Haaga-Helia
ammattikorkeakoulu Oy

Pelikehityksessä käytettävien avoimen ja suljetun lähdekoodin pelimoottoreiden vertailu

Mika Aho

Opinnäytetyö
Tietojenkäsittelyn koulutusohjelma
2017

Haaga-Helia ammattikorkeakoulu Oy

Tiivistelmä

28.11.2017

Tekijä(t)
Aho, Mika

Koulutusohjelma
Tietojenkäsittelyn koulutusohjelma

Opinnäytetyön otsikko

Pelikehityksessä käytettävien avoimen ja suljetun lähdekoodin pelimoottoreiden vertailu

Sivu- ja liitesivumäärä
40

Opinnäytetyön otsikko englanniksi
Comparison of open and closed source game engines in game development.

Opinnäytetyö käsittelee pelikehityksessä käytettävien pelimoottoreiden vertailua pelikehittäjän näkökulmasta. Tutkimukseen on otettu kolme avoimen ja kolme suljetun lähdekoodin pelimoottoria, joiden etuja ja haittoja pelikehitykselle vertaillaan. Aiheen valinnassa korostuu käytännönläheisyys, sillä pelikehittäjän toiminnan helpottamiseksi tutkimustiedolla on käyttöä ajantasaisena informaationa.

Opinnäytetyössä käytetään kvalitatiivista tutkimustapaa, joka mahdollistaa osallistuvan havainnoinnin pelikehittäjänä ja pelimoottoreiden testaajana sekä vertailevaa tutkimusta osoittamaan pelimoottoreiden ominaisuuksien erot. Vertailevaa tutkimusta varten on kerätty tietoa kuudesta pelimoottorista ja hyödynnetty valmiiksi kerättyä tietoa, kuten aiemmin tutkittuja aineistoja sekä opinnäytetöitä. Lisäksi on testattu ja kokeiltu pelimoottoreita myös itse. Tutkimustapojen avulla on saatu koottua lopullinen tietopaketti.

Vertailtavat avoimen lähdekoodin moottorit ovat Cocos2d-x, Godot Engine sekä Torque 3D ja suljetun lähdekoodin pelimoottorit CryEngine, Unity3D sekä Unreal Engine. Pelimoottoreita vertaillaan pelilogiikan ohjelmointikielten, fysiikkamoottorin, äänimoottorin, verkkopelimahdollisuuksien, 2D- ja 3D-ominaisuuksien sekä kustannusten näkökulmista.

Opinnäytetyön johtopäätökset antavat viitteitä siitä, mikä pelimoottori saattaisi ominaisuuksiensa vuoksi sopia parhaiten halutun lopputuloksen saavuttamiseen. Tutkimus ottaa kantaa myös siihen, onko avoimen lähdekoodin pelimoottorilla yhtä kattava käytettävyys kuin suljetun lähdekoodin pelimoottorissa.

Asiasanat

peliala, pelikehitys, pelimoottori, avoin lähdekoodi, suljettu lähdekoodi, vertailu

Sisällys

1 Johdanto

Tämä opinnäytetyö käsittelee pelikehityksessä käytettävien pelimoottoreiden vertailua pelikehittäjän näkökulmasta. Tutkimukseen on otettu kolme avoimen ja kolme suljetun lähdekoodin pelimoottoria, joiden etuja ja haittoja pelikehitykselle vertaillaan. Työllä ei ole varsinaista toimeksiantajaa, mutta aiheen valinnan käynnisti oma osuuteni vapaa-ajalla tapahtuvassa pelikehitysprojektissa. Projektin tarkoituksena on tuottaa mobiilipeli, jonka pelimoottoriksi valikoitui avoimen lähdekoodin pelimoottori. Aiheen valinnassa korostui myös käytännönläheisyys; aloittelevan tai työelämässä kauemminkin toimineen pelikehittäjän toiminnan helpottamiseksi tutkimustiedolla on käyttöä ajantasaisena informaatiopakettina.

Aloittelevan sekä kokeneemman pelikehittäjän on aiempaa helpompaa aloittaa pelikehitys, sillä tarjolla on toinen toistaan loistavampia pelimoottoreita, joista osa on avoimeen lähdekoodiin perustuvia, usein maksuvapaita pelimoottoreita ja osa suljetun lähdekoodin pelimoottoreita. Suljettujen lähdekoodienkin pelimoottoreissa on yleensä ilmainen peruskäyttö ja maksut syntyvät vasta kaupallistamisvaiheessa. Haastavinta on valinnanvaikeus. Valinnanvaikeuteen ja kysymykseen valinnasta avoimen ja suljetun lähdekoodin välillä saadaan apua opinnäytetyön vertailun tuloksena syntyvästä informaatiopakeista.

Vertailevan tutkimuksen tavoitteena on saada aikaan pelikehittäjälle tiivis yhdensivun mittainen ominaisuustaulukko, josta ilmenee kunkin vertailussa olevan avoimen ja suljetun lähdekoodin pelimoottorin ominaisuudet yhdellä silmäyksellä. Yhdessä johtopäätösten kanssa opinnäytetyöstä saa tietopaketin siitä, mikä pelimoottori soveltuu ominaisuuksiensa vuoksi parhaiten halutun lopputuloksen saavuttamiseen pelikehityksen sujuvuuden kannalta. Vertailun yhteenveto ottaa myös kantaa siihen, onko avoimen lähdekoodin pelimoottorilla yhtä kattava käytettävyys kuin suljetun lähdekoodin pelimoottorissa vai tuoko suljetun lähdekoodin pelimoottori toimintoihin lisää ominaisuuksia. Opinnäytetyön vertailun pelimoottoreita vertaillaan pelilogiikan ohjelmointikielten, fysiikkamoottorin, äänimoottorin, verkkopelimahdollisuuksien, 2D- ja 3D-ominaisuuksien sekä varsinkin kustannusten näkökulmista.

Vertailtavat avoimen lähdekoodin moottorit ovat Chukongin kehittämä Cocos2d-x, Godot Engine Communityn kehittämä Godot Engine sekä Garage Gamesin kehittämä Torque 3D. Vertailussa olevat suljetun lähdekoodin pelimoottorit ovat Crytekin kehittämä CryEngine, Unity Technologiesin kehittämä Unity3D ja Epic Gamesin kehittämä Unreal Engine. Pelimoottorit on valikoitu vertailukohteiksi niiden saatavuuden ja suosion takia. Vertailussa olevien avoimen lähdekoodin sekä suljetun lähdekoodin pelimoottorien käyttöönotto on helppoa ja maksullistenkin moottoreiden peruskäyttö on usein ilmaista.

2 Keskeiset käsitteet

Asset	Pelin lisäosa, joka voi olla esimerkiksi musiikkia, grafiikkaa tai skripti.
Cross-platform	Laitteisto- ja käyttöjärjestelmäriippumaton ohjelma
Framework	Ohjelmistokehys. Tarjoaa valmiita komponentteja, joiden on tarkoitus nopeuttaa rakennettavan ohjelmiston valmistumista.
GitHub	Versionhallintapalvelu
HLAPI-toiminto	Unity-pelimoottorin korkean tason ohjelmistorajapinta, joka on tarkoitettu erityisesti moninpeliominaisuuksien rakentamiseen.
LLAPI-toiminto	Unity-pelimoottorin matalan tason ohjelmistorajapinta, joka tarkoitettu edistyneempien verkko-ominaisuuksien rakentamiseen.
MIT-lisenssi	Vapaan lähdekoodin lisenssi, joka sallii ohjelman käytön, mikäli tekijänoikeustiedot säilytetään.
Olio-ohjelmointi	Ohjelmointia, jossa perinteisesti monta oliota käsittelee yksittäisiä asioita, mutta oliot yhdessä suorittavat samaa ohjelmaa.
Renderöinti	Digitaalisen tiedon muuttaminen näytölle sopivaan esitysmuotoon
Script, Scripting	Komentosarja, jonka suorittaa toinen ohjelma.
Software	Ohjelmisto
Streaming	Suoratoisto. Käytetään esimerkiksi videoissa sekä musiikissa.
Syntaksi	Ohjelmointikielen rakenne
TCP	Standardoitu tiedonsiirtoprotokolla, joka toimii yhdessä internet-protokollan kanssa.
UDP	Tiedonsiirtoprotokolla, joka toimii yhdessä internet-protokollan kanssa, mutta ei sisällä esimerkiksi virheenkorjausta.
Wrapper, kääreluokka	Sisältää toisen luokan tai komponentin toiminnallisuuksia

3 Opinnäytetyön tavoitteet

Opinnäytetyön empiirisen osan tuotoksena syntyy vertailu, josta koostetaan tiivis informaatiokokonaisuus pelikehittäjälle pelin tekemiseen käytettävistä pelimoottorivaihtoehdoista ja tärkeimmät ominaisuudet valitsemisen tueksi.

Opinnäytetyöni tietoperusta tukee tavoitetta saada aikaan pelikehittäjälle tiivis tietopaketti pelin tekemiseen käytettävien avoimen ja suljetun lähdekoodin pelimoottoreiden valinnan kriteereistä. Tutkimuksen avulla yritetään selvittää erilaisten pelimoottorien haitat ja hyödyt pelikehittäjälle sekä niiden sopivuudet kuhunkin tekemiseen. Vertailun yksi näkökulma on, että saako avoimen lähdekoodin pelimoottorilla aikaan saman kuin suljetun lähdekoodin pelimoottorilla.

Tämän opinnäytetyön tietoperusta on kasattu pääosin pelikehityksestä, peliteollisuudesta ja erilaisista pelimoottoreista kirjoitetuista teorioista ja kirjallisuudesta. Lisäksi tutkitaan ja vertaillaan pelimoottoreita yleisellä tasolla, jotta päästään selvittämään minkä pelimoottorin käyttöönotolla pelikehittäjä pääsisi parhaimpaan tulokseen riippuen halutusta lopputuloksesta.

3.1 Tutkimuskysymys

Opinnäytetyön tarkoitus on vastata seuraaviin kysymyksiin:

- Millaisia eroja on avoimeen lähdekoodiin ja suljettuun lähdekoodiin perustuvissa pelimoottoreissa?
- Mitkä ovat tärkeimmät pelikehitykseen valittavan pelimoottorin valintaan liittyvät tekijät?
- Kumpaa pelimoottorityyppiä pelikehittäjän kannattaa pelikehityksessä käyttää?

3.2 Tutkimusmenetelmät

Tämän opinnäytetyön empiirisessä osuudessa käytetään vertailevaa tutkimustapaa. Tämä tutkimustapa jaetaan yleensä kahteen kategoriaan: ryhmäero- tai korrelaatiotutkimukseen.

Ryhmäerojen vertailussa laajemman populaation muuttujista mitataan ominaisuuksia, jolloin saadaan selvitettyä tutkittavan ryhmän eroavaisuudet ja yhtäläisyydet sekä tekijät, jotka aiheuttavat eroavaisuuksia ryhmän sisällä. (Kamk 2017.)

Korrelaatiotutkimus taas yrittää löytää yhtäläisyyksiä tutkimalla kohdejoukkoa kokonaisuutena, jossa tutkittavien joukkoon otetaan mukaan selittäviä muuttujia. Nämä selittävät

muuttujat auttavat tutkijaa löytämään yhteyksiä ja niistä löytyy usein myös mahdollinen syy selitettäville muuttujille sekä seurauksille. (Kamk 2017.)

Vertailevassa tutkimuksessa korostuvat seuraavat asiat: On mietittävä miksi valittuja tutkimuskohteita vertaillaan ja mitä vertailussa halutaan selvittää. Sen lisäksi tulee pohtia, että mitkä koehenkilöt tai tutkittavat asiat valitsemalla tutkimuksen tavoite saavutetaan. Vertailevat tutkimukset jaetaan karkeasti tutkimustavoitteiden mukaan joko teoriaa testaaviin, kehittäviin tai kuvaileviin tutkimuksiin. Usein vertailevaa tutkimusta tehdäänkin yhdistämällä kuvaileva ja selittävä tutkimus, kuten tässäkin opinnäytetyössä. (Kamk 2017.)

Opinnäytetyössä käytetään sekä laadullista eli kvalitatiivista tutkimustapaa, joka mahdollistaa myös oman osallistuvan havainnoinnin pelikehittäjänä ja pelimoottoreiden testaajana sekä vertailevaa tutkimusta osoittamaan pelimoottoreiden ominaisuuksien erot informatiivisesti. Näin saadaan koottua lopullinen vertailu eli informaatiopaketti. Osallistuva ja kohdistettu havainnointi tapahtuu aina tutkimuskohteeksi valituissa tilanteissa, tapahtumissa tai asioissa. (Grönfors 1985, 100–102.)

Kvalitatiivisessa työmenetelmässä jaotellaan havainnointitavat tutkijan oman osallistumisen kautta: havainnoidessaan tutkija joko havainnoi ilman osallistumista, osallistuu itse toimintaan tai tekee piilohavainnointia. (Grönfors 1985, 87–98.)

Tämän kvalitatiivisen opinnäytetyön tekijänä havainnoin itse osallistumalla eli testaan ja käytän tutkittavia pelimoottoreita myös itse, samalla kun tutkin muiden pelikehittäjien mielipiteitä ja käyttökokemuksia.

3.3 Tutkimus

Kuten vertailun teoriaa käsittelevä sivusto (Uiah 2007) kertoo, on vertaileva tutkimus helppo suunnitella: tutkitaan tutkittavan aineiston tapauksia tai yksilöitä, jotka kuuluvat samaan kategoriaan, mutta eroavat kuitenkin jollakin tavalla toisistaan. Vertailun aikana tarkastellaan yhtäläisyyksiä ja eroavaisuuksia. Kun niitä löytyy, tutkitaan esimerkiksi eroavaisuuksien suhdetta toisiinsa. Kun valittujen tutkittavien ominaisuuksien erot ja yhtäläisyydet on saatu selville, päästään tuottamaan vertaileva taulukko, kuten tämän opinnäytetyön ominaisuustaulukko.

Vertailevaa tutkimusta varten kerätään tietoa kuudesta eri pelimoottorista ja hyödynnetään osin jo valmiiksikin kerättyä tietoa, kuten aiemmin koottua materiaalia ja tutkittuja aineistoja sekä opinnäytetöitä. Testaan ja kokeilen pelimoottoreita myös itse.

4 Aikaisemmat tutkimukset aiheesta

Tutkimuksia, joissa vertaillaan avoimen lähdekoodin pelimoottoreita nimenomaan suljetun lähdekoodin pelimoottoreihin, ei opinnäytetyön kirjoitushetkellä tuntunut löytyvän. Useistakin tässä työssä käsitellyistä pelimoottoreista löytyy vertailuja erilaisilta keskustelufoorumeilta sekä esimerkiksi muista opinnäytetöistä. Niiden näkökulmat ovat kuitenkin usein erilaisten ominaisuuksien vertailua, kuten parhaat graafiset ominaisuudet, mutta lisävertailu tiettyjen avoimien ja suljettujen lähdekoodien pelimoottoreiden välillä puuttuu.

Avoimen ja suljetun lähdekoodin pelimoottoreiden keskinäisen vertailun puuttuminen ei sinänsä yllätä, koska pelikehittäjien pelimoottorin valintaa ohjaa usein ohjelmointikieli tai kenties pelimoottorin tuttuus itselle, joten valintaa ei välttämättä edes kyseenalaisteta. Aloittava pelikehittäjä valitsee usein jonkun tutun suositteleman pelimoottorin, koska tietämys on eri pelimoottorien hyödyistä ja haitoista on puutteellinen.

5 Pelialan kehitys

Tässä luvussa keskitytään pelialan historiaan ja sen kehityskaareen sellaiseksi, millaisena se nykypäivänä meille näyttäytyy.

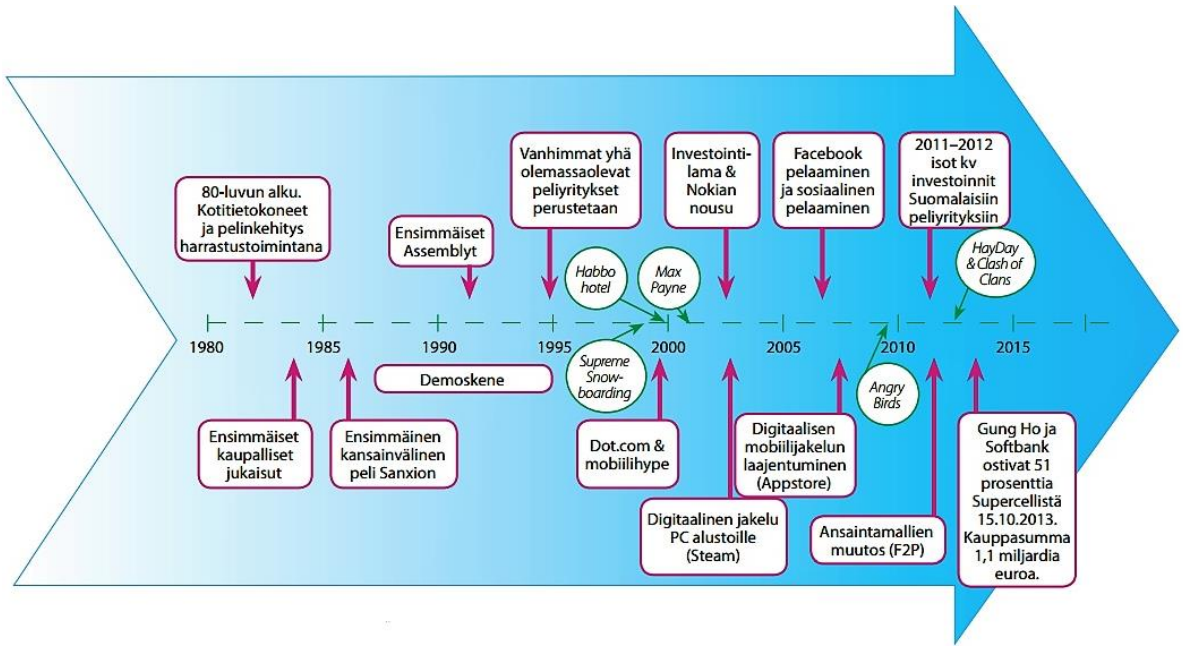
5.1 Peliteollisuuden nykytila ja historia

Tekes on tehnyt laajan raportin (2013, 7), joka sisältää samalla selvityksen suomalaisen pelitoimialan kehityksestä 1980-luvulta nykypäivään (kuva 1). Tällä aikavälillä peliteollisuus on noussut Suomen sisältötuotannollisen teollisuuden yhdeksi merkittävimmäksi vientitoimialaksi ja kasvu on syntynyt nimenomaan viihdepeleistä.

Tekes (2013, 8-9) on kuvannut Suomen pelitoimialan kehitysvaiheet eri vuosikymmeninä näin:

- 1982 – 1991
    - Suomeen muodostuu oma pelikulttuuri
- 1992 – 1997
    - pelinkehitysharrastajat järjestäytyvät pelinkehittäjäryhmiksi - käynnistyy ammattimainen pelikehitys sekä yritystoiminta
- 1997 – 2001
    - ensimmäiset suuret investoinnit ja mobiilipelikokeilut sekä ensimmäiset pelihitit konsoleille (Supreme Snowboading, Max Payne I sekä II)

- 2002 – 2005
    - taantuma investoinneissa, Nokian nousu (mm. N-gage) ja mobiilikehityksen vahva tuleminen
- 2005 – 2007
    - digitaaliset jakelukanavat syntyvät (PC Steam) ja ne aiheuttavat arvoketjumurroksen PC-jakeluun
- 2008 – 2010
    - digitaalinen mobiilijakelu yleistyy (Apple/AppStore), sosiaalinen pelaaminen alkaa, Facebook, arvoketjumurros myös konsolijakelussa (Playstation Network, Xbox Live), Angry Birds -ilmiö syntyy, samoin peliteollisuuden ja viihdeteollisuuden integraatio
- 2011 – 2012
    - ensimmäiset merkittävän suuret kansainväliset investoinnit pelitoimialalle (81,3 miljoonaa USD), pelitoimialan start up -buumi alkaa
- 2012 –
    - ansaintamallin murros johtuen digitaalisen jakelun ja mobiilialustojen yleistymisestä (free-to-play; ilmiöt HayDay ja Clash of Clans)



Kuva 1 Pelitoimialan kehitys Suomessa aikajanalla vuosina 1982–2013 (Tekes 2013, 8-9)

5.2 Peliteollisuuden tulevaisuus

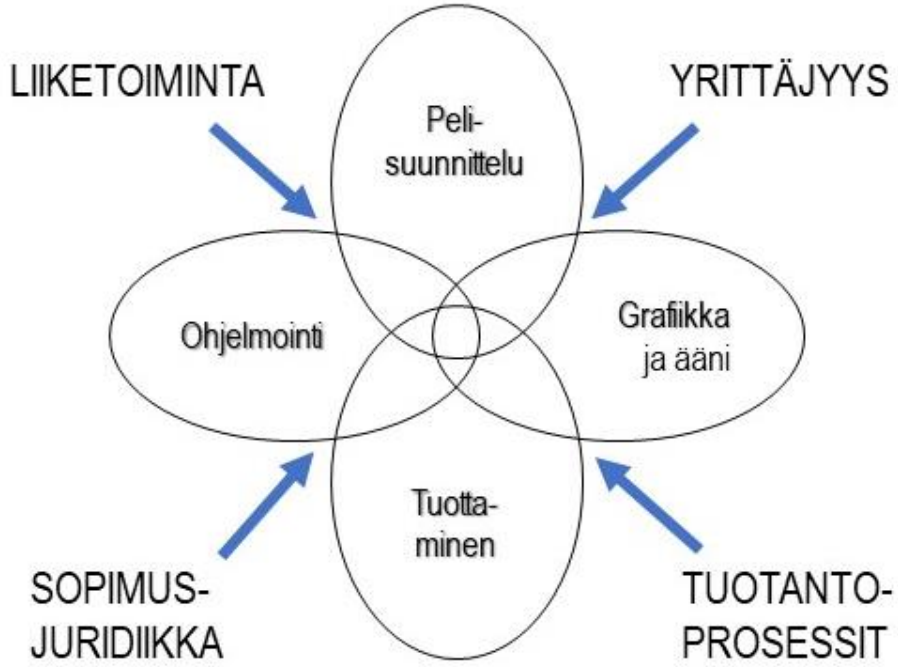
Tekesin raportissa (2013, 44) Suomen Pelinkehittäjät ry arvioi silloisen, vuoden 2012 lopussa tehdyn ennusteen mukaan, että koko toimiala saavuttaa 1,5 miljardin liikevaihdon vuonna 2020.

Voittoa tavoittelematon pelialan yhdistys Neogames, on taho, joka pyrkii koordinoimaan, ohjaamaan ja tukemaan suomalaista peliteollisuutta kasvuun ja kehitykseen. Neogames yhdistää pelialan eri sektorien toimijat ja ajaa kaikkien yhteisiä etuja (Neogames 2017).

Neogames-yhdistyksen tulevaisuusnäkökulma pelitoimialle on sen työllisyyttä edistävä luonne. Yhdistys arvioi vuodelle 2017 seuraavasti: pelialalla on arviolta n. 280 avoinna olevaa työpaikkaa, mutta kokeneiden ammattilaisten puute on suurin este alan kasvulle (Hiltunen, Latva & Kaleva 2016, 18).

5.3 Mistä peliala koostuu

Peliala käsittää laajan kokonaisuuden eri osa-alueita: liiketoiminnan, yrittäjyyden, erilaiset tuotantoprosessit sekä sopimusjuridiikan (kuva 2). Pelisuunnittelu on osa pelikehitystä. Pelikehityksessä tarvitaan lisäksi grafiikkaa ja ääntä, tuottamista sekä ohjelmointia. (Manninen 2007, 11.)



Kuva 2 Pelisuunnittelun asemoituminen koko pelikehityksen prosessiin (Manninen 2007, 11)

6 Pelimoottoreiden esittely

Tässä luvussa tehdään katsaus pelimoottoreihin: mitä ne ovat, miten ne ovat syntyneet, mikä on niiden rooli pelikehityksessä ja miten pelimoottori yleensä valitaan.

6.1 Mikä on pelimoottori

Pelimoottorit ovat kehitystyökaluina käytettyjä ohjelmistoja tai ohjelmistokehyksiä (Software Framework), jotka on tarkoitettu säästämään aikaa pelikehityksessä tarjoamalla valmiita perustoiminnallisuuksia, kuten grafiikan renderöinnin, fysiikkamoottorin, äänet, komentosarjat (Scripting), animaatiot, tekoälyn, tarvittavat ominaisuudet verkkoon sekä muistinhallintaa (Enger 2013).

Pelimoottorit sisältävät ns. valmiita työkaluja, joiden avulla pelinkehittäjät kokoavat pelin, yleensä graafisessa ympäristössä, eikä pelinkehittäjän itse tarvitse tehdä kaikkea alusta saakka (Ward 2008). Pelimoottori on toisin sanoen yleismaailmallinen runko, jossa on sisäänrakennettuna valmiita komponentteja työskentelyn helpottamiseksi sekä sen nopeuttamiseksi (Watkins 2011, 4).

Pelikehityksessä pelimoottoriin kasataan pelin kaikki objektit, joita sanotaan asseteiksi. Assetteja ovat mm. pelien hahmot, kentät sekä kaikki lähdekoodin palaset, joita kutsutaan scripteiksi. Kun kaikki pelin assetit kasataan pelimoottorissa yhteen ja liitetään pelin toiminnallisuuksiin sekä graafisiin komponentteihin, saadaan aikaan valmis peli. (Game Objects 2017).

Pelin objekteihin pystytään lisäämään erilaisia ominaisuuksia, kuten vaikka törmäyksentunnistus ja fysiikkamallinnus. Nämä lisätyt ominaisuudet saavat scriptien avulla pelin assetit pyörimään ja toimiaan keskenään halutulla tavalla (Unity. Documentation 2017).

Pelimoottoreiden pelilogiikan rakentamiseen käytettävät erilaiset skriptikielet, riippumatta kielestä, on tarkoitettu helpoksi omaksua, jolloin sitä voi kirjoittaa myös henkilö jolla ei ole kokemusta ohjelmoinnista. Tällöin voi joitakin pelissä tapahtuvia asioita ohjelmoida vaikkapa tasosuunnittelija nopeasti ja varsinainen pelimekaniikan toteutus jää ohjelmoijien huolehdittavaksi (Gamasutra 2015).

6.2 Pelimoottoreiden lyhyt historia

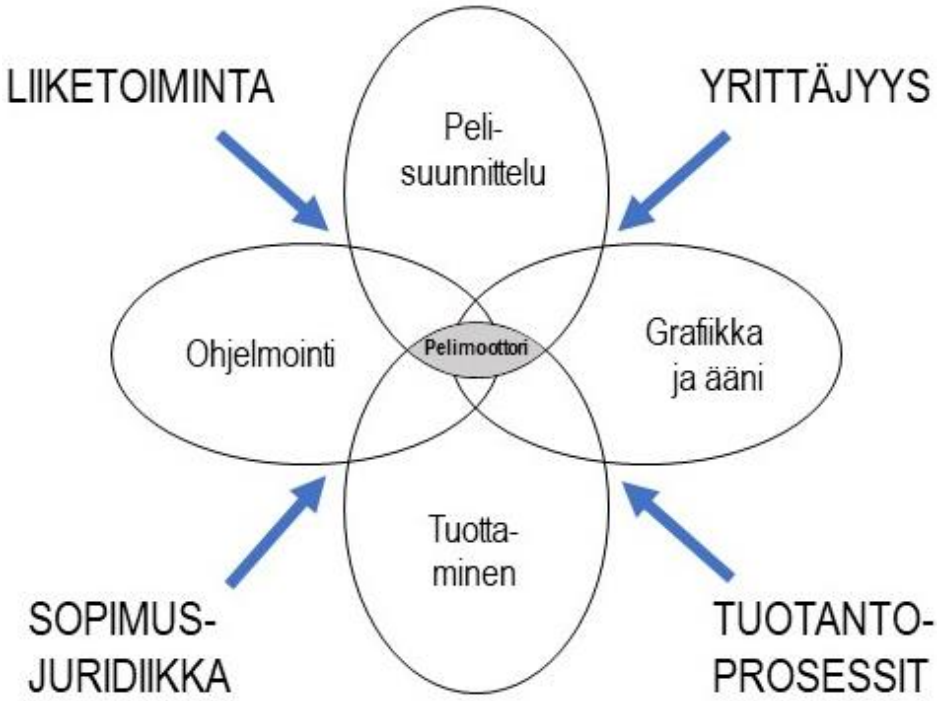
Ensimmäisen uudelleen käytettävän 2D-pelimoottorin kehitti id Software vuonna 1993 peliin nimeltä Doom. Pelimoottorissa käytettiin kaksiulotteisia sprite-grafiikoita, jotka näytettiin 3D-maailman päällä (Tolonen 2012, 2).

Kolmiulotteiset grafiikat mullistivat pelinkehittäjien tavan tehdä pelejä, sillä tämän jälkeen uudelleen käytettävät, sekä avoimen että suljetun lähdekoodin, 2D- sekä 3D-pelimoottorit

alkoivat kehittyä vauhdilla. Avoimen lähdekoodin lisenssillä id Software julkaisi ensimmäisen pelimoottorinsa id Tech 4:n, vuonna 2011.

6.3 Pelimoottorin rooli pelikehityksessä

Kuten kohdassa 4.2 kerrottiin, käsittää peliala suuren kokonaisuuden, jossa on eri osa-alueita. Kaikkien pelikehityksessä tarvittavien osa-alueiden pohjalla on koko toiminnan moottori, nimensä mukaisesti pelimoottori, joka tarjoaa valmiita työkaluja pelin kehittämiseen (Manninen 2007, 11). Pelimoottorin rooli on koko pelikehitysprosessin pohjalla (kuva 3).



Kuva 3 Pelimoottorin asemoituminen koko pelialaan, mukailtu kuva (Manninen 2007, 11)

6.4 Pelimoottorin valinta

Varsinkin aloittelevan pelinkehittäjän kannattaa miettiä jo ennen pelimoottorin valintaa sitä, mihin tarkoitukseen peliä ollaan kehittämässä. Itse pelimoottorin käyttäminen ei välttämättä maksa mitään. Mutta heti, jos pelin aikoo kaupallistaa, tulee vastaan erilaisina prosenttimäärinä laskutettavia kustannuksia, sillä pelimoottorin valmistaja haluaa pelin kaupallisesta toiminnasta itselleen osuuden.

Aloittavan pelikehittäjän pelimoottorin valintaa ohjaavat ensisijaisesti valmiina tarjotut pelimoottorit, jotka nopeuttavat pelin valmistumista (Techbbs 2016). Kun pelikehittäjä lähtee

valitsemaan pelimoottoria, hän törmää ensimmäiseksi pelimoottoreissa käytettäviin ohjelmointikieliin, jotka asettavat omat kriteerinsä valinnalle.

Jos kehittäjä haluaa käyttää vain osaamaansa kieltä, hän joutuu valitsemaan pelimoottoreista, joissa tätä kieltä käytetään. Pelimoottoritarjonta laajenee, mikäli pelikehittäjä osaa tai hänellä on aikaa alkaa opetella uusia ohjelmointikieliä. Mikäli pelikehittäjän ohjelmointikielten osaaminen on suppeaa, hän joutuu valitsemaan mahdollisimman pitkälle valmiiksi räätälöidystä pelimoottoriratkaisusta, joka tyypillisimmin on suljetun lähdekoodin pelimoottori.

Tämän lisäksi tulee ottaa huomioon lukuisat eri ominaisuudet, joita tulevassa pelissä tarvitaan ja valita pelimoottori niiden mukaan. Pelikehittäjän tulee huomioida myös pelimoottorin kustannukset ja minkälaisissa eri tilanteissa niitä saattaa syntyä. Esimerkkinä vaikkapa musiikki tai grafiikat, joiden tuottamisessa niin ikään pelikehittäjän omat taidot ratkaisevat, sillä jos näitä ei pystytä tuottamaan itse, niiden tekemisestä pitää maksaa ulkopuoliselle tuottajalle.

6.5 Avoimen ja suljetun lähdekoodin pelimoottorin ero

Avoimen lähdekoodin pelimoottori eroaa suljetun lähdekoodin pelimoottorista yksinkertaisesti niin, että avoimeen lähdekoodin voi tehdä itse muutoksia. Pelimoottoria voi kustomoida eli siihen voi esimerkiksi lisätä ominaisuuksia, joita siinä ei vielä ole, tai korjata mahdollisia virheitä pelimoottorin koodissa.

Suljetun lähdekoodin pelimoottori otetaan käyttöön sellaisenaan eikä siihen voi tehdä muutoksia. Kaikki muutokset tehdään pelimoottorin valmistajan toimesta. Avoimen lähdekoodin pelimoottorin ominaisuudet eivät ole useinkaan aivan niin kehittyneitä kuin suljetun lähdekoodin pelimoottorin, joten pelikehittäjän tulee osata itse enemmän.

Avoimen lähdekoodin pelimoottorin käyttö ei koskaan maksa, kun taas suljetun lähdekoodin pelimoottorin käytössä tulee varautua lisenssipohjaisiin kustannuksiin, varsinkin pelin kaupallistamisvaiheessa.

7 Pelimoottoreiden vertailu

Tässä luvussa esitellään jokainen vertailtava avoimen sekä suljetun lähdekoodin pelimoottori. Koska opinnäytetyön tarkoitus on tuottaa informaatiopaketti, keskitytään tässä

luvussa tutkimaan kunkin pelimoottorin ominaisuuksia, hyötyjä sekä sen käytölle asetettuja mahdollisia rajoitteita.

7.1 Vertailuun valittujen pelimoottoreiden esittely

Vertailuun valitut pelimoottorit tulivat valituiksi lähtökohtaisesti avoimen ja suljetun lähdekoodin näkökulmista. Vertailun kaksi avoimen lähdekoodin pelimoottoria valittiin siksi, että ne ovat hyvin laajasti käytössä, joten niistä löytyi verrattain hyvin tietoa.

Kolmas avoimen lähdekoodin Godot-pelimoottori taas on tullut tutuksi oman peliprojektin myötä. Se on kahta muuta avoimen lähdekoodin pelimoottoria tuntemattomampi, mutta sen käyttäjäkunta kasvaa koko ajan. Suljetun lähdekoodin pelimoottorit valikoituivat pääosin niiden tunnettuuden vuoksi, jolloin niistä on löydettävissä eniten tietoa vertailuun.

Tähän vertailuun otetut avoimen lähdekoodin eli open source -pelimoottorit ovat Cocos2d, Godot Engine, Torque 3D. Suljetun lähdekoodin eli closed source -pelimoottorit taas ovat CryEngine, Unity sekä Unreal Engine (taulukko 1).

Avoimen lähdekoodin pelimoottorit		
Cocos2d	Godot Engine	Torque 3D
Suljetun lähdekoodin pelimoottorit		
CryEngine	Unity	Unreal Engine

Taulukko 1 Vertailuun valitut pelimoottorit

Cocos2D-x

Cocos2D sai alkunsa vuonna 2008, kun argentiinalaiset Lucio Torre, Daniel Moisset, Rayentray Tappa sekä Ricardo Quesada aloittivat työskentelyn Python-pelimoottorin parissa. Pelimoottorin taustalla oli tavoite tehdä peli PyWeek-tapahtumaan, jossa osallistujat kilpailevat pelin tekemisessä. PyWeek-tapahtumaan peli toteutettiin siten, että siihen käytettiin Python-ohjelmointikieltä, mutta muut avustavat ohjelmointikielet, kuten C++ olivat sallittuja (Pyweek 2010).

Myöhemmin tapahtumaan toteutettu peli ja siihen rakennettu pelimoottori nimettiin Cocos2D:ksi. Monen vaiheen jälkeen Cocos2D-versio 1.0 julkaistiin vuonna 2010. Cocos2D-

x taas on crossplatform-versio, joka tarkoittaa alustariippumatonta pelimoottoria. (Retro.moe 2017.)

Kuva 4 pelistä Badland, joka on Cocos2D-x:llä toteutettu (Badlandgame)

Godot

Godot-pelimoottorin kehityksen aloittivat vuonna 2007 Juan 'reduz' Linietsky ja Ariel 'punto' Manzur. Vuonna 2014 pelimoottorin lähdekoodi julkaistiin GitHubissa täysin avoimena. Pelimoottori julkaistiin MIT-lisenssin alla, joten kaikki pelit, joita kehittäjät sillä tekevät ovat täysin heidän omiaan. (Godot game engine.)

Pelimoottorin lähdekoodin ollessa avoin, voi pelimoottoriin tehdä omia muutoksia. Vuonna 2015 pelimoottori sai paljon lisää ominaisuuksia, esimerkiksi uudelleen kirjoitetut 2D-ominaisuudet sekä monen näytön tuen.

Vuonna 2016 Godot-pelimoottori sai Mozillan avoimen lähdekoodin tuen, 20 000 dollaria, jonka avulla pelimoottoriin on tarkoitus lisätä Websocket, WebAssembly sekä WebGL 2.0 -tuki. (Godot game engine.)

Kuva 5 pelistä Black Velvet Bandit, joka on Godotilla toteutettu (Okamgame)

Torque 3D

Torque3d sai alkunsa vuonna 1998, kun Dynamix kehitti ensimmäisen pelinsä nimeltä Starsiege Tribes. Vuonna 2001 Dynamix julkaisi pelille jatko-osan, Tribes 2:n, jonka julkaisun jälkeen Dynamix sulki ovensa ja osa kehittäjistä perusti uuden yrityksen nimellä GarageGames. Torque 3D lisensoitiin tälle uudelle yritykselle nimellä Torque Game Engine.

Moottoriin rakennettiin uusia kehittyneempiä ominaisuuksia vuonna 2007. Vuonna 2009 tulleen version myötä pelimoottori nimettiin Torque3d-nimiseksi. Avoimeen lähdekoodiin siirryttiin vuonna 2013 ja pelimoottori on nykyään ladattavissa GitHub-versionhallintapalvelussa.

Torque3d on saatavilla Linux- sekä Windows-käyttöjärjestelmille. Pääasiassa Torque3D on tehty Windows-päätelaitteille, mutta pelimoottorin ympärille avoimen lähdekoodin julkaisun myötä kehittynyt yhteisö rakentaa omia versioitaan myös Linux- sekä Mac-käyttöjärjestelmille. Mac-käyttöjärjestelmän versio on toistaiseksi jäänyt niin sanotulle kokeelliselle tasolle eikä ole vielä varteenotettava vaihtoehto.

Kuva 6 pelistä Blockland, joka on Torquella toteutettu (Blockland)

CryEngine

CryEngine sai alkunsa, kun nuori opiskelija nimeltään Cevat Yerli keräsi vuonna 1998 yhteen joukon ihmisiä, joilla oli sama kiinnostuksen kohde kuin hänellä itsellään, pelimoottorit ja pelit. He nimesivät tämän joukon nimellä Crytech. Vuonna 1999 Cevat lähti kahden veljensä kanssa E3-messuille Yhdysvaltoihin esitelläkseen tekemäänsä pelidemoa.

Veljekset saivat pelidemonsa esiteltyä Nvidian edustajalle, ja Nvidia osti kyseisen pelidemon näytönohjainten testausta ja kykyjen esittelyä varten. Crytek yrityksenä oli syntynyt. Pian tämän jälkeen veljekset solmivat sopimuksen Ubisoft-peliyhtiön kanssa. (Polygon 2013.)

Vuonna 2004 Crytek julkaisi pelin nimeltä Far Cry, joka nosti Crytekin lopullisesti merkittäväksi pelikehittäjäksi muiden pelialan yritysten rinnalle ja CryEngine pelimoottorin kaikkien tietoisuuteen. Tällä hetkellä CryEnginessä mennään versiossa viisi. Pelimoottori ei tue ainakaan kirjoitus hetkellä Android- tai iOS-päätelaitteita (Cryengine 2016).

Kuva 7 pelistä Sniper Ghost warrior 3, joka on CryEnginellä toteutettu (Sniperghostwarrior3)

Unity

Unity3D sai alkunsa vuonna 2002, kun tanskalainen ohjelmoija nimeltään Nicholas Francis kysyi apua keskustelufoorumilla asiaan, joka liittyi hänen kehittämäänsä pelimoottoriin. Pian tähän kysymykseen vastasi saksalainen Joachim Ante, joka kehitti myös omaa pelimoottoriaan.

Heidän keskustelunsa johtivat siihen, että he alkoivat kehittää keskenään vain yhtä pelimoottoria. David Helgason kuuli projektista ja halusi mukaan pelimoottorin kehittämiseen, koska oli vakuuttunut, että nämä kaksi kehittäjää olivat keksineet jotain, millä voisi olla tulevaisuutta. (Haas 2002.)

Aluksi tarkoitus oli kehittää pelejä, mutta kehittäjät päättivät pian, että pelimoottorin teknologia olikin se asia, jota he oikeasti halusivat myydä. Tämä tarvitsi tuekseen pelin, jolla päästäisiin esittelemään pelimoottorin kykyä. Tällainen julkaistiinkin vuonna 2005, nimeltään Gooball. Unity-versio 1.0 julkaistiin vuoden 2005 loppupuolella. (Haas 2002.)

Unityllä voidaan kehittää monelle alustalle. Unity tukee 25 eri alustaa ja lista kasvaa koko ajan, kun uusia varteenotettavia alustoja syntyy. Näiden tuettujen alustojen joukossa on mm. Android, iOS, Windows ja Mac OS (Unity).

Kuva 8 pelistä Cities Skylines, joka on Unityllä toteutettu (Paradoxplaza)

Unreal Engine

Unreal Engine tuli julkisuuteen vuonna 1998, kun Epic Games julkaisi Unreal-pelinsä. Pelin sekä pelimoottorin teki poikkeukselliseksi ei pelkästään se, että se oli graafisesti hieno, vaan se mahdollisti myös pelin sisällön muuttamisen helposti muokattavaksi editorin ja UnrealScriptin avulla. Pelaajat ja kehittäjät pääsivät tekemään peliin omia muutoksiaan. (Informit 2009).

Tällä hetkellä Unreal Engine menee versiossa neljä, joka julkaistiin vuonna 2012. Vuonna 2015 se julkaistiin kaikkien saataville täysin ilmaisena. Se sisältää kuitenkin rojaltimaksun, josta kerrotaan enemmän kohdassa 6.7 Ureal Engine f) kustannukset. Pelimoottorin lähdekoodi on saatavilla GitHub-palvelun kautta, mutta muutokset lähdekoodiin täytyy julkaista Epic Gamesin kautta.

Kuva 9 pelistä PlayerUnknowns Battlegrounds, joka on Unreal Enginellä toteutettu (Player Unknowns Battlegrounds)

7.2 Tutkittavat ominaisuudet

Vertailuun valitut ominaisuudet on valittu kategorioista, joita yleensä käytetään, kun halutaan vertailla eri pelimoottoreita. Valittujen ominaisuuksien lisäksi on olemassa runsaasti muitakin ominaisuuksia, mutta tähän työhön valitut ominaisuudet tuovat esiin varsinkin aloittelevalle pelikehittäjälle ne tärkeimmät asiat, joita kehittäjä tulee todennäköisesti ensimmäisessä pelissään tarvitsemaan.

Näiden pohdintojen perusteella tähän opinnäytetyön vertailuun valitut pelimoottorien tutkittavat ominaisuudet ovat:

a) Pelilogiikan ohjelmointikieli
b) Fysiikkamoottori
c) Äänimoottori
d) Verkkopelimahdollisuus
e) 2D- ja 3D-ominaisuudet
f) Kustannukset

7.3 Cocos2d-x

a) Pelilogiikan ohjelmointikieli

Riippuen alustasta, jolle peliä ollaan kehittämässä, ovat kaikki tuetut kielet käytettävissä: C++, Lua ja Javascript. Huom! Windows Phone 8:lle ei ole käytettävissä JavaScript-kieltä.

b) Fysiikkamoottori

Cocos2d-x:n asennuksen mukana tulee valmiina integroitu fysiikkamoottori nimeltään Chipmunk. Vaikka Cocos2d-x sivustolla mainitaan useaankin otteeseen, että pelimoottorin mukana on myös toinen yleisistä fysiikkamoottorivaihtoehdoista, Box2D, ei se pidä täysin paikkaansa, sillä Chipmunk on integroitu pelimoottoriin ikään kuin tiiviimmin kuin Box2d. Kehittäjän on kyllä mahdollista integroida Box2d erikseen pelimoottoriin. (Lavaxp 2014.)

Box2D tukee jatkuvaa törmäyksen tunnistusta, jota Chipmunk taas ei tue. Tästä syntyy ongelma, jota kutsutaan tunnelointiefektiksi. Tämä tarkoittaa, että kovalla vauhdilla liikkuvien objektien törmäys jää fysiikkamoottorilta huomaamatta, jolloin objektit vain kulkevat toistensa läpi. (Chipmunk Game Dynamics 2017).

Fysiikkamoottoreissa on myös käytetty eri ohjelmointikieliä, jolla saattaa olla merkittävä vaikutus kehittäjän fysiikkamoottorin valintaan. Box2d on rakennettu C++-kielellä, joten se saattaa olla helpompi lähestyä kehittäjälle, jolla on olio-ohjelmoinnista kokemusta. Chipmunk on kehitetty C-kielellä joka taas ei ole olio-ohjelmointikieli (Chipmunk Game Dynamics 2013.)

c) Äänimoottori

Ladattavassa paketissa tulee mukana äänimoottori nimeltä SimpleAudioEngine. Tällä pystytään hallitsemaan pelien tarvitsemia ääniä, kuten taustamusiikkia ja eri ääniefektejä. Moottori pitää sisällään myös muutaman ääniefektin, kuten äänen panoroinnin (pan), silmukat (loop), äänenvahvistuksen (gain) sekä äänenkorkeuden (pitch).

Cocos2d-x tukemat ääniformaatit vaihtelevat päätelaitteen tai lähinnä sen käyttöjärjestelmän mukaan. Android-alustalla tuettavat ääniformaatit ovat erittäin kattavat. (Developers.)

Android-alustoilla suositellaan käytettäväksi Ogg Vorbis -formaattia, koska sen vastine MP3 saattaa vaatia lisenssiä pelin pelaajamäärien kasvaessa (GameFromScratch 2015). iOS taas ei tue ollenkaan Ogg Vorbis -formaattia, mutta tukee sen sijaan MP3-formaattia. MP3:a ei tueta, kun peliä kehitetään Windows-alustalle. Windows-alustalle tehtäessä äänet ja musiikit pitää toteuttaa MID- tai WAV-formaateissa. (Cocos2d-x).

d) Verkkopelimahdollisuudet

Cocos2d-x-pelimoottorissa on olemassa verkko-ominaisuuksia, mutta mitä tulee verkkopeliominaisuuksiin, niin parhaaseen lopputulokseen kehittäjien mukaan pääsee käyttämällä ilmaisia C/C++ -kirjastoja, kuten esimerkiksi RakNet-kirjastoa. Tämä vaatii kuitenkin ns. kääreluokan (wrapper) rakentamista (Cocos2d-x Forums 2014).

Eräs verkkopeliin liittyvistä ongelmista on myös palvelinpään ohjelmointitarve. Tähän tarjotaan yhtenä ratkaisuna AppWarp-palvelua, joka on ilmainen tiettyyn rajaan asti. AppWarp toimii vaihtoehtona palvelinpään ohjelmoinnille myös muissakin pelimoottoreissa. (Appwarp 2017).

e) 2D ja 3D ominaisuudet

Vaikka pelimoottorin nimi on hieman harhaanjohtavasti Cocos2D-x, on moottoriin tulossa hyvää vauhtia myös 3D-tuki. Versiosta 3 lähtien on pelimoottoriin sisällytetty erilaisia 3D-ominaisuuksia, ja niitä parannetaan koko ajan (Cocos2d-x).

Kuitenkin verrattuna esimerkiksi mihin tahansa suljetun lähdekoodin pelimoottoriin, kuten vaikkapa Unity 3D:hen, ovat Cocos2d-x:n 3D-ominaisuudet vielä paljon jäljessä. Tällä hetkellä kehittäjät suosittelevatkin esimerkiksi Unreal-pelimoottorin käyttöönottoa 3D-pelin kehittämiseen (Cocos2d-x Forums 2014).

f) Kustannukset

Cocos2d-x on täysin ilmainen eikä sisällä minkäänlaista lisenssimaksua. Pelimoottori ei myöskään vaadi, että peleissä, jotka sillä kehitetään, pitäisi näyttää mitään viitteitä Cocos2d-x-moottorin käytöstä. Cocos2d-x-pelimoottorin kustannukset syntyvät lähinnä pelin levityksestä ja mahdollisista verkkopeliin liittyvistä lisenssi- ja palvelinkustannuksista tai Assettien ostamisesta (Open Source Initiative.)

7.4 Godot Engine

a) Pelilogiikan **ohjelmointikieli**

GDScript on Godotin oma natiivi skriptikieli, joka pohjautuu Python-ohjelmointikieleen. Yhteisö haluaisi tukea mm. JavaScript-kielelle sen suosittuuden takia, mutta Godotin kehittäjät ovat luoneet omasta sekä monen muun mielestä hyvän ohjelmointikielen (Github 2014), joka tarjoaa kaiken mitä pelimoottori tarvitsee.

JavaScript ja Lua eivät esimerkiksi tarjoa ominaisuuksia, jotka ovat tärkeitä pelimoottorille. GDscript on myös suunniteltu yksinkertaiseksi. Tämä avaa mahdollisuuden kielen nopealle oppimiselle ja omaksumiselle, ja nopeuttaa näin ollen pelin tekemistä. (Godotdocs.)

b) Fysiikkamoottori

Godotissa on natiivi fysiikkamoottori, joka on Godotin kehittäjien luoma. Jos kehittäjä kuitenkin haluaa käyttää omaa fysiikkamoottoriaan, kuten esimerkiksi Box2D:ta, hänellä on mahdollisuus muokata itse pelimoottorin toimintaa siten, että saa haluamansa fysiikkamoottorinsa käyttöön (Godot 2016).

c) Äänimoottori

Godotin äänimoottorissa on kaksi erilaista lähestymistapaa äänen toistamiseen. Näistä kumpikin tukee eri tavalla eri ääniformaatteja. Näistä ensimmäisenä voidaan mainita SamplePlayer, joka tukee WAV-muodossa olevia äänitiedostoja eli pakkaamatonta ääntä. Tällä vähennetään päätelaitteelle pakkauksen purkamisen aiheuttamaa prosessorin kuormitusta.

SamplePlayerin tarkoitus on toistaa ääniä, joita tarvitaan usein ja monia samaan aikaan. Tämä tarkoittaa sitä, että nämä äänet on ladattava muistiin, jotta ne ovat nopeasti saatavilla ilman suurta prosessori rasitusta.

Toinen tapa toistaa ääntä, on niin sanottu streaming-malli, jota Godotissa kutsutaan nimellä StreamPlayer. StreamPlayer on tarkoitettu lähinnä musiikin toistamiseen. Tämä tukee pakattuja Opus-, MPC- sekä Ogg Vorbis -ääniformaatteja. Nämä äänet voivat olla pakattuja, koska musiikkia ei tarvitse ladata muistiin, vaan se voidaan toistaa ilman pelissä tapahtuvaa interaktiota. (GameFromScratch 2015.)

d) Verkkopelimahdollisuus

Godotin kolmanteen versioon on tulossa niin sanottu korkean tason rajapinta, jonka tarkoitus on helpottaa verkkopelien toteutusta ilman matalamman tason API-ohjelmointitaitoja. Kirjoitushetkellä Godotin versiossa 2.1 on käytössä matalamman tason ohjelmistorajapinta TCP sekä UDP yhteyksien luomiselle (Godot Engine 2017). Koska ne ovat matalan tason ohjelmistorajapintoja, niiden käyttö vaatii enemmän tuntemusta UDP- ja TCP-tekniikoista. (Godot Engine 2016.)

Jos kehittäjä haluaa helpottaa verkko-ominaisuuksien rakentamista, voi Godotiin ladata GDNet-kirjaston, joka on kääreluokka Enet-kirjastolle. Tämä tarjoaa mahdollisuudet käyttää UDP-protokollaa peleissä. Huomioitavaa on, että Enet ei kirjoitushetkellä tue IPv6-protokollaa. GDnet sekä Enet ovat ilmaisia.

e) 2D- ja 3D-ominaisuudet

Godot on erityisen hyvä 2D-pelien luomiseen, mutta on alusta asti tukenut myös 3D-pelien kehittämistä. Keskustelualueilta löytyy kuitenkin paljon keskustelua siitä, että vaikkakin 3D-moottorin ollessa tehokas, on se edelleen hieman lapsenkengissä verrattuna muihin pelimoottoreihin. 2D-ominaisuudet saavat ylistystä poikkeuksetta lähes jokaisella keskustelualueella. 3D-ominaisuuksista puhutaan kyllä melko positiiviseen sävyyn, mutta samalla mainitaan, että ne eivät ole vielä tarpeeksi kypsiä. (Reddit 2016.)

f) Kustannukset

Kuten Cocos2D-x-moottorin kohdalla, on Godotkin ilmainen ja sen lähdekoodi on avoin. Godotin ytimeen voi kuka tahansa tehdä muutoksia tai lisäyksiä oman pelin niin vaatiessa. Kustannuksia syntyy tässäkin moottorissa lähinnä ulkoisten palveluiden ostotarpeesta, kuten musiikki, grafiikat, mahdolliset palvelimet jne. ellei näitä ole omasta takaa.

7.5 Torque 3D

a) Pelilogiikan ohjelmointikieli

Pelimoottorin ohjelmointikieli on C, mutta pelilogiikan ohjelmointi tapahtuu TorqueScript nimisellä kielellä, ja se muistuttaa hyvin paljon C++ kieltä (Garagegames 2009). TorqueScript on kehitetty täysin Torque 3D-pelikehitystä varten, ja vaikka se

muistuttaakin C-kieltä, se on helpompi omaksua kuin C- tai C++ -kielet. (Github 2016.).

b) Fysiikkamoottori

Windows- sekä Linux-versioissa käytetty fysiikkamoottori on PhysX. PhysX on näytönohjainvalmistaja Nvidian tekemä fysiikkakirjasto.

c) Äänimoottori

Torque 3D-äänimoottori on tehty sillä ajatuksella, että sen perusominaisuudet toimivat kaikilla päätelaitteilla, joihin peli tehdään. Sen parhaimpana etuna on 3D-ääni, jonka avulla äänet voidaan tuottaa pelissä olevan objektin paikan mukaan.

Torque 3D-äänimoottori tukee natiivisti tällä hetkellä WAV- ja OGG/Vorbisääni-formaatteja. Jos äänimoottorin ominaisuuksia haluaa laajentaa, voidaan ottaa käyttöön eri äänimoottoreita, kuten esimerkiksi FMOD, joka avaa käyttöön mm. uusia ääniformaatteja, joita Torque 3D:n oma äänimoottori ei tue. (Garagegames Torque 3-D.)

d) Verkkopelimahdollisuus

Torque 3D on niin sanottu asiakas- (client) ja palvelin- (server) mallinen verkkopelituki. Tällä tarkoitetaan mallia, jossa asiakkaat eli pelaajien päätelaitteet ottavat yhteyden palvelimeen, ja pelin tieto liikkuu keskitetyn palvelimen kautta.

Palvelin voi olla joko täysin pelin palvelemiseen tehty (dedicated server) tai palvelimella voi olla myös pelaaja, joka ottaa yhteyden omalla tietokoneellaan pyörivään palvelimeen (hosted server) ja muut pelaajat ottavat yhteyttä tähän samaan palvelimeen.

Torque 3D omaa hyvin kehittyneen korkeantason verkkopeli-APIn, koska sille alussa suunnitellut sekä sillä tehdyt pelit ovat hyvin voimakkaasti verkkopeliorientoituneita. (Garagegames Torque 3-D.)

e) 2D- ja 3D-ominaisuudet

Torque 3D on nimensä veroisesti suunniteltu 3D-strategiapelien tekemiseen. Torque 3D:stä on olemassa erillinenkin versio nimeltään Torque 2D, joka on tarkoitettu erityisesti 2D-pelien tekemiseen.

Kumpikin, sekä Torque 2D että Torque 3D, on julkaistu MIT-lisenssin alla. Se tarkoittaa, että pelin lähdekoodi on avoimesti saatavilla ja on muokattavissa kenen tahansa toimesta (Garagegames-yhteisö 2011).

f) Kustannukset

Torque 3D on täysin avoimeen lähdekoodiin perustuva, ilmainen pelimootttori. Kustannuksia voi syntyä, jos rakennetaan esimerkiksi verkkopeli, joka tarvitsee erillisen palvelimen. Tällöin palvelin tuo lisäkustannuksia. (Garagegames-yhteisö 2012.)

Peliin on mahdollista myös ostaa valmiiksi tehtyjä osia, ns. assetteja. Näitä ovat esimerkiksi musiikki, grafiikka tai valmiit skriptit, joita voi implementoida omaan peliin. Garage games tarjoaa myös maksettua tukea pelinkehittäjille. Tästä voi siis myös syntyä kustannuksia. (Garagegames-yhteisö 2012.)

7.6 CryEngine

a) Pelilogiikan ohjelmointikieli

Pelimoottorin pääohjelmointikieli on C++. Tällä rakennetaan pelin pääasiallinen pelimekaniikka. Lua-kielellä taas voidaan rakentaa esimerkiksi asioita, joiden pitää aktivoitua tietyillä hetkillä. (Cryengine-yhteisö 2016.)

b) Fysiikkamoottori

CryEnginessä on moottorin kehittäjän oma fysiikkamoottori nimeltään CryPhysics. CryPhysicsin etu on monisäikeisyys eli fysiikkamoottori pystyy hyödyntämään nykyaikaisien prosessorien moniytimisyyden hyödyksi. Myös mobiililaitteissa on nykyään enemmän kuin yksi fyysinen suoritin jolloin myös mobiilipeleissä voitaisiin hyödyntää Cryphysicsin kaltaista fysiikkamoottoria. (Mechwarrior online 2012.)

c) Äänimoottori

CryEngine ei itsessään sisällä äänimoottoria, vaan se on aina sisältänyt ohjelmisto kerroksen joka mahdollistaa kolmannen osapuolen äänimoottorien käyttämisen. Pelimoottorin alkuaikoina tämä tarkoitti, että kehittäjät olivat sidoksissa vain yhteen ääni-

moottoriin, nimeltään Wwise. (Cryengine 2015.) Crytek on kuitenkin toteuttanut tuoreimpaan pelimoottorin versioon uuden audio-kerroksen nimeltään Audio Translation Layer, joka mahdollistaa omavalintaisen äänimoottorin liittämisen CryEngineen (Cryengine 2016).

d) Verkkopelimahdollisuus

CryEnginen verkkopelimodulia kutsutaan nimellä CryNetwork. Se antaa käyttöön hyvin laajat verkkopelin kehityksessä tarvittavat ominaisuudet. Keskustelualueita tutkiessa tulee kuitenkin hyvin nopeasti ilmi, että varsinkin uusimmassa CryEngine-versio 5:ssä verkkopeliominaisuudet ovat erittäin huonosti dokumentoituja. (Cryengine 2016.)

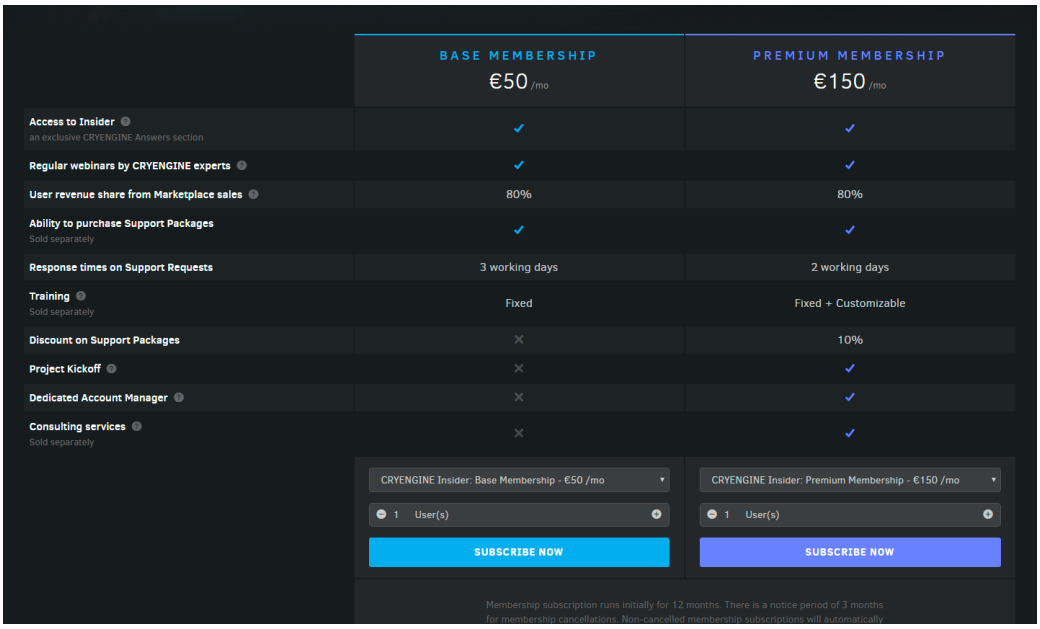
e) 2D- ja 3D-ominaisuudet

Pelimoottori on pääsääntöisesti tarkoitettu 3D-pelien tekemiseen. Pelimoottorin keskustelualueilla on viestejä usealta kehittäjältä, joissa he kaikki toteavat, että 2D-pelin tekeminen näin laadukkaalla pelimoottorilla olisi oikeastaan hyvän pelimoottorin tuhlausta.

f) Kustannukset

CryEngine V-pelimoottori on ladattavissa ilmaiseksi, jolloin ei ole oikeutettu jäsenyyteen. Jäsenyyskäytöstä veloitetaan peruskuukausimaksu, jonka suuruus on 50 euroa kuukaudessa (PC Gamer 2016). Pelinkehittäjällä on lisäksi mahdollisuus ostaa myös itselleen myös premium-tasoinen jäsenyys (kuva 10) hintaan 150 euroa kuukaudessa.

Perusjäsenmaksun lisäksi maksettavalla premium-maksulla kehittäjä tai kehittäjät pääsevät esimerkiksi seuraamaan Crytekin järjestämiä webinaareja tai saavat tukipyyntöhinsä nopeammin vastauksen kuin he, jotka eivät ole premium-hintaa maksaneet. (Crytek 2017.)



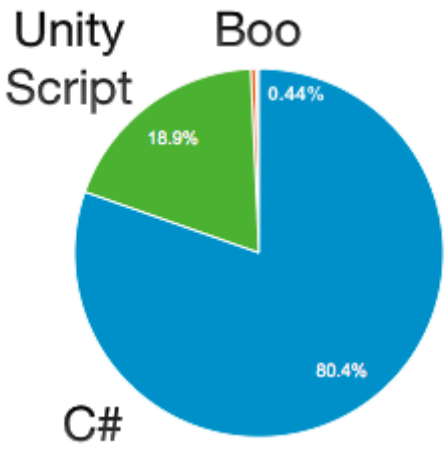
Kuva 10 Kaksitasoisen jäsenyys (Crytek 2017)

7.7 Unity

a) Pelilogiikan ohjelmointikieli

Pääsääntöiset pelilogiikan ohjelmointi- tai skriptaus-kielet Unityssä ovat UnityScript, C# ja Boo. Näistä jälkimmäinen muistuttaa syntaksiltaan hyvin paljon Python-kieltä. UnityScript muistuttaa ja pohjautuukin löyhästi Javascript- tai enemmänkin Ecmascript-kieleen, joten kehittäjät, joilla on kokemusta Javascript-kielestä, omaksuvat tämän helpommin (3D Game Engine Programming 2012).

Kolmesta vaihtoehtoisesta pelilogiikan ohjelmointikielestä (kuva 11) käytetyin vuonna 2014 oli ehdottomasti C#, sillä vain murto-osa peleistä käytti Boo- tai Unityscript-kieltä (Unity Blogs 2014).



Kuva 11 Unity pelilogiikan ohjelmointikielet (Unity Blogs 2014)

Unity ilmoittikin tämän opinnäytetyön kirjoittamisvuonna, että se aikoo poistaa sekä Boo- että Unityscript-kielet tuettujen ohjelmointikielten listalta kokonaan, joka tarkoittaa, että tulevaisuudessa kaikki pelit ja niiden ns. assetit täytyy kirjoittaa C#-kielellä. (Unity Blogs 2017.)

b) Fysiikkamoottori

Unity 3D käyttää fysiikkamoottorinaan PhysX-kirjastoa. Unityn versiossa 5.0, tuli käyttöön PhysX 3, joka paransi fysiikkamoottoria monella osa-alueella ja on täysin uudelleen kirjoitettu versio kahdesta.

c) Äänimoottori

Unity3D sisältää itsessään tehokkaan äänimoottorin, jolla voidaan tuottaa esimerkiksi 3:a ääntä ja erilaisia efektejä, kuten kaiku (echo) tai erilaisia filteröintejä. (Unity).

Tuettuja audio-formaatteja ovat mp3, ogg, wav, aiff, mod, it, s3m sekä xm. Unity 3D:hen voidaan integroida myös omavalintaisia äänimoottoreita, kuten FMod tai Wwise, jotka ovat yleisimpiä käytettyjä äänimoottoreita (Unity).

d) Verkkopelimahdollisuus

Unity sisältää HLAPI- ja LLAPI-komponentit, joiden on tarkoitus kattaa kaikki mitä modernit pelit verkkopelaamiseen vaativat. Kyseistä verkkopelaamisen mahdollisuutta kutsutaan nimellä UNET. (Unity Blogs 2014.)

Unity on jakanut verkkopelien kehittäjät kahteen kategoriaan: kehittäjiin, jotka tekevät monipelejä ja kehittäjiin, jotka tekevät edistyneempiä verkkopelejä tai rakentavat laajan infrastruktuurin pelinsä taustalle.

e) 2D- ja 3D-ominaisuudet

Unity omaa hyvät työkalut sekä 2D- että 3D-pelien tekemiseen. Alussa Unity oli suunnattu vain 3D-pelien tekemiseen, mutta 2D-pelien tekemiseen saatiin enemmän tukea Sprite-editorin sekä muiden 2D-pelien tekemiseen suunnattujen apuvälineiden muodossa. (Haas 2002.)

f) Kustannukset

Jos muut pelimoottorit ovatkin peruskustannuksiltaan ilmaisia, on Unityssä käytössä hieman erikoisempi hinnanmuodostus, jossa portaita on kolme. Ensimmäinen porras on täysin ilmainen. Jos kehitetty peli tuottaa vuotuista liikevaihtoa yli 100 000 dollaria, on Unityn käytöstä luovuttava tai vaihtoehtoisesti kehittäjän tulee suorittaa tilaus, jonka kustannus määräytyy pelin vuosittaisen liikevaihdon mukaan.

Tästä matka jatkuu kalleimmalle asteelle aina liikevaihdon kasvaessa. Liikevaihtoon lasketaan myös mahdollinen joukkorahoitus. Toisella portaalla maksetaan kuukausittainen 35 dollarin maksu, joka sallii vuosittaisen liikevaihdon 200 000 dollaria. Kolmannella portaalla maksetaan kuukausittain 125 dollaria ja vuosittainen liikevaihto saa olla kuinka suuri tahansa. (Game Development 2016.)

Kun kehittäjä on ollut viimeisellä portaalla kaksi vuotta, hän voi halutessaan lopettaa tilauksen ja ilmoittaa tilauksen päättämisestä Unitylle. Tällöin on vielä oikeutettu kolmeen seuraavaan päivitykseen ennen tilauksen päättymistä. (Unity Blogs 2016.)

7.8 Unreal Engine

a) Pelilogiikan ohjelmointikieli

Unreal Engine 4 sisältää tehokkaan visuaalisen ohjelmointitavan, jota pelimoottorissa kutsutaan nimellä Blueprint. Peliin voi myös rakentaa elementtejä, jotka voidaan valjastaa hallittavaksi myös Blueprintin kautta. Tämä vaatii ohjelmoijalta vain ohjelmakoodin rakentamista siten, että halutut asiat näkyvät Blueprints-editorissa. Blueprints- ja

C++-ohjelmoinnin yhdistelmällä on tarkoitus antaa pelisuunnittelijoiden tehdä asioita ilman ohjelmointitaitoa, ja jättää ohjelmointi niille, jotka sitä osaavat. Tämän tarkoitus on nopeuttaa pelin tekemistä. (Reddit 2016.)

c) Fysiikkamoottori

Unreal engine 4 käyttää fysiikkamoottorinaan PhysX-kirjastoa (Unreal Engine). Pelimoottoriin voi liittää oman haluamansa fysiikkamoottorin suoraan lähdekoodin kautta, mutta tämän jälkeen ongelmaksi voi muodostua tulevat pelimoottorin versiot. Jos kehittäjä joutuu muuttamaan pelimoottorin lähdekoodia omalla koodillaan, täytyy tämä versiopäivityksen yhteydessä tehdä aina uudelleen.

Yksi vaihtoehto on tehdä muutokset pelimoottorin lähdekoodiin ja lähettää koodimuutosehdotus GitHubin kautta Ecic Gamesin tarkastettavaksi ja liitettäväksi mahdollisesti suoraan tuleviin versioihin. Tällöin ongelma poistuisi, jolloin se hyödyttäisi myös muita kehittäjiä. (Unreal Eninge Forums 2015.).

b) Äänimoottori

Unreal Engine 4:ssä on sisäänrakennettu äänimoottori. Tällä hetkellä se tukee vain WAV-muotoisia äänitiedostoja, joka on tietynlainen rasite, sillä tiedostomuoto on pakkaamatonta ääntä, jolloin tiedostojen koko kasvaa varsinkin pitkissä musiikkitiedostoissa. Samalla myös itse pelin kokonaiskoko kasvaa. (Unreal Engine.)

Unreal Engine 4:ään voidaan myös integroida FMOD-äänimoottori siten, että se näkyy ja toimii yhtenä osana pelimoottorin graafisessa kehitysympäristössä. Tätä visuaalista skriptauskieltä kutsutaan nimellä Blueprint. (Elsilä 2015, 20.)

c) Verkkopelimahdollisuus

Unreal Engine 4:ää on rakennettu alusta alkaen tukemaan myös verkkopelaamista. Verkko-ominaisuudet on rakennettu niin sanotulla asiakas-/palvelinmallilla. Tällöin yksi peliin osallistuvista tietokoneista tekee päätökset ja kertoo niistä asiakaskoneille, jotka päivittävät tilanteen saatujen tietojen perusteella.

Myös yksinpeli toimii tällä mallilla, mutta tällöin ei ole toista tietokonetta tekemässä päätöksiä, vaan pelaajan oma tietokone on samalla sekä asiakas että palvelin. (Unreal Engine. Networking Overview.)

d) 2D- ja 3D-ominaisuudet

Unreal Engine 4 tukee 2D-pelien toteuttamista. Pelimoottoriin tuotiin 2D-pelikehitykseen järjestelmä, jota kutsutaan nimellä Paper 2D. Tämä mahdollistaa 2D-pelikehityksen, mutta pelimoottorin kaikki ominaisuudet ovat kuitenkin edelleen vahvasti sidottuja 3D-pelien kehittämiseen. Se tekee pelinkehittäjien mielestä Unreal Enginen käytön 2D-pelikehityksessä hieman hankalaksi. (Unreal Engine Forums 2017).

Unreal Enginen vahva visuaalinen skriptikieli, Blueprint, tekee 2D-pelin kehityksestä nopeaa, kunhan vain omaksuu kyseisen työkalun käytön. 2D-ominaisuuksien osalta pelimoottorissa on myös jonkin verran niin kutsuttuja bugeja, joihin kehittäjä tulee matkan varrella varmasti törmäämään. (Unreal Engine Forums 2015. 2D- ja 3D-ominaisuudet).

e) Kustannukset

Unreal Engine -pelimoottorin avulla tehtyjen pelien myynnistä peritään 5 % rojaltimaksu vuosineljännettäin, kun pelin myynti ylittää 3000 dollaria. Esimerkiksi, jos peli myy kolmen ensimmäisen kuukauden aikana 3200 dollarin edestä, täytyy maksaa 5 % rojaltimaksu Epic gamesille 3000 dollarin ylimenevästä osasta, joka on siis 200 dollaria. Muita kustannuksia ei pelimoottorin käytöllä ole eli pelimoottorin käyttö itsessään itsessään ilmaista. (Unreal Engine. Unreal Engine end User License Agreement 2017.)

7.9 Ominaisuustaulukko

Taulukkoon on koottu tutkitun kuuden ominaisuuden tiedot kunkin pelimoottorin kohdalta.

	Cocos2d-x	Godot Engine	Torque	CryEngine V	Unity	Unreal Engine
Pelilogiikan ohjelmointi-kielet	C++ Lua JavaScript	GDScript	TorqueScript	Lua C#	C#	GLSL Cg HLSL UnrealScript
Fysiikka-moottori	Chipmunk (korvattavissa)	Oma (korvattavissa)	Ei mukana, (valinnainen)	CryPhysics	PhysX	PhysX
Äänimoot-tori	Simple Audio Engine (korvattavissa)	Sampleplayer, Streamplayer (korvattavissa)	Oma (korvattavissa)	Ei mukana, (valinnainen)	Oma (korvattavissa)	Oma (korvattavissa)
Verkkopeli-mahdolli-suudet	Websocket (laajennetta-vissa)	TCP/UDP (laajennetta-vissa)	TCP/UDP	UDP	UDP, Websockets	UDP
2D- ja 3D-ominaisuu-det	2D	2D, 3D	3D	3D	2D, 3D	3D
Kustan-nukset	täysin ilmainen	täysin ilmainen	täysin ilmainen	Ilmainen peruskäyttö, lisäpalvelut maksulla	Rajattu versio (ilmainen), täysversio (maksullinen)	Pelimyynnin tuotosta 5 % ro-jaltimaksu

8 Johtopäätökset

Tässä luvussa pohditaan edellisen kohdan vertailun satoa ja tehdään yhteenvetoa sekä johtopäätöksiä kerätyn tiedon pohjalta. Johtopäätöksissä käytetään myös omaa kokemuspohjaa sekä eri keskustelupalstoilta poimittuja muiden pelikehittäjien mielipiteitä.

Opinnäytetyöni lähtöoletuksena oli, että avoimen ja suljetun lähdekoodin pelimoottorit voisi asettaa vastakkain ja vertailla niiden maksuttomuutta ja maksullisuutta. Tutkimuksen alussa syntyi myös olettamus, että maksuton sopisi kustannussyistä paremmin aloittelijalle ja maksullinen taas paremmin ammattilaiselle, koska ammattilaisille kustannukset eivät ole niin suuri kriteeri pelimoottorin valinnassa.

Työn edetessä näkökulma tarkentui. Ei ollutkaan vertailtavissa olevaa näkökulmaa maksuton tai maksullinen eikä näin ollen myöskään niiden sopimista erityisesti aloittelevalle tai ammattikehittäjille. On vain olemassa erilaisia pelimoottoreita, joiden lähdekoodit ovat joko avoimia ja suljettuja, ja joiden pääkustannukset syntyvätkin yleensä vasta pelin kaupallistamisvaiheessa. Kehittäjän ohjelmointitaitojen puutteesta saattaa koitua myös kustannuksia, sillä toimintoja joutuu ostamaan muilta, jos niitä ei osaa tuottaa itse. Nämä kustannukset eivät ole lainkaan riippuvaisia käytettävän pelimoottorin lähdekoodin avoimuudesta tai suljettuudesta.

Pelimoottorien valintaa ohjaa myös sen käyttämä ohjelmointikieli – pelikehittäjän tulee osata tai opetella pelimoottorin käyttämä ohjelmointikieli halutessaan käyttää kyseistä pelimoottoria. Aloittelevan pelinkehittäjän täytyy osata joku pelimoottorin ohjelmointikielistä, että pystyy ylipäätään alkaa kehittää mitään peliä. Kehittyneemmälle, useamman ohjelmointikielen osaajalle, jää valinta eri ohjelmointikieliä käyttävien pelimoottoreiden välillä riippuen myös pelimoottorin tarjoamista muita ominaisuuksista.

Suljetun lähdekoodin pelimoottorit saattaisivatkin yllättäen sopia paremmin aloittelijalle, koska niissä on niin sanotusti valmiimmaksi suunniteltuja kokonaisuuksia, jolloin pelikehittäjän omat koodaustaidot eivät tarvitse olla niin kattavat kuin avoimen lähdekoodin pelimoottoreilla kehitettäessä.

Vertailututkimuksessa paljastui myös alla olevia asioita, joilla kaikilla on omat vaikutuksensa pelimoottorin valintaan. Valintaan liittyvät tekijät ovat tiivistettynä: käytön helppous ja tuki, kaupallisen pelin tuottamisnopeus, laatu ja kustannukset.

Suljetun lähdekoodin pelimoottorin vikoja ei pysty itse korjaamaan, vaan korjauspyynnöt pitää ensin toimittaa pelimoottorin valmistajalle. Valmistaja suorittaa korjauksen omalla aikataulullaan riippumatta kehittäjän aikataulusta tai kiireestä. Kehittäjä joutuu joko odottamaan pelimoottorin valmistajan tekemiä korjauksia tai kiertämään mahdolliset pelimoottorin ongelmat, joka saattaa aiheuttaa paljon ylimääräistä työtä. Avoimen lähdekoodin pelimoottorissa vastaavat ongelmat saatetaan korjata nopeasti yhteisön toimesta ja saada uusi, korjattu versio nopeasti kaikkien käyttöön.

Suljetun lähdekoodin laatu on usein parempaa kuin avoimen lähdekoodin. Tämä näkyy varsikin visuaalisuudessa eli 3D-renderöinnissä. Koska maailma muuttuu, ovat suljetunkin lähdekoodin pelimoottorit avanneet omaa lähdekoodiansa julkisesti näkyville ja lähdekoodiin saa jopa tehdä muutoksia, mutta muutettua lähdekoodia ei saa levittää.

Jos kehittäjä korjaa jonkin virheen pelimoottorissa, hänen kannattaa ilmoittaa se valmistajalle. Jos muutoksia ei ilmoita, niin tulevat versiopäivitykset saattavat tuoda vanhat ongelmat takaisin ja kehittäjän tekemät omat korjaukset ja muutokset häviävät, koska niitä ei ole tehty uuteen versioon, joka korvaa edelliset versiot. Osasyynä lähdekoodin avaamiselle on varmasti ohjelmoijien pelikehityksen helpottaminen. Lähdekoodin avulla on helpompi löytää ja poistaa omat virheet.

Koska suljetun lähdekoodin pelimoottoreita kehitetään suurten yritysten toimesta, ovat ne useimmiten paremmin dokumentoituja ja omaavat hyvät kauppapaikat eri asseteille. Assetit ovat siis eräänlaisia myytäviä palasia liittyen esimerkiksi grafiikkaan tai musiikkiin. Ne voivat olla myös ohjelmistokirjastoja, joista voidaan ostaa omaan käyttöön palanen vaikkapa oman pelin tekoälyn hallintaan. Avoimen lähdekoodin pelimoottoreilla ei ole vastaavaa kauppapaikkaa.

Varsinaisesti avoin tai suljettu lähdekoodi eivät eroa toisistaan aloittelevan pelinkehittäjän näkökulmasta. Jos aloitteleva kehittäjä on valmis maksamaan jäsenmaksuja, joita suljetun lähdekoodin pelimoottoreissa usein on, hän saa tuekseen kanavan, jota kautta voi kysyä apua.

Avoimen lähdekoodin pelimoottoreiden käyttö on varteen otettava vaihtoehto varsinkin siitä vaiheessa, kun pelinkehittäjän koodaustaidot kehittyvät. Suosituimmilla avoimen lähdekoodin pelimoottoreilla on paljon kehittäjiä, joten ongelmat ja virheet tulevat usein paljon nopeammin löydetyiksi ja korjatuiksi kuin suljetun lähdekoodin, jonka takana on joustamattomammat eikä niin ketterät organisaatiot, joiden päivityksillä on omat syklinsä.

Vastausta tutkimuskysymykseen kumpaa pelimoottorityyppiä pelikehityksessä kannattaa käyttää ei suoraan voida antaa. Tutkimus antaa viitteitä siitä, että aloittelevan kehittäjän kannattaisi käyttää suljetun lähdekoodin pelimoottoria, mikäli omat ohjelmointitaidot eivät riitä lähdekoodin kehittämiseen. Ei ole järkevää käyttää avoimen lähdekoodin pelimoottoria, jos ei pysty käyttämään sen tuomia etuja tai tuottamaan yhtä laadukasta lopputulosta kuin suljetun lähdekoodin pelimoottorilla. Kirjoitushetkellä avoimen lähdekoodin pelimoottoreilla ei päästä laadullisesti samaan lopputulokseen kuin suljetun lähdekoodin pelimoottoreilla.

Tutkimuksen perusteella voisi antaa jonkinlaisen suosituksen, että aloittelijan olisi kätevämpää käyttää suljetun lähdekoodin pelimoottoria halutessaan kehittää peliä helpommin ja saada sen nopeammin markkinoille. Jos ja kun kehittäjän omat ohjelmointitaidot kehittyvät ja suljetun lähdekoodin pelimoottorin kaupallistamisvaiheessa syntyneet kustannukset kasvavat liian suuriksi, kannattaa miettiä siirtymistä avoimen lähdekoodin pelimoottorin käyttäjäksi, mikäli se tarjoaa omien vaatimusten mukaiset asiat.

Mikäli suljetun lähdekoodin pelimoottorilla päästään haluttuun lopputulokseen eikä sen kehittämisen tietynlainen hitaus häiritse eivätkä myöskään kaupallistamisen kustannukset ole esteenä, niin ei ole olemassa erityistä syytä lähteä käyttämään avoimen lähdekoodin pelimoottoreita siltikään, vaikka ohjelmointitaidotkin olisivatkin kehittyneet. Kaikki riippuu siitä, mitä pelimoottorilta ja peliltä halutaan. Avoimen lähdekoodin pelimoottoreiden laatu kuitenkin kehittyy huimaa vauhtia, ja laajan kehittämisjoukon vuoksi ne saattanevat saavuutta lähitulevaisuudessa suljetun lähdekoodin pelimoottoreiden laatutason, jolloin nämäkin suositukset tietenkin vanhenevat.

Opinnäytetyön prosessi opetti etsimään relevantteja lähteitä. Lähdemateriaalia löytyi todella runsaasti ja lähteistä uusimmat löytyvät paremmin verkosta kuin painetusta kirjallisuudesta, johtuen peliteknologian nopeasta kehityksestä. Suurin haaste oli löytää tiedollisesti oikeassa olevia lähteitä, koska monesti eri lähteistä löytyy kirjoituksia samasta aiheesta, mutta täysin eri faktoilla.

Tutkimustyö opetti myös sen, että itse tiedon keräämisessä on todella paljon työtä, ja että se on itse asiassa tutkimuksen suurin osuus. Kun lähdetietoja oli kattava määrä, oli itse opinnäytetyön kirjoittaminen enemmänkin tiedon muotoilua ja opinnäytetyön rakenteen miettimistä, joka vei kyllä kohtuullisesti aikaa sekin. Ajankäyttö työssäkäyvänä perheellisenä oli toisinaan myös haasteellista. Mutta seuraavan opinnäytetyön tekemisessä olisin taas viisaampi.

Lähteet

3D Game Engine Programming 2012. Luettavissa: https://www.3dgep.com/scripting-unity-3-5/ Luettu: 12.10.2017.

Appwarp 2017. Luettavissa: http://appwarp.shephertz.com/pricing/ Luettu: 8.10.2017.

Badlandgame. Luettavissa: http://badlandgame.com/badland-game Luettu: 8.11.2017

Blockland. Luettavissa: http://blockland.us/ Luettu: 8.11.2017.

Chipmunk Game Dynamics 2013. Why Chipmunk? Luettavissa: https://chipmunk-physics.net/aboutChipmunk.php Luettu: 8.11.2017.

Chipmunk Game Dynamics 2017. Continuous Collision Detection. Luettavissa: https://chipmunk-physics.net/forum/viewtopic.php?f=1&t=7543 Luettu: 8.11.2017.

Coco Cocos2d-x. Graphics. Luettavissa: http://www.cocos2d-x.org/wiki/3D_Graphics Luettu: 8.10.2017.

Cocos2d-x 2014. Forums. 2D- ja 3D-ominaisuudet. Luettavissa: http://discuss.cocos2d-x.org/t/is-cocos2d-x-still-relevant-in-the-age-of-unity3d-and-unreal-engine/34211/5 Luettu: 3.10.2017.

Cocos2d-x Forums 2014. Multiplayer. Luettavisssa: http://discuss.cocos2d-x.org/t/i-need-creating-multiplayer-game/16747/10 Luettu: 3.10.2017.

Cocos2d-x. ProgrammingGuide. Luettavissa: http://www.cocos2d-x.org/wiki/Audio Luettu: 3.10.2017.

Codeandweb 2017. Luettavissa: https://www.codeandweb.com/physicseditor/tutorials/creating-physics-shapes-for-cocos2d-x Luettu: 6.9.2017.

Cryengine-yhteisö 2016. Luettavissa: https://www.cryengine.com/community_archive/viewtopic.php?f=355&t=134429 Luettu: 3.10.2017.

Cryengine 2014. When to use what language Lua/ C++? Luettavissa: https://www.cryengine.com/community_archive/viewtopic.php?f=314&t=120075 Luettu: 3.10.2017.

Cryengine 2015. So what is the truth about the sound engine licensing? Luettavissa: https://www.cryengine.com/community_archive/viewtopic.php?f=355&t=132824 Luettu: 5.10.2017.

Cryengine 2016. Engine Modules. Luettavissa: http://docs.cryengine.com/display/CEPROG/CryNetwork Luettu: 3.10.2017.

Cryengine 2016. What is the Audio Translation Layer (ATL)? Luettavissa: http://docs.cryengine.com/pages/viewpage.action?pageId=23308234 Luettu: 5.10.2017.

Crytek 2017. Jäsenyys. Luettavissa: https://www.cryengine.com/get-cryengine/memberships/ Luettu: 14.10.2017.

Developers. Luettavissa: https://developer.android.com/guide/topics/media/media-formats.html Luettu: 8.10.2017.

Enger, M. 2013. Game Engines: How do they work? Luettavissa: https://www.giantbomb.com/profile/michaelenger/blog/game-engines-how-do-they-work/101529/ Luettu: 8.9.2017.

Elsilä, J. 2015. Äänimoottorin käyttö osana äänisuunnittelua pelinkehitysympäristössä. Opinnäytetyö. Turun ammattikorkeakoulu. Luettavissa: https://www.theseus.fi/bitstream/handle/10024/96913/Elsila_Jussi.pdf?sequence=1 Luettu: 29.10.2017.

Frosenbyte 2015. Luettavissa: http://www.frozenbyte.com/wordpress2013/wp-content/uploads/2015/05/MP1.jpg Luettu: 8.11.2017.

Gamasutra 2015. Luettavissa: https://www.gamasutra.com/blogs/EdEarl/20150406/240469/Hidden_Costs_of_Scripting_Game_Behaviour.php Luettu: 3.10.2017.

Game Development 2016. Luettavissa: https://gamedev.stackexchange.com/questions/107778/what-happens-if-i-make-more

than-100k-with-the-free-unity-license Luettu: 16.10.2017.

GameFromScratch 2015. Cocos2d-x. Luettavissa: http://www.gamefromscratch.com/post/2015/11/23/Cocos2D-X-Tutorial-Series-Playing-Sound-Effects-and-Music.aspx Luettu: 20.9.2017.

GameFromScratch 2015. Godot. Luettavissa: http://www.gamefromscratch.com/post/2015/02/06/Godot-Engine-Tutorial-Part-4-Playing-Sound-FX-and-Music.aspx Luettu: 20.9.2017.

GameObjects 2017. Unty3D. Luettavissa: http://docs.unity3d.com/Manual/GameObjects.html Luettu: 7.8.2017.

Garagegames 2009. Luettavissa: https://www.garagegames.com/community/forums/viewthread/83679 Luettu: 15.9.2017.

Garagegames Torque 3-D. Luettavissa: http://docs.garagegames.com/torque-3d/official/content/documentation/Engine/Audio/SFXOverview.html Luettu: 15.9.2017.

Garagegames 2011. Community. Luettavissa: https://www.garagegames.com/community/forums/viewthread/124756 Luettu: 15.9.2017.

Garagegames 2012. Community. Luettavissa: https://www.garagegames.com/community/forums/viewthread/132215 Luettu: 28.9.2017.

Github 2014. Luettavissa: https://github.com/godotengine/godot/issues/937 Luettu: 9.10.2017.

Github 2016. Luettavissa: https://github.com/GarageGames/Torque2D/wiki/TorqueScript-Overview Luettu: 15.9.2017.

Godot 2016. Luettavissa: https://godotengine.org/qa/9157/box2d-velocity-acceleration-would-calculated-automatically Luettu: 8.10.2017.

Godot Engine 2016. Luettavissa: https://godotengine.org/article/godots-new-high-level-networking-preview Luettu: 21.11.2017.

Godot Engine 2017. Luettavissa: https://godotengine.org/qa/17549/multiplayer-in-2-1-3 Luettu: 21.10.2017.

Godotdocs. Scrpiting. Luettavissa: http://docs.godotengine.org/en/stable/learning/step_by_step/scripting.html Luettu: 9.10.2017.

Godot game engine. Luettavissa: https://ipfs.io/ipfs/QmXoypizjW3WknFiJnKLwHCnL72vedxjQkDDP1mXWo6uco/wiki/Godot_(game_engine).html Luettu: 9.10.2017.

Grönfors, M. 1985. Kvalitatiiviset kenttätyömenetelmät. 2. painos. Wsoy. Helsinki.

Haas, J. 2002. A History of the Unity Game Engine. Luettavissa: https://web.wpi.edu/Pubs/E-project/Available/E-project-030614-143124/unrestricted/Haas_IQP_Final.pdf Luettu: 11.10.2017.

Hiltunen, KooPee, Latva, S. & Kaleva, J-P. 2016. The Game Industry of Finland. Report 2016. Luettavissa: http://www.neogames.fi/wp-content/uploads/2017/04/Finnish-Game-Industry-Report-2016_web_070529.pdf Luettu: 8.10.2016.

Informit 2009. Luettavissa: http://www.informit.com/articles/article.aspx?p=1377834 Luettu: 8.10.2016.

Kamk 2017. Vertaileva tutkimus. Luettavissa: http://www.kamk.fi/opari/Opinnaytetyopakki/Teoreettinen-materiaali/Tukimateriaali/Tutkimustyypit/Vertaileva Luettu: 18.8.2017.

Lavaxp 2014. Integrating Box2d with Cocos2d-x. Luettavissa: http://www.lavaxp.net/integrating-box2d-with-cocos2d-x/ Luettu: 8.11.2017.

Manninen, T. 2007. Pelisuunnittelun käsikirja. Ideasta eteenpäin. 1. painos. Kustannus Oy Rajalla. Oulu.

Mechwarrior online 2012. Luettavissa: https://mwomercs.com/forums/topic/3991-does-the-cryengine-3-support-physx/ Luettu: 18.9.2017.

Neogames 2017. Hub of the Finnish Game Industry. Luettavissa: http://www.neogames.fi/ Luettu: 8.10.2017.

Okamgame. Luettavissa: https://okamgames.com/ Luettu: 8.11.2017.

Open Source Initiative. MIT lisenssi. Luettavissa: https://plus.google.com/+opensourceini tiative Luettu: 6.9.2017.

Paradoxplaza. Luettavissa: http://www.paradoxplaza.com/cities-skylines/CSCS00GSK-MASTER.html Luettu: 8.11.2017.

PC Gamer 2016. Luettavissa: http://www.pcgamer.com/cryengine-v-releases-today-on-a-pay-what-you-want-basis/ Luettu: 15.10.2017.

Player Unknowns Battlegrounds. Luettavissa: https://playbattlegrounds.com/main.pu Luettu: 8.11.2017.

Polygon 2013. The Story Of Crytek: From X-Isle Through Redemption. Luettavissa: https://www.polygon.com/features/2013/7/11/4503782/crytek-x-isle-redemption Luettu: 12.11.2018.

Pyweek. 2010. PyWeek Game Progamming Challenge Rules. Luettavissa: https://pyweek.org/s/rules/ Luettu: 7.10.2017.

Reddit 2016. Godot. Luettavissa: https://www.reddit.com/r/godot/comments/4iwsey/why_arent_you_making_a_3d_game_with_godot/ Luettu: 10.10.2017.

Reddit 2016.Unrealengine. Luettavissa: https://www.reddit.com/r/unrealengine/comments/4gebhz/people_who_use_a_mix_of_c_and_blueprint_when_do/ Luettu: 10.10.2017.

Retro.moe 2017. The history of Cocos2d in a glimpse. Luettavissa: https://retro.moe/2017/04/16/cocos2d-in-a-glimpse/ Luettu: 7.10.2017.

Sniper Ghost Warrior 3. Luettavissa: http://sniperghostwarrior3.com/ Luettu: 8.11.2017.

Techbbs 2016. Keskustelufoorumi pelimoottoreista. Luettavissa:

https://bbs.io-tech.fi/threads/pelimoottoreista-keskustelua-unreal-engine-unity-source-jne.645/ Luettu: 15.10.2017.

Tekes 2013. Peliteollisuus – kehityspolku. Luettavissa: https://www.tekes.fi/globalassets/julkaisut/peliteollisuus_kehityspolku.pdf Luettu: 6.9.2017.

Tolonen, J. 2012. Pelimoottorit ennen, nyt ja huomenna. Opinnäytetyö. Centria ammattikorkeakoulu. Luettavissa: https://www.theseus.fi/bitstream/handle/10024/50817/loppu-tyo.pdf?sequence=1 Luettu 15.10.2017.

Uiah 2007. Vertailu. Luettavissa: http://www2.uiah.fi/projects/metodi/072.htm Luettu: 18.8.2017.

Unity. Audio Effects. Luettavissa: https://docs.unity3d.com/Manual/class-AudioEffectMixer.html Luettu: 12.10.2017.

Unity. Audio Files. Luettavissa: https://docs.unity3d.com/Manual/AudioFiles.html Luettu: 12.10.2017.

Unity. Build once, deploy anywhere. Luettavissa: https://unity3d.com/unity/features/multiplatform Luettu: 11.10.2017.

Unity Blogs 2014. Documentation, Unity scripting languages and you. Luettavissa: https://blogs.unity3d.com/2014/09/03/documentation-unity-scripting-languages-and-you/ Luettu: 12.10.2017.

Unity Blogs 2014. All about the Unity networking transport layer. Luettavissa: https://blogs.unity3d.com/2014/06/11/all-about-the-unity-networking-transport-layer/ Luettu: 16.10.2017.

Unity Blogs 2016. Evolution of our products and pricing. Luettavissa: https://blogs.unity3d.com/2016/06/16/evolution-of-our-products-and-pricing/ Luettu: 16.10.2017.

Unity Blogs 2017. UnityScript's long ride off into the sunset. Luettavissa: https://blogs.unity3d.com/2017/08/11/unityscripts-long-ride-off-into-the-sunset/ Luettu: 12.10.2017.

Unity. Documentation 2017. Creating and Using Scripts. Luettavissa: http://docs.unity3d.com/Manual/CreatingAndUsingScripts.html Luettu: 6.9.2017.

Unreal Engine. Audio System Overview. https://docs.unrealengine.com/latest/INT/Engine/Audio/Overview/ Luettu: 11.10.2017.

Unreal Engine Forums 2015. Physics Eninge. Luettavissa: https://forums.unrealengine.com/development-discussion/c-gameplay-programming/28256-how-much-work-is-it-to-replace-the-physics-engine Luettu: 18.10.2017.

Unreal Engine Forums 2015. 2D- ja 3D-ominaisuudet. Luettavissa: https://forums.unrealengine.com/community/general-discussion/37008-how-powerful-is-ue4-s-2d-game-egine Luettu: 5.11.2017.

Unreal Engine Forums 2017. Luettavissa: https://forums.unrealengine.com/community/general-discussion/105396-why-do-people-say-do-not-create-2d-games-in-ue4 Luettu: 5.11.2017.

Unreal Engine. Networking Overview. Luettavissa: https://docs.unrealengine.com/latest/INT/Gameplay/Networking/Overview/ Luettu: 5.11.2017.

Unreal Engine. Physics Simulation. Luettavissa: https://docs.unrealengine.com/latest/INT/Engine/Physics/ Luettu: 18.10.2017.

Unreal Engine. Unreal Engine End User License Agreement 2017. Luettavissa: https://www.unrealengine.com/en-US/eula Luettu: 5.11.2017.

Ward, J. 2008. What is a Game Engine? Luettavissa: https://www.gamecareerguide.com/features/529/what_is_a_game_.php Luettu: 8.9.2017.

Watkins, A. 2011. Creating Games with Unity and Maya. Elsevier Inc. Burlington.