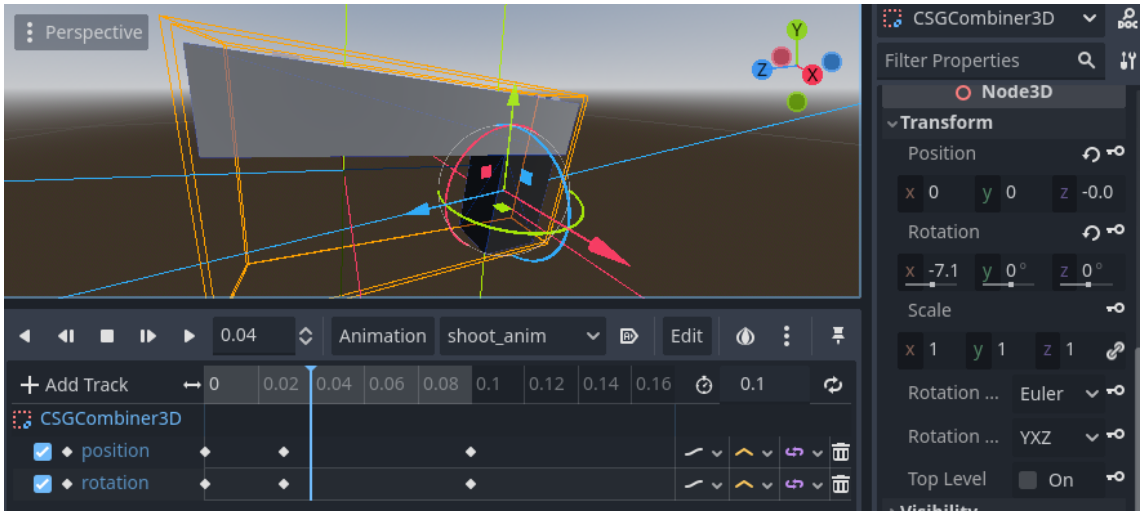
Unityssä ei ole sisäänrakennettua ohjelmointiympäristöä joten käytin projektin sen osuuden ohjelmoimiseen Microsoft Visual Studio 2022 ohjelmaa. Godotissa on sisäänrakennettu ohjelmointiympäristö ja se tukee GDScript kielellä ohjelmointia, joten käytin sitä GDScript

toteutuksen tekoon. Godotin C# osuuden ohjelmoin Visual Studio Code ohjelmointiympäristöllä, koska sisäänrakennetussa ohjelmointiympäristössä C# tuki on minimaalinen ja ulkoisen ohjelman käyttö on suositeltua.

4.3 Pelihahmo ja animointi

Pelihahmo on se objekti pelissä, mitä pelaaja liikuttaa käyttäen näppäimistöä ja hiirtä. Ensimmäisen persoonan ammuntapeleissä (First Person Shooter) pelikamera sijoitetaan pelihahmon pään sisälle, jotta pelaaja saa vaikutelman, että hän katsoisi pelimaailmaa pelihahmon silmin. Pelihahmoa liikutetaan eteen, taakse ja sivuttain käyttämällä WASD näppäimiä ja kameraa käännellään hiirellä.

Videopeleissä animoinnilla luodaan eloa pelimaailmaan ja pelihahmoihin esimerkiksi animoimalla pelihahmoille kävelyanimaatio. Tässä projektissa animointityökalua käytettiin pistoolin ampumisrekyylin animoimiseen (kuva 4). Animointityökaluun syötettiin ensin pistoolin perustilan sijainti, tämän jälkeen ajaksi asetettiin 0,02 sekuntia ja pistoolia käännettiin sekä liikutettiin hieman taaksepäin. Lopuksi aika asetettiin 0,1 sekuntiin ja pistooli palautettiin alkuperäiseen asentoon. Animaation kesto on siis yhteensä 0,1 sekuntia ja kun animaation käynnistää liikkuu pistooli syötettyihin asentoihin annettuina aikoina saaden aikaan nopean aseen rekyyliä muistuttavan liikkeen. Molempien Godotin sekä Unityn animointityökalut toimivat samalla tavalla.

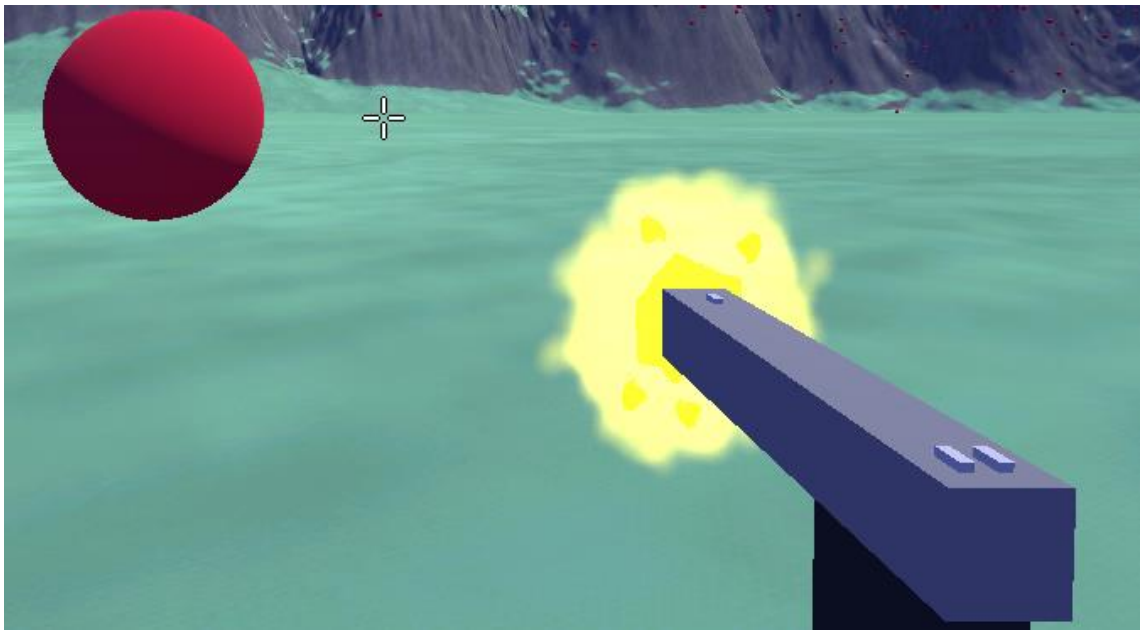


KUVA 4. Kuvakaappaus Godotin animointityökalusta

4.4 Äänet ja efektit

Äänillä ja efekteillä luodaan videopeleihin syvyyttä ja tunnelmaa. Videopelien äänimailmaan voi olla monimutkainen ja sisältää tunnelmaan sopivaa taustamusiikkia, pelihahmojen keskustelua tai aseiden räiskintää. Äänimaailma voi olla myös pelkistetty ja sisältää pelkästään muutamia eri ääniefektejä. Tärkeintä on, että äänimaailma sopii pelin teemaan ja on osana rakentamassa pelikokemusta. (Juegoadmin 2023.)

Tämän opinnäytetyön projektin äänimailman koostuu vain kolmesta itse ääninauhurilla tehdystä ääniklipistä. Ääniklippi yhdistetään peliobjektiin ja koodissa määritetään sille ehto, ehdon toteutuessa ääniklippi soitetaan ja jos pelihahmo on tarpeeksi lähellä äänilähdettä kuullaan ääni kuulokkeista. Projektin äänet ovat pallon kimpoamis ääni sen osuessa maahan, pallon poksahdus ääni pelaajan osuessa siihen pistoolilla sekä pistoolin laukausu ääni ammuttaessa. Ampumiseen tehtiin myös partikkelityökalulla sekä Paint3D:llä piirretyllä räjähdyskuviolla aseen suuliekkiä muistuttava visuaalinen efekti (kuva 5).



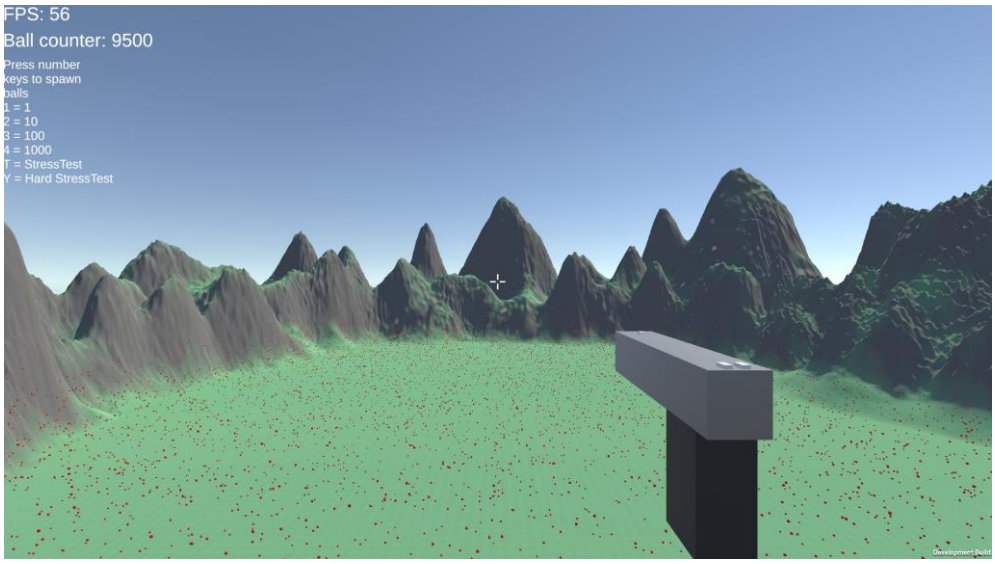
KUVA 5. Ammuttaessa syntyvä suuliekkiefekti

4.5 Stressitestit

Stressitestin ideana on luoda peliin suorituskyvyllisesti raskas skenaario ja samalla kerätä tietoa tietokoneen suoritustehoista. Näitä tietoja vertailemalla pystytään toteamaan kumpi pelimoottori suoriutui stressitestistä paremmin. Halusin stressitestissä kuormittaa tietokonetta monella eri

tapaa, joten tein testiä varten peliin pallon muotoisen esineen. Tähän palloon fysiikkamoottori kohdistaa painovoimaa muistuttavaa alaspäin kohdistuvaa voimaa. Pallon osuessa maahan siihen kohdistetaan vastavoima, joka kimmottaa sen takaisin ylöspäin yhtä korkealle, kuin mistä se tippuikin.

Tein peliin kevyen sekä raskaan stressitestin ja molempien toimintaidea on sama. Kuten kuvasta 6 on nähtävissä testin alkaessa pelihahmo siirretään kentän laidalle vuoren päälle ja kamera käännetään pelimaailman keskikohtaa kohti, jotta koko kenttä on näkyvissä. Sen jälkeen, joka sekunti synnytetään tietty määrä palloja satunnaisiin kohtiin taivaalle, kuitenkin niin että jokainen pallo on kameran nähtävissä. Kevyessä testissä joka sekunti syntyy 100 palloa 25 sekunnin ajan, eli yhteensä 2500 palloa. Raskaassa testissä palloja syntyy 500 joka sekunti eli yhteismäärä on 12500. Testin tulokset saadaan talteen kirjoittamalla ruudunpäivitysnopeus konsoliin joka sekunti 30 sekunnin ajan ja sen jälkeen kokoamalla tulokset excel taulukkoon.



KUVA 6. Kuvakaappaus pelin Unity toteutuksesta kun raskas stressitesti on käynnissä

4.6 Tietokoneen komponentit

Kuvassa 7 on listattuna stressitesteissä käytetyn tietokoneen komponentit. Prosessorina on AMD Ryzen 5 5600X ja näytönohjaimena Nvidia Geforce GTX 980 Ti. RAM-muistia on yhteensä 16 GB ja molemmat stressitestit on asennettu SSD-levylle. Tietokoneen komponentit on tarkoitettu raskaaseen pelaamiseen joten suoritustehon pitäisi riittää kohtalaisesti stressitestien aikana.

Processor	AMD Ryzen 5 5600X 6-Core Processor 3.70 GHz
Installed RAM	16,0 GB
Graphics Card	NVIDIA GeForce GTX 980 Ti (6 GB)
Storage	1.82 TB SSD Corsair MP600 PRO NH

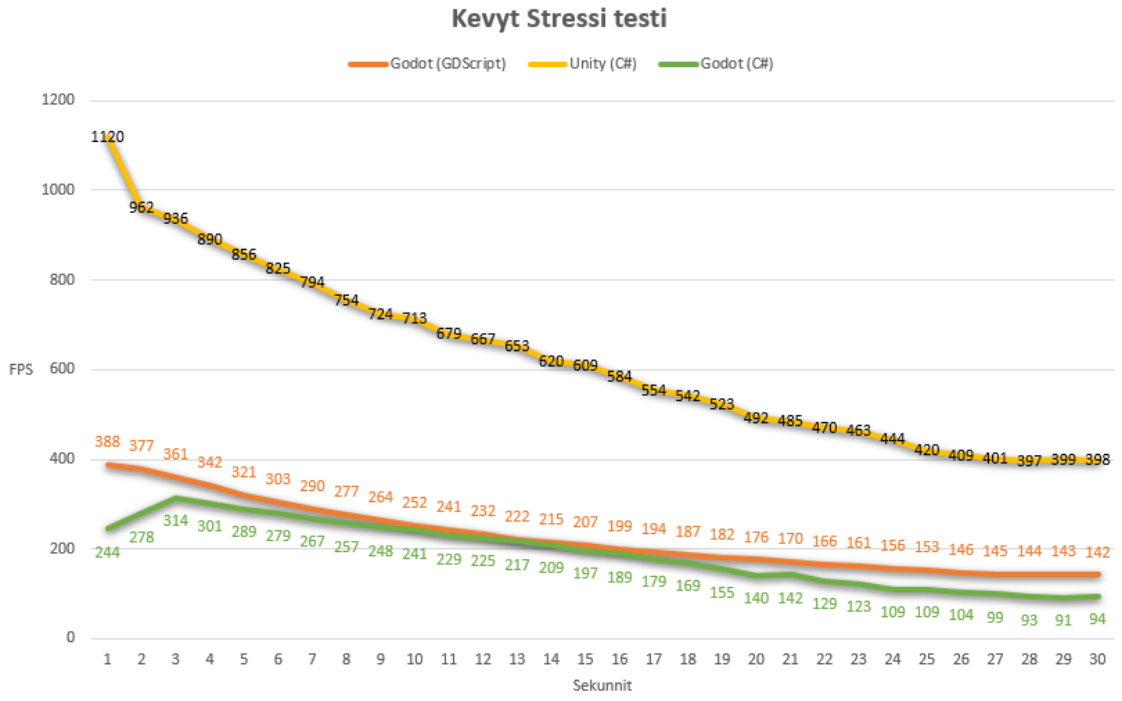
KUVA 7. Projektille olennaiset tietokoneen komponentit

5 TULOKSET JA JOHTOPÄÄTÖKSET

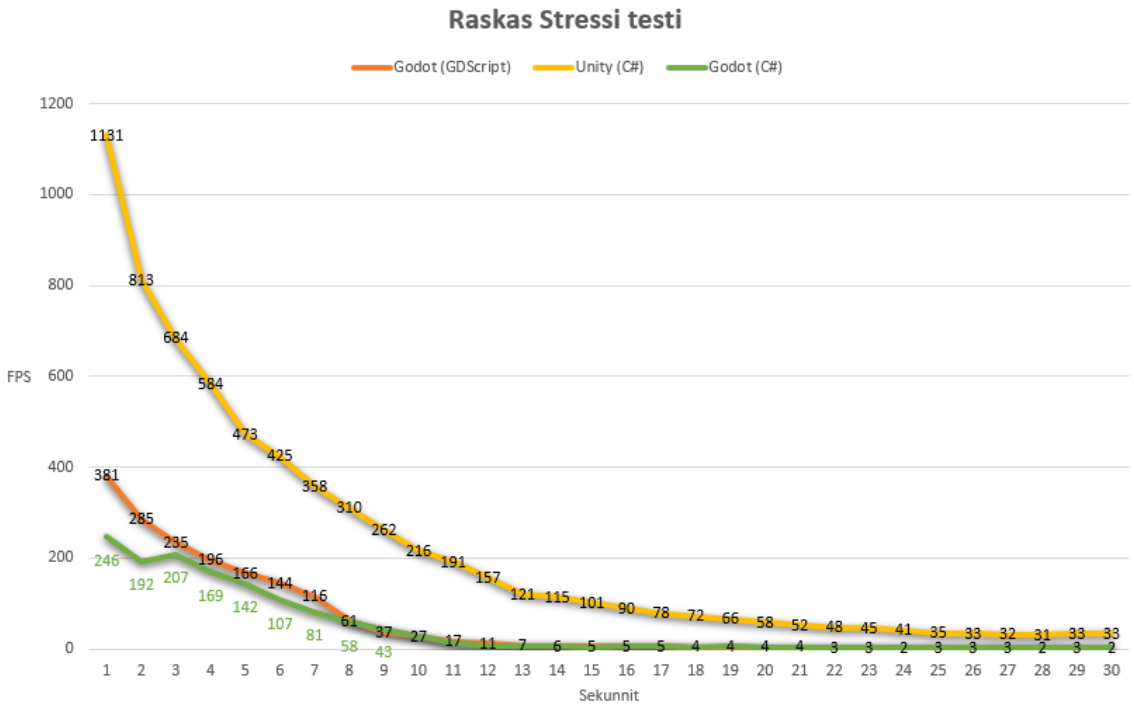
5.1 Stressitestien tulokset

Testien tulokset koottiin yhteen ja niistä luotiin kuvien 8 ja 9 excel kaaviot. Tuloksista nähdään ruudunpäivitysnopeus eli FPS stressitestien aikana ja kuinka tasainen kuormituksen lisääminen laskee sitä. Päällimmäinen tarkoitus stressitesteille on nähdä pysyykö ruudunpäivitysnopeus tarpeeksi suurena jotta pelaaminen on vielä mahdollista ja nautinnollista.

Tuloksista näkee selvästi, että Unity suoriutui molemmista testeistä paljon Godot Engineä paremmin. Stressitestin ensimmäisiä sekuntteja tutkimalla nähdään, että Unityn ruudunpäivitysnopeus on melkein kolminkertainen verrattuna Godotin tuloksiin. Molemmat pelimoottorit säilyttivät kuitenkin tasaisen pelattavuuden kevyessä testissä. Raskaan testin loppupuolella huomioitavaa on, ettei Unityn ruudunpäivitysnopeus mennyt kertaakaan alle kolmenkymmenen, joka on monen vanhemman pelikonsolin maksimimäärä. Godot Enginen ruudunpäivitysnopeus laski taas reilusti alle kolmenkymmenen, joka tekee pelaamisesta miltei mahdotonta kuvan pätkimisen takia. Unity on siis selvästi Godottia tehokkaampi vaihtoehto 3D-pelikehitykseen, varsinkin jos tarkoituksena on tehdä raskas videopeli.



KUVA 8. Kevyen stressitestin tuloskaavio



KUVA 9. Raskaan stressitestin tuloskaavio

5.2 Profiloija

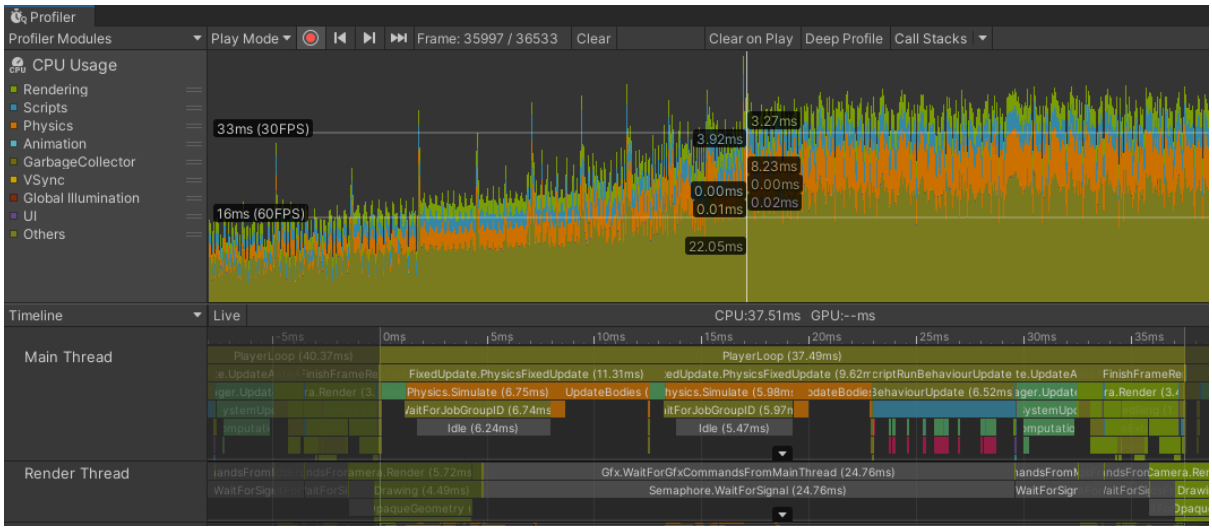
Videopelien kehityksessä yksi tärkeä työkalu on profiloija (profiler), sen avulla peliohjelmoija voi tarkastella ruudunpäivitysnopeutta, prosessorin ja näytönohjaimen suoritusaikoja sekä muita suorituskykyyn vaikuttavia tekijöitä. Näiden tietojen avulla voidaan löytää suorituskykyyn liittyvien ongelmien syyt ja optimoida videopeli. (Lark Editorial Team 27.6.2024).

Unityyn sekä Godottiin on molempiin sisäänrakennettu profiloija. Profiloijan käyttötarkoitus on yleisesti videopelin suoritustehon optimointi, mutta tässä opinnäytetyössä profiloijaa tullaan käyttämään pelkästään ongelmakohtien löytämiseen ja niiden tutkimiseen. Ongelmakohtia ei siis ole tarkoitus korjata.

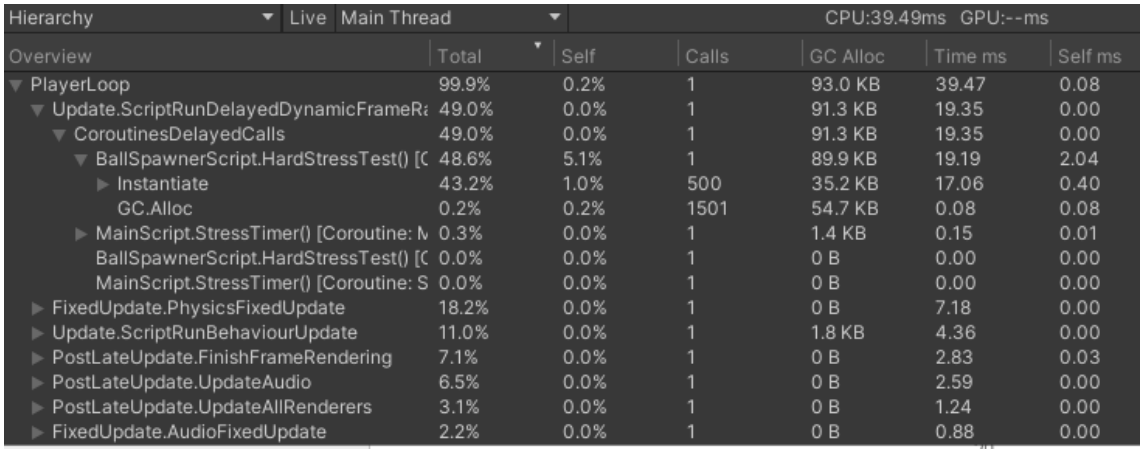
Kuva 10 on kuvankaappaus Unityn profiloijan näkymästä stressitestin loppuosassa. Kuvasta näkee kuinka monta millisekuntia yhteen ruudunpäivitykseen kuluu aikaa. Tavoiteajat näkyvät kuvan vasemmassa reunassa ja ne kertovat, että saavuttaakseen yli 60 FPS on ruudunpäivitykseen kuluvan ajan oltava alle 16 millisekuntia tai 30 FPS saavuttamiseen alle 33 millisekuntia. Profiloijassa vasemmalla näkyvät värikkäät piikit ovat stressitestin sekunnin välein tapahtuvat uusien 500 pallon synnytykset, jotka selvästi kuormittavat prosessoria. Kuvassa 11 näkyy

tarkemmin tällaisen kuormituspiikin suorittamisajat eriteltynä. Kuvasta nähdään kuinka pelkkä pallojen synnyttäminen vie prosessorilta 17 millisekuntia.

Kuvassa 10 profiloijan alareunan näkymässä on tarkastelussa yksittäinen ruudunpäivitys, joka tapahtui viimeisen pallojen synnytys kierroksen jälkeen. Tästä nähdään kuinka paljon aikaa prosessorilla ja näytönohjaimella kului eri tehtävien suorittamiseen. Main Thread osiossa näkyvät värikkäät liuskat ovat prosessorin tehtävät ja Render Thread osiossa näytönohjaimen tehtävät. Prosessorilla kului kyseisen ruudun suorittamiseen yli 37 millisekuntia pääosin fysiikkamoottoripäivitysten takia, kun taas näytönohjain suoritti omat tehtävänsä alle 6 millisekunnissa ja joutui odottamaan prosessoria noin 30 millisekuntia. Tästä nähdään siis selvästi, että Unityn toteutuksessa suorituskyvyn pullonkaulana on prosessori eikä näytönohjain.



KUVA 10. Unityn profiler näkymä raskaan stressitestin loppuosasta

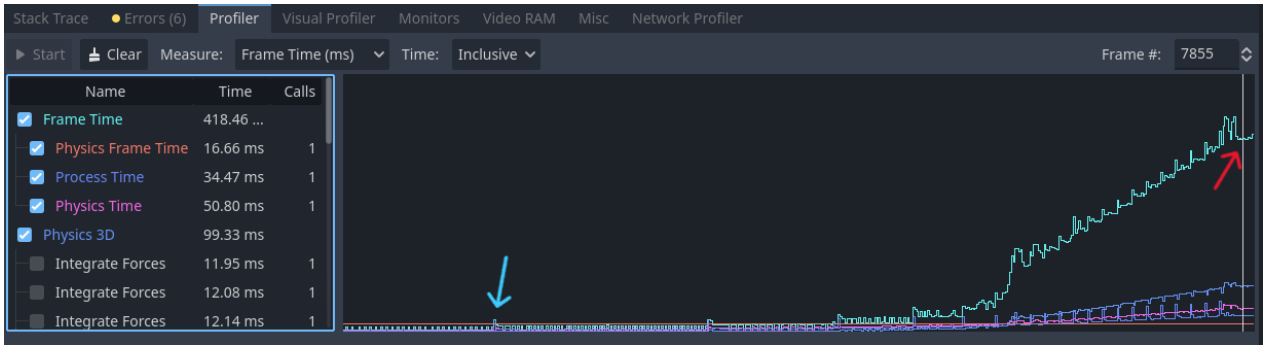


Overview	Total	Self	Calls	GC Alloc	Time ms	Self ms
PlayerLoop	99.9%	0.2%	1	93.0 KB	39.47	0.08
Update.ScriptRunDelayedDynamicFrameRa	49.0%	0.0%	1	91.3 KB	19.35	0.00
CoroutinesDelayedCalls	49.0%	0.0%	1	91.3 KB	19.35	0.00
BallSpawnerScript.HardStressTest() [C	48.6%	5.1%	1	89.9 KB	19.19	2.04
Instantiate	43.2%	1.0%	500	35.2 KB	17.06	0.40
GC.Alloc	0.2%	0.2%	1501	54.7 KB	0.08	0.08
MainScript.StressTimer() [Coroutine: M	0.3%	0.0%	1	1.4 KB	0.15	0.01
BallSpawnerScript.HardStressTest() [C	0.0%	0.0%	1	0 B	0.00	0.00
MainScript.StressTimer() [Coroutine: S	0.0%	0.0%	1	0 B	0.00	0.00
FixedUpdate.PhysicsFixedUpdate	18.2%	0.0%	1	0 B	7.18	0.00
Update.ScriptRunBehaviourUpdate	11.0%	0.0%	1	1.8 KB	4.36	0.00
PostLateUpdate.FinishFrameRendering	7.1%	0.0%	1	0 B	2.83	0.03
PostLateUpdate.UpdateAudio	6.5%	0.0%	1	0 B	2.59	0.00
PostLateUpdate.UpdateAllRenderers	3.1%	0.0%	1	0 B	1.24	0.00
FixedUpdate.AudioFixedUpdate	2.2%	0.0%	1	0 B	0.88	0.00

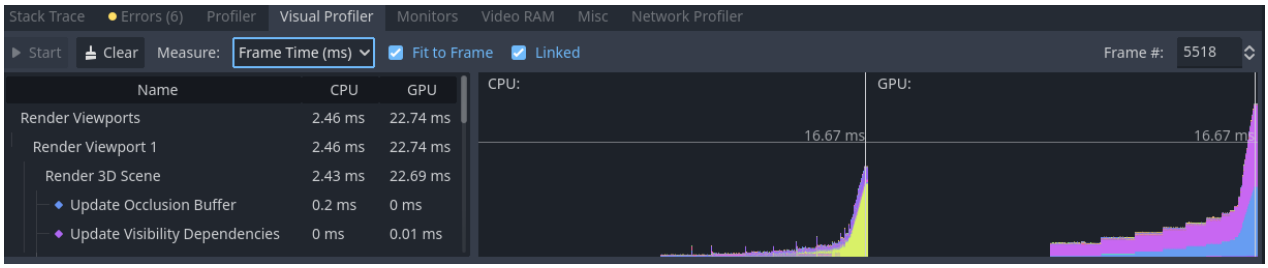
KUVA 11. Unity profilerin hierarkianäkymä stressitestin kuormituspiikistä

Kuva 12 on kuvankaappaus Godotin profiloijasta, jossa näkyy koko raskaan stressitestin profilointi. Profiloija ei ole mielestäni yhtä kattava, kuin Unityn profiloija eikä siihen löytynyt yhtä kattavaa ohjedokumentointia. Kuvasta voidaan kuitenkin nähdä samoja asioita, kuin Unityn profiloijasta. Kuvan alareunassa näkyvä oranssi tasainen viiva on fysiikkapäivityksille varattu aika eli 16,66 millisekuntia ja tavoittaakseen yli 60 FPS tulisi ruudunpäivitykseen käytettävän ajan pysyä sen alapuolella. Kuvasta on kuitenkin nähtävissä, että ruudunpäivitykseen käytetty aika eli vaaleansininen viiva kasvaa stressitestin edetessä paljon yli optimaalisen ajan. Vaaleansininen nuoli osoittaa ensimmäistä kuormituspiikkiä, joka on se hetki kun stressitesti käynnistyi ja ensimmäiset 500 palloa synnytettiin. Punainen nuoli osoittaa ruutuun, jota kuvassa tarkastellaan ja tulokset näkyvät kuvan vasemmassa laidassa. Yhteen ruudunpäivitykseen meni aikaa yli 418 millisekuntia, joka selittää Godotin huonot tulokset raskaassa stressitestissä.

Godotin profiloijan ohjemateriaalin ja esimerkkien puutteesta johtuen en osaa tulkita tuloksia ja selvittää syytä miksi ruudunpäivitykseen käytettävä aika on niin suuri. Godotissa on myös visuaalinen profiloija, jonka tulokset raskaasta stressitestistä näkyvät kuvassa 13. Visuaalisen profiloijan tulosten perusteella suorituskyvyn ongelmat johtuvat mahdollisesti näytönohjaimesta, sillä näytönohjaimella kului yhteen ruudunpäivitykseen yli 22 millisekuntia kun taas prosessorilla aikaa kului 2,4 millisekuntia. Godotin visuaalisen profiloijan käyttöön ja tulkintaan on kuitenkin vielä huonommin ohjeita ja esimerkkejä kuin perus profiloijan, joten en osaa tehdä lopullista johtopäätöstä siitä mitä nuo tulokset tarkoittavat ja johtuuko Godotin huono suoritusteho näyönohjaimesta, prosessorista vai molemmista.



KUVA 12. Godot profilerin näkymä raskaasta stressitestistä



KUVA 13. Godot visuaalisen profilerin näkymä raskaasta stressitestistä

Profiloijien tuloksia tarkastellessa on selvää, että molempien Unityn sekä Godotin toteutuksien yhtenä suurena ongelmana suoritustehossa on raskas ja huonosti optimoitu koodi liittyen pallojen synnyttämiseen sekä fysiikkamoottorin työmäärä. En ole aiemmin käyttänyt kummankaan pelimoottorin profilereita, mutta nyt lyhyen tutustumisen jälkeen valitsisin paljon mielummin Unityn profiloijan. Sen käyttöön löytyi paljon ohjeita ja esimerkkejä ja tulokset olivat mielestäni paljon kattavampia ja niitä oli helpompi tulkita.

5.3 Kustannukset

Kustannuksia vertaillessa Godot on selvästi parempi vaihtoehto, sillä se on täysin ilmainen eikä rajoita käyttäjää juuri lainkaan. Godot on julkaistu MIT-lisenssin alaisena joten sitä on ilmaista käyttää, muokata sekä kopioida kunhan muistaa jakaa tekijänoikeusilmoituksen sekä lisenssilausunnon pelin jakamisen yhteydessä. (Godot 2024.)

Unityllä on neljä eri maksusuunnitelmaa, henkilökohtainen, ammattilainen, yritys sekä teollisuus. Unity henkilökohtainen on ilmainen harrastelijoille, yksittäisille ohjelmoijille sekä pienille yrityksille, jos Unityn avulla saadut tulot ovat tarpeeksi alhaiset. Muiden suunnitelmien hintoihin vaikuttaa se kuinka monta pelinkehittäjää yrityksellä on sekä yrityksen viimeisen vuoden tulot. (Unity 2024.) Vuoden 2023 syyskuussa Unity ilmoitti muuttavansa maksusuunnitelmiaan kehittäjämäärä perusteisesta maksusta latausmäärään perustuviin maksuihin. Tämä olisi tarkoittanut, että Unityä käyttävä yritys olisi joutunut maksamaan rojalteja jokaisesta pelin lataus kerrasta. Ilmoitus muutoksesta tuli yllättäen ilman sopivaa varoitusaikaa ja monet pelinkehittäjät eivät olleet tyytyväisiä tähän päätökseen. Muutos pakottaisi heidät valitsemaan, joko aikaa ja rahaa vievän pelimoottorin vaihdon tai rojaltien maksamisen väliltä. Suurta keskustelua aiheutti myös lataus rojaltien mahdollinen väärinkäyttö pelinkehittäjiä kohtaan. Lataamalla tietyn pelin yhä uudelleen ja uudelleen, joukko ihmisiä olisi voinut kerryttää kyseisen pelin kehittäjälle suuren laskun. Valtavan negatiivisen palautteen ja boikotointien takia, Unity ilmoitti syyskuussa 2024 peruvansa

latausmääriin perustuvat maksut ja palaavansa vanhoihin maksusuunnitelmiin. Samassa ilmoituksessa kuitenkin kerrottiin vuoden 2025 alussa tulevista hintojen korotuksista koskien ammattilais sekä yritys maksusuunnitelmia. (Parrish 2024.)

Tällä hetkellä molemmat pelimoottorit ovat minulle ilmaisia, koska olen vain harrastelija, mutta tulevaisuutta ajatellessa pidän Godottia paljon parempana ja varmempana vaihtoehtona hintavertailussa. Godot on täysin ilmainen nyt ja uskon, että tulee aina olemaan. Unity on taas luonut epäluottamusta itsensä ja pelinkehittäjien välille viimeisien vuosien aikana. Minusta on selvää, että Unityn tärkein tavoite on tehdä mahdollisimman paljon rahaa ja he ovat selvästi valmiita tekemään suuriakin muutoksi ilman heidän käyttäjiensä mielipidettä.

5.4 Ominaisuudet ja mukavuudet

Vertaillessa pelimoottoreiden välisiä eroja ominaisuuksissa ja mukavuuksissa perustuvat tulokset pakostakin henkilökohtaisiin mielipiteisiini ja mieltymyksiini. Kaikissa pelimoottoreiden osa-alueissa en myöskään huomannut eroja. Esimerkiksi animointi- ja partikkelityökalut olivat mielestäni tekemissäni simppeleissä animaatioissa ja efekteissä hyvin samanlaisia käyttää, ja siten keskenään yhtä hyviä. Äänimaailman toteutus oli myös yhtä helppoa kummallakin pelimoottorilla.

Ohjelmointi pelimoottoreiden välillä erosi paljon ohjelmointikielestä riippuen. Henkilökohtaisesti pidin Godotin GDScriptillä ohjelmoimisesta eniten, koska mielestäni käyttöliittymän sisäänrakennettu ohjelmointiympäristö oli mukava ominaisuus ja selkeytti ohjelmointia. Toiseksi mielekkäintä oli Unityn ohjelmointi käyttäen ulkoisena ohjelmointiympäristönä Microsoftin Visual Studio 2022:sta ja C#–ohjelmointikieltä. C#–ohjelmointikielen ja Visual Studio Code ohjelmointiympäristön käyttö Godotin toisessa osuudessa oli mielestäni ohjelmoinnissa tympeintä. Tulokseen vaikuttaa varmasti se että tämä projekti oli ensimmäinen kerta kun käytän Godotilla ohjelmoimiseen C#–ohjelmointikieltä. GDScript–ohjelmointikielenä on itselleni mieluisampi kuin C#, mutta uskon sen johtuvan siitä, että GDScript on tunnetusti helpompi kieli oppia ja molempien ohjelmointikielien käyttö on minulle vielä suhteellisen uutta pelikehityksessä.

Selvänä erona pelimoottoreiden ominaisuuksissa on aikaisemmin tuloksissa käytetyt profiloijat. Unityn profiloija oli Godotin omaan verrattuna mielestäni todella paljon parempi ja kattavampi. Uskon, että tuo ero tulee vaikuttamaan isosti pelimoottorin valintaani tulevaisuudessa.

6 POHDINTA

Opinnäytetyö ja projektien vertailu onnistui hyvin, mutta lopullista valintaa pelimoottoreiden väliltä en osaa vielä tehdä. Ajatukseni opinnäytetyötä aloittaessa oli saada selkeä vastaus siihen kumpi pelimoottori, Godot vai Unity, sopii minun tarpeisiini paremmin. Molemmat pelimoottorit tulevat todennäköisesti olemaan osana tulevaisuuden peliprojektejani. Unityä tulen käyttämään 3D-pelikehityksessä sen suorituskyvyn ja profiloijan takia. Godottia aion käyttää 2D-pelikehityksessä sen käyttöliittymän ja edullisuuden takia. Molemmista pelimoottoreista ja videopelikehityksestä opin opinnäytetyön aikana paljon uutta, ja seuraava projektiaihekkin löytyi pelien optimoinnista ja siihen tutustumisesta. Optimointi liittyy tämän opinnäytetyön aiheeseen todella läheisesti, mutta se olisi ollut aihepiirinä liian laaja ottaa mukaan tähän projektiin. Tarkoituksenani on tutustua pelien optimointiin tulevaisuudessa ja aion käyttää tätä opinnäytetyötä ja stressitestejä pohjana nuille tutkimuksille. Mahdollisesti siten saan Godotista kilpailukykyisemmän 3D-videopelikehitysalustan Unityyn verrattuna.

LÄHTEET

Bradley, Kain 2021. Game design and development. Purposes of a game world. Hakupäivä 3.5.2024. https://ecampusontario.pressbooks.pub/gamedesigndevelopmenttextbook/chapter/purposes-of-a-game-world/ .

Crichton-Stuart, Eliza 2023. Newzoo Global Games Market Report 2023: $187.7B Revenue, Growth Factors, and Industry Trends. Gam3s.gg. Hakupäivä 22.9.2024 https://gam3s.gg/news/newzoo-global-games-market-report-2023/.

Dobbin, Jolene. 14.8.2024. GPU vs CPU for Gaming: What Matters Most for PC Performance. hp. Hakupäivä 7.5.2025. https://www.hp.com/us-en/shop/tech-takes/gpu-vs-cpu-for-pc-gaming?pStoreID=newegg%2Fgb-en%2Fshop%2Flist.aspx%3Fsel?utm_source=tech_takes&utm_medium=organic&utm_campaign=share_article&utm_content=copy_link.

Dornhege, Pablo. Quickly explained: What is a game engine and why do I need it? Digital.DTHG. Hakupäivä 6.5.2025. https://digital.dthg.de/en/engine/.

Duarte, Fabio. 24.4.2025. How Many Gamers Are There? (New 2025 Statistics). Exploding Topics. Hakupäivä 5.5.2025. https://explodingtopics.com/blog/number-of-gamers.

Endoflife.date 2024. Unity. Hakupäivä 3.5.2024. https://endoflife.date/unity.

Github 2024. Godot Engine. Hakupäivä 11.4.2024. https://github.com/godotengine/godot?tab=readme-ov-file.

Godot Docs 2024. Godot release policy. Release support Timeline. Hakupäivä 11.4.2024. https://docs.godotengine.org/en/stable/about/release_policy.html#release-support-timeline.

Godot 2024. License. The engine. Hakupäivä 16.9.2024. https://godotengine.org/license/.

Gregory, Jason 2018. Game Engine Architecture. Third edition. A K Peters: CRC Press. Hakupäivä 30.5.2024. O'Reilly. Vaatii käyttöoikeuden.

Hakaniemi, Katariina. 26.3.2023. Suomessa on 50–100 e-urheilijaa, jotka elättävät itsensä pelaamalla – "Maailmanmestarimme ovat tienanneet useita miljoonia euroja". Helsingin Uutiset. Hakupäivä 5.5.2025. https://www.helsinginuutiset.fi/teemat/5764661.

Juegoadmin. 8.8.2023. Importance of Sound Design in Games. JuegoStudios. Hakupäivä 7.5.2025. https://www.juegostudio.com/blog/sound-design-in-games.

Lark Editorial Team 27.6.2024. Performance Profiling. Lark. Hakupäivä 8.5.2025. https://www.larksuite.com/en_us/topics/gaming-glossary/performance-profiling.

Neogames 2023. Tietoa toimialasta. Hakupäivä 22.9.2024. https://neogames.fi/fi/tietoa-toimialasta/ .

Parrish, Ash 2024. Unity has eliminated its controversial runtime fee. TheVerge. Hakupäivä 8.10.2024. https://www.theverge.com/2024/9/12/24242937/unity-runtime-fee-cancelled-subscription-pricing.

Serrano, Harold 2.2.2019. How does the Game Engine Loop make a game possible? Haroldserrano. Hakupäivä 6.5.2025. https://www.haroldserrano.com/blog/the-heart-of-a-game-engine-the-game-engine-loop.

Sibony, Joseph 2024. The Best Game Engines You Should Consider for 2024. Incredibuild. Hakupäivä 30.5.2024. https://www.incredibuild.com/blog/top-gaming-engines-you-should-consider .

Unity 2024. Plans and pricing. Hakupäivä 1.10.2024. https://unity.com/products.

Wayline 19.1.2024. Unity vs. Godot. Hakupäivä 7.5.2025. https://www.wayline.io/blog/unity-vs-godot.