KARELIA-AMMATTIKORKEAKOULU
Tietojenkäsittelyn koulutusohjelma

Janne Mononen

VIIDEN 3D-PELIMOOTTORIN VERTAILU UUDEN KEHITTÄJÄN NÄKÖKULMASTA

Opinnäytetyö
Marraskuu 2016

Karelia AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Marraskuu 2016
Tietojenkäsittelyn koulutusohjelma

Tikkarinne 9
80220 JOENSUU
013 260 600

Tekijä(t)
Janne Mononen

Nimeke
Viiden 3D-pelimoottorin vertailu uuden kehittäjän näkökulmasta

Toimeksiantaja
Joensuu Games

Tiivistelmä

Tässä opinnäytetyössä vertaillaan viittä helposti saatavilla olevaa 3D-pelimoottoria uuden tai kokemattoman kehittäjän näkökulmasta. Opinnäytetyön tavoitteena on vertailla pelimoottoreiden ominaisuuksia ja selventää lukijalle minkälaiselle kehittäjälle tai kehittäjäryhmälle opinnäytetyöhön valitut pelimoottorit sopivat. Opinnäytetyö on tutkimuksellinen.

Opinnäytetyöhön valittuja pelimoottoreita vertailtiin sellaisten ominaisuuksien osalta, jotka ovat tärkeitä kaikissa 3D-peliprojekteissa. Ominaisuuksia vertailtiin niiden monipuolisuuden, helppokäyttöisyyden ja dokumentaation kattavuuden osalta. Opinnäytetyössä luotiin myös katsaus pelimoottoreiden käyttökustannuksiin ja rojaltimaksuihin.

Opinnäytetyön tuloksena arvioidaan minkälaiselle kehittäjälle tai kehittäjäryhmälle vertailun pelimoottorit soveltuvat. Opinnäytetyön lopputuloksena kehittyi myös paljon vertailutietoa ja huomioita pelimoottoreiden ominaisuuksista. Opinnäytetyön tuloksien on tarkoitus helpottaa uuden kehittäjän pelimoottorivalintaa.

Kieli	Sivuja 95
Suomi	Liitteet

Asiasanat

pelimoottori, vertailu, Unity, Unreal Engine, Cryengine, Autodesk, Amazon, Stingray, Lumberyard

Karelia AMMATTIKORKEAKOULU

THESIS
November 2016
Degree Programme In Business Information Technology

Tikkarinne 9
80220 JOENSUU
013 260 600

Author(s)
Janne Mononen

Title
Comparison of Five 3D Game Engines from New Developers Point of View

Commissioned by
Joensuu Games

Abstract

In this thesis five easy to acquire 3D game engines are compared from the view of a new or unexperienced developer. The aim of the thesis is to compare the features of the chosen game engines and to give recommendations of good use cases of the engines. This thesis is investigational so there is no game or application developed during the making of this thesis.

The selected game engines were compared by the features that are important in the development of all 3D game projects. The features were compared by their versatility, ease of use and coverage of documentation. This thesis also examines the use expenses and royalty amounts of the chosen game engines.

As the result of this thesis each of the chosen game engines is evaluated on how it applies to different sort of developers or development teams. Also as a result a lot of comparison information and observations between the features of chosen game engines were developed. The result of this thesis is intended to make it easier for a new developer to choose which game engine to use in development.

Language	Pages 95
Finnish	Appendices

Keywords

game engine, comparison, Unity, Unreal Engine, Cryengine, Autodesk, Amazon, Stingray, Lumberyard

Sisältö

Käsitteet

.NET-kehitysympäristö — on Microsoftin kehittämä monia ohjelmointikieliä tukeva ohjelmointikirjasto ohjelmistojen kehitykseen (Microsoft 2016a).

Co-op — (engl. Cooperation) tarkoittaa pelitapaa, jossa vähintään kaksi pelaajaa työskentelee pelissä yhdessä tavoitteen saavuttamiseksi (Tom's Hardware 2003).

Collider — on peliobjektissa oleva fyysinen muoto, jota käytetään fysiikkalaskujen laskemiseen.

DCC — (engl. Digital Content Creation) tarkoittaa digitaalista sisällän tuottamista.

DLL — (engl. Dynamic Link Library) on dynaaminen linkkikirjasto, joka sisältää ohjelmakoodia ja tietoa, jota voidaan käyttää useassa ohjelmassa joihin linkkikirjasto on sisällytetty (Microsoft Support 2016).

Entiteetti — (engl. Entity) on itsenäinen pelimaailman osa, jolla on jotain toiminnallisuutta.

Epäsuora valaistus — on valoa, joka ei tule pinnalle suoraan valonlähteestä, vaan se on ensin kimmonnut jostain toisesta pinnasta.

FBX — on tiedostomuoto, jota käytetään useissa pelimoottoreissa peliobjektien tietojen tallentamiseen (Autodesk 2016a).

Mono-kehitysympäristö — on avoin kehitysympäristö, joka toimii pohjana käyttää .NET-arkkitehtuuria (Mono Project 2016).

Node — on visuaalisessa ohjelmoinnissa ja materiaalien muokkauksessa käytettävä tietorakenne joita voi yhdistää toisiinsa.

NPC — (engl. Non-player-character) on pelihahmo, joka on tekoälyn ohjaama.

Partikkelijärjestelmä — on järjestelmä, jolla voidaan simuloida epämääräistä ja satunnaista tapahtumaa kuten räjähdystä tai lumisadetta.

Pbr — tarkoittaa physically based rendering, eli fysiikkaan pohjautuvaa renderöintiä. Tällä tavalla kutsuttu ja ohjelmoitu shader pyrkii mallintamaan valon käyttäytymistä luonnollisesti ja fysiikan lakeja noudattaen. (Russell 2015.)

Rasterointi — on prosessi jossa vektori- tai 3D-grafiikasta luodaan 2D-rasterikuva (Wikipedia 2016a).

Renderöinti	(engl. Rendering) on prosessi jossa mallista luodaan kuva ohjelmallisesti (Wikipedia 2016b).
Skriptaus	(engl. Scripting) on tulkittavan ohjelmointikielen kirjoittamista. Tulkittava ohjelmointikoodi ei tarvitse erillistä kääntämistä.
Shader	on tietokoneen näytönohjaimella ajettava tietokoneohjelma, joka tekee toimintoja pikselille tai vertexille (Wikipedia 2016c).
UMG	(engl. Unreal Motion Graphics) on työkalu jota käytetään Unreal Enginessä työskennellessä käyttöliittymägrafiikoiden kanssa (Unreal Engine Documentation 2016f).
Verteksi	on 3D-mallissa avaruudellinen kohta jossa yhtyvät vähintään kolme mallin reunaa (Slick 2016).
Vokseli	on yksikkö, joka kuvastaa pistettä kolmiulotteisessa ympäristössä (WhatIs 2007).

1 Johdanto

Pelikehityksen aloittaminen ei ole koskaan ollut niin helppoa kuin nykyään. Kehittäjille on nykyisin tarjolla useita loistavia pelimoottoreita ja monet niistä ovat jopa ilmaisia käyttää tai niiden käytön aloittaminen on ilmaista. Uudelle kehittäjälle voi kuitenkin tulla myös valinnanvaikeus pelimoottoria valittaessa, johon tämä opinnäytetyö pyrkii tuomaan helpotusta. Tässä opinnäytetyössä vertaillaan viittä 3D-pelimoottoria niiden käyttöehtojen, kustannusten, ominaisuuksien, helppokäyttöisyyden ja monipuolisuuden osalta. Vertailun tavoitteena on selvittää kunkin vertailtavan pelimoottorin pelikehityksen kannalta tärkeimpien ominaisuuksien hyvät ja huonot puolet. Vertailun tavoitteena on myös luoda ehdotus siitä, että kenelle tai minkälaiselle projektiryhmälle kukin vertailtavista pelimoottoreista soveltuu. Vertailtavat moottorit ovat Unity Technologiesin kehittämä Unity 3D, Epic Gamesin kehittämä Unreal Engine 4, Crytekin kehittämä CryEngine V, Amazonin kehittämä Lumberyard sekä Autodeskin kehittämä Stingray. Nämä pelimoottorit valikoituivat vertailukohteiksi niiden suosion ja saatavuuden takia. Kaikki pelimoottorit ovat helposti saatavissa ja kaikkia moottoreita voi vähintään kokeilla ilman maksua. Unity, Unreal Engine ja CryEngine ovat pelimoottoreina vanhempia kuin Lumberyard ja Stingray. Lumberyard ja Stingray valittiin tähän opinnäytetyöhön myös tästä syystä. On mielenkiintoista nähdä mitä ominaisuuksia uudet pelimoottorit tarjoavat vanhempien pelimoottoreiden rinnalla. Kaikki vertailuun valikoituneet pelimoottorit ovat 3D-pelimoottoreita. Osa valituista pelimoottoreista tukee myös 2D-kehitystä, mutta 2D-ominaisuudet on tästä opinnäytetyöstä rajattu pois vertailun helpottamiseksi ja aiheen rajaamiseksi. 2D-pelimoottoreita on vertaillut blogikirjoituksen muodossa esimerkiksi Jamie Flanagan (2014).

Yleiskatsausta moottoriin luotaessa käydään läpi moottorin käyttöehdoista sitä, sisältääkö pelimoottorien käyttöehdot moottorin käyttöä rajoittavia tekijöitä tai sisältääkö käyttöehdot jotain muuta toimintaa huomattavasti rajoittavaa. Yleiskatsauksessa käydään läpi myös moottorin suorat kustannukset ja rojaltimaksujen

suuruus. Samalla luodaan katsaus muutamaan moottorilla julkaistuun peliin. Kolmansien osapuolien sovelluksien tarpeesta koituvia kustannuksia ei tässä opinnäytetyössä lasketa tarkkaan, mutta ylimääräiset kustannukset huomioidaan kuitenkin moottoreita vertailtaessa.

Pelimoottoreiden ominaisuuksia vertaillaan järjestyksessä, jonka voisi kuvitella sopivan pelinkehitysprosessiin kokonaisuutena. Vertailtaviksi ominaisuuksiksi on myös valittu ainoastaan sellaisia ominaisuuksia, joita todennäköisesti käytetään 3D-peliä kehitettäessä. Vertailun ulkopuolelle on tietoisesti jätetty pois ominaisuuksia, joiden käyttökohteet ovat rajallisia tai ne palvelevat vain isoa projektiryhmää. Ensimmäisenä vertaillaan moottorien käyttöliittymien ulkoasua sekä niiden muokattavuutta. Käyttöliittymien vertailusta siirrytään vertailemaan sitä, kuinka sisällön tuonti tapahtuu moottoriin staattisten ja animoitujen objektien muodossa. Objektien tuonnin jälkeen vertaillaan, kuinka tuotuihin objekteihin saadaan luotua materiaalit moottorin sisällä ja kuinka monipuolisesti tai millä tavalla näitä materiaaleja voi muokata. Seuraavana vertailtavana asiana on käyttöliittymän ja valikoiden teko. Vertailua tehdään moottorin sisältämien työkalujen pohjalta sekä myös kolmansien osapuolten sovellusten ja teknologioiden osalta. Käyttöliittymän luonnin jälkeen opinnäytetyössä siirrytään vertailemaan pelilogiikan luontia. Pelilogiikan sisällyttämisen osalta moottoreita vertaillaan ohjelmointikielen osalta tai visuaalisen ohjelmoinnin osalta. Seuraavaksi arvioidaan moottorien valaistuksen ja jälkikäsittelyefektien monipuolisuutta ja helppoutta. Valaistuksen ja efektien jälkeen arvioidaan vertailtavien pelimoottoreiden profiloinnin ja kehittäjänäkymien monipuolisuutta ja käytön helppoutta. Viimeisenä asiana ennen loppupohdintaa käydään läpi alustat joille kehitetyn pelin voi pelimoottorilla kääntää ja selvitetään, kuinka tämä käännös tapahtuu. Vertailu aloitetaan aina käymällä ensimmäisenä läpi Unityn ominaisuudet. Unityn ominaisuuksien läpikäynnin jälkeen muista moottoreista vertaillaan ominaisuudet vähintään samalla kattavuudella, mutta myös moottoreille yksilölliset ominaisuudet huomioidaan.

2 Mikä on pelimoottori ja mitä se tarjoaa käyttäjälle

Pelimoottoreiden ominaisuuksia vertailtaessa ja vertailua lukiessa on hyvä olla käsitys siitä, mikä pelimoottori on, mistä se koostuu ja mitä hyötyjä se tarjoaa käyttäjälle. Lyhyesti ilmaistuna pelimoottori on kehitysympäristö, joka sisältää työkaluja sekä lähdekoodin, joiden avulla peli saadaan kehitettyä nopeasti (Kalderon 2011).

Michael Enger (2013) jakaa blogikirjoituksessaan pelimoottorin pääkomponentit seitsemään eri osaan: käyttäjän syötteiden hallinta, grafiikoiden tuotto, äänet, verkkotoiminnallisuudet, fysiikat, graafinen käyttöliittymän ja skriptaus. Tätä listausta täydentää Eyali Kalderonin (2011) tekemä blogikirjoitus, josta listaan voidaan vielä lisätä tekoäly, lokitietojen kirjaus ja resurssien hallinta. Kuten tästä listauksesta voidaan huomata, on pelimoottori monen jo yksistään opinnäytetyöksi riittävän kokonaisuuden summa. Seuraavaksi käydään läpi jokainen näistä edellä mainituista komponenteista ja tutustutaan hieman niiden toimintaan.

2.1 Käyttäjän syötteiden hallinta

Jokainen peli vaatii käyttäjän syötteiden käsittelyn interaktiivisuuden luomiseksi. Ilman interaktiivisuutta peli olisi oikeastaan pikemminkin elokuva. Perinteisiin ohjainlaitteisiin kuuluvat esimerkiksi hiiri ja näppäimistö sekä käsissä pideltävä ohjain, joka voi sisältää nappeja, liipaisimia ja ohjainsauvoja. Näissä perinteisissä ohjaimissa on kahdenlaisia käyttäjän syötteitä. Nappia voidaan painaa, nostaa ylös tai pitää pohjassa ja pelimoottori voidaan asettaa kuuntelemaan näitä napin toimintoja. Pelaajan voi esimerkiksi laittaa liikkumaan eteenpäin W-näppäimen painalluksella ja liikkumisen voi lopettaa, kun W-näppäin nostetaan ylös. Ohjainsauvoilla, liipaisimilla ja hiirellä on liikkumisrata ja tätä liikkumisrataa pelimoottoreissa voidaan kuunnella portaattomasti. Portaaton kuuntelu mahdollistaa esimerkiksi pelaajahahmon liikuttamisen hitaalla nopeudella, kun ohjainsauvaa on työnnetty eteenpäin vain hieman. Perinteisellä napilla tällainen ei onnistu.

Perinteisten ohjaustapojen rinnalle on viime vuosikymmenen aikana noussut myös vaihtoehtoisia ohjaustapoja. Vuonna 2006 julkaistu Nintendo Wii toi yli 100 miljoonan konsolin myynnillään liikeohjauksen massoille (Nintendo 2016). Liikeohjauksen lisäksi iso osa pelaamisesta tapahtuu nykyään älypuhelimella tai tabletilla, joissa pääasiallisena ohjaustapana on kosketusnäytön käyttäminen. Liikeohjaimien kategoriaan sisältyy myös Microsoftin Kinect, joka pystyy kameroillaan seuraamaan yhtä aikaa kuutta eri henkilöä ja 25 niveltä henkilöä kohden (Microsoft 2016b).

2.2 Grafiikan tuottaminen

Pelimoottorin tulee kyetä tuottamaan ja piirtämään grafiikkaa sekä pelaajalle että myös kehittäjälle pelin kehitysvaiheessa. Kehitysvaiheessa kehittäjä näkee pelimoottorin näköportista kasatun pelimaailman ja voi myös muokata pelimaailmaa kyseisen portin kautta. Valmiissa pelissä pelaaja näkee pelimaailman renderöitynä omaan sovellusikkunaan. Kolmiulotteinen pelimaailma muutetaan 2D-kuvaksi renderöijällä, joka tietynlaisen grafiikkaprosessin kautta muuttaa pelimaailmassa olevat mallit, tekstuurit ja valot käyttäjän näkemäksi kuvaksi. Sujuvan kokemuksen takia tavoitteena on aina piirtää uusi kuva vähintään 30 kertaa sekunnissa, mikä jättää yhdelle kuvalle aikaa näkyä 33 millisekuntia. Koska kyseessä on näin lyhyt aika, on renderöijän kyettävä muodostamaan käyttäjälle näkyvä kuva erittäin tehokkaasti. Tästä syystä yleisesti käytettävä renderöintimetodi on rasterointi. Rasteroinnissa maailmasta luodaan maailmaa katsovalle kameralle yksinkertainen 2D-kuva. Värit ja valaistus kuvaan saadaan näytönohjaimella toimivista ohjelmista, jotka laskevat maailmassa oleviin malleihin värit ja varjostukset maailmassa olevien valojen, tekstuurien ja pinnanmuotojen mukaisesti. Näitä näytönohjaimella toimivia ohjelmia kutsutaan shadereiksi. Shadereita voi ohjelmoida käytettäväksi moneen eri käyttötarkoitukseen. Ohjelmoija voi esimerkiksi tehdä oman shaderin sarjakuvamaisen lopputuloksen mahdollistamiseksi tai shaderin, joka pyrkii piirtämään mallin mahdollisimman realistisena (Wikipedia 2016c).

Eri pelimoottoreiden renderöijät eroavat ainakin siinä, ovatko ne esi- vai jälkirenderöijiä (Owens 2013), onko valaistus etukäteen laskettua vai reaaliaikaista ja

käyttääkö renderöijä pbr-shadereita. Viime vuosien aikana kaikkiin isoihin pelimoottoreihin on tullut pbr-renderöinti-ominaisuus. Kun pelaajat vertailevat eri pelimoottoreita on usein puheenaiheena grafiikan laatu. Grafiikan laatuun vaikuttaa moni asia, mutta pelimoottorin renderöijä ja shaderit antavat peleille pelimoottorille tyypillisen ulkoasun ja jotkut pelaajat voivat tunnistaa pelissä käytetyn moottorin pelkästään grafiikan perusteella (GameDev 2015).

2.3 Äänet

Pelien äänimaailma on monille pelaajille immersion kannalta välttämätön. Äänimaailma on ollut mukana peleissä jo 1970-luvun arcade-peleistä lähtien. Kuten muutkin pelien osa-alueet, on myös ääniteknologia kehittynyt huimasti pelaamisen alkuajoista tähän päivään. Nykyään pelimoottoreissa on yleensä oma äänimoottori, jossa tapahtuu äänen prosessointi ja toisto. Hyvänä esimerkkinä äänimoottorista mainittakoon Audiokineticin Wwise, jota käytetään äänimoottorina muun muassa Unityssä, Unrealissa ja CryEnginessä (Audiokinetic 2016).

Alkuaikojen monoäänipiippauksista on siirrytty monipuolisiin stereo- ja tilaääni-järjestelmiin, joilla äänimaailma saadaan tehtyä rikkaaksi ja ympäristön mukaiseksi. Nykyisissä peleissä 3D-äänet ovat tärkeässä osassa myös pelimekaniikkojen kannalta. Esimerkiksi kilpailullisissa peleissä voi olla tärkeää kuulla mistä päin vihollinen on tulossa tai vaihtoehtoisesti pelaaja voi kulkea hiljaisempaa vauhtia, jotta vihollinen ei kuule pelaajan askeleita.

Kehitysvaiheessa äänien asettelun ja toistamisen pelimaailmassa tulisi olla yksinkertaista ja sellaista se yleensä onkin. Pelimaailmassa on yksinkertaisimmillaan kuuntelija, joka on yleensä kiinni pelikuvan näyttävässä kamerassa sekä äänilähteitä joista lähtevät äänet kuuntelija ottaa kiinni ja toistaa pelaajalle. Jos käytössä on 3D-äänet, niin äänen voimakkuus ja muut arvot riippuvat siitä, kuinka kaukana lähde on kuuntelijasta ja missä suunnassa lähde on kuuntelijaan nähden. Myös ympäristö vaikuttaa yleensä siihen, miltä ääni kuulostaa.

2.4 Verkkotoiminnallisuudet

Pelimoottoreiden tarjoamat verkkotoiminnallisuudet vaihtelevat paljon. Nykyisin pelimoottoreissa on yleensä työkalut moninpelin luontiin. Moninpelin tekemisen työkaluihin kuuluu verkkoisännän hallinta, peliin liittyvien pelaajien hallinta, aulan hallinta ja pelaajien tietojen synkronointi pelaajien välillä.

Pelimoottorit pyrkivät kuitenkin erottumaan tarjoamalla yksilöllisimpiä palveluita kehittäjän käyttöön. Esimerkiksi Unity tarjoaa kehittäjälle helpon tavan integroida mainokset kehitettyyn peliin (Geig 2015). Amazonin Lumberyard puolestaan tarjoaa kehittäjälle mahdollisuuden integroida suositun striimauspalvelu Twitchin toiminnallisuuksia suoraan kehitettävään peliin. Lumberyardiin voi integroida myös laajasti skaalautuvat moninpeliserverit pelaajamäärien mukaan vaihtuvalla hinnoittelulla (Barr 2016).

2.5 Fysiikat

Fysiikkamoottori on lyhyesti sanottuna tietokoneohjelma, jolla voidaan simuloida jotain fyysistä järjestelmää. Fyysisiä järjestelmiä voivat olla esimerkiksi kiinteiden kappaleiden simulointi, pehmeät ainekset ja nestedynamiikat. Pelimoottoreissa käytettävät fysiikkamoottorit eivät tuota läheskään täydellistä fysiikan mallinnusta, vaan moottoreissa käytetään hieman yksinkertaistettuja laskutoimituksia ja pienempää tarkkuutta (Wikipedia 2016d.)

Pelimoottoreihin soveltuvia fysiikkamoottoreita on monia. Yleisiä ovat ainakin Havok ja Nvidian PhysX. Yleensä pelimoottoreissa käytetään yksinkertaistettua mallia fysiikanmallinnuksen laskemiseen. Käytännössä tämä tarkoittaa sitä, että esimerkiksi ihmishahmomaisella pelaajalla on erillinen yksinkertaisempi muoto fysiikoiden mallintamiseen. Tämä muoto voi olla esimerkiksi laatikko tai sylinteri. Mitä monimutkaisempi tämä fysiikoita mallintava muoto on, sitä hitaampaa on laskea fyysisiä laskuja. Pelimoottoreissa tämän fysiikoissa käytettävän mallin luonti tapahtuu hieman eri tavoilla. Malli voidaan joissakin pelimoottoreissa generoida itse pelimoottorin sisällä ja generointiin voidaan käyttää erilaisia algoritmeja, jotka tuottavat tarkkuudeltaan erilaisen mallin. Toisissa pelimoottoreissa

fyysinen malli pitää taas luoda ulkopuolisessa sovelluksessa ja tuoda pelimoottoriin näkyvän mallin mukana.

2.6 Graafinen käyttöliittymä

Peleissä tietoa pelaajalle välitetään graafisen käyttöliittymän kautta. Graafinen käyttöliittymä voi olla avaruudellisesti niin sanotusti näyttötilassa, jolloin käyttöliittymä piirretään kaiken muun pelikuvan päälle 2D:nä, renderöitynä pelimaailmaan 3D-elementtinä tai suoraan osana pelimaailmaa.

Kuvaan 1 on koostettu Marcus Andrewsin (2010) blogikirjoituksesta sekä Erik Fagerholtin ja Magnus Lorentzonin (2009, 48–51) opinnäytetyöstä kaavio, joka esittää pelin käyttöliittymän osat. Ei-diegeettinen näyttötapa on perinteisin, siinä käyttöliittymä piirretään suoraan ruudulle ja on irrallinen pelimaailmasta. Avaruudellisesti esitetty käyttöliittymä voi ilmentyä esimerkiksi pelaajan nimitekstinä, joka on koko ajan hahmon päällä ja seuraa hahmoa minne hän sitten meneekin. Meta-näyttötapa tarkoittaa käyttöliittymän osaa, joka tapahtuu pelimaailmassa, mutta jota ei näytetä pelaajalle avaruudellisesti. Yleisin käyttökohde meta-näyttötavalle on ruudulla näkyvät tehosteet, kun pelaajalle tapahtuu jotain pelimaailmassa. Esimerkiksi pelaajan vahingoittuessa ruudulla voi näkyä veriroiskeita. Diegeettinen näyttötapa tarkoittaa sitä, että käyttöliittymä on sulautettu suoraan pelimaailmaan. Käytännössä tämä voi tarkoittaa sitä, että pelaajalla ei ole muuta tapaa tietää kuinka monta panosta lippaassa on jäljellä, kuin katsoa läpinäkyvän lippaan sivusta panosten määrä. Kuvassa 2 vasemmassa alareunassa näkyvät käyttöliittymän elementit eivät ole esitetty avaruudellisesti 3D-tilassa eivätkä ne ole olemassa fiktionaalisessa pelimaailmassa, ne ovat siis esitetty ei-diegeettisen esitystavan mukaisesti. Kuvassa näkyvän aseen ammusmäärä on nähtävissä myös aseen fyysisessä tähtäimessä. Tähtäin on olemassa pelimaailmassa ja se myös näytetään avaruudellisesti 3D-tilassa, joten se on diegeettinen esitystapa. Kuvan reunoilla näkyvä vaalea monikulmiokuvio kuvastaa pelaajan loukkaantumista ja se on esitetty meta-näyttötapana.

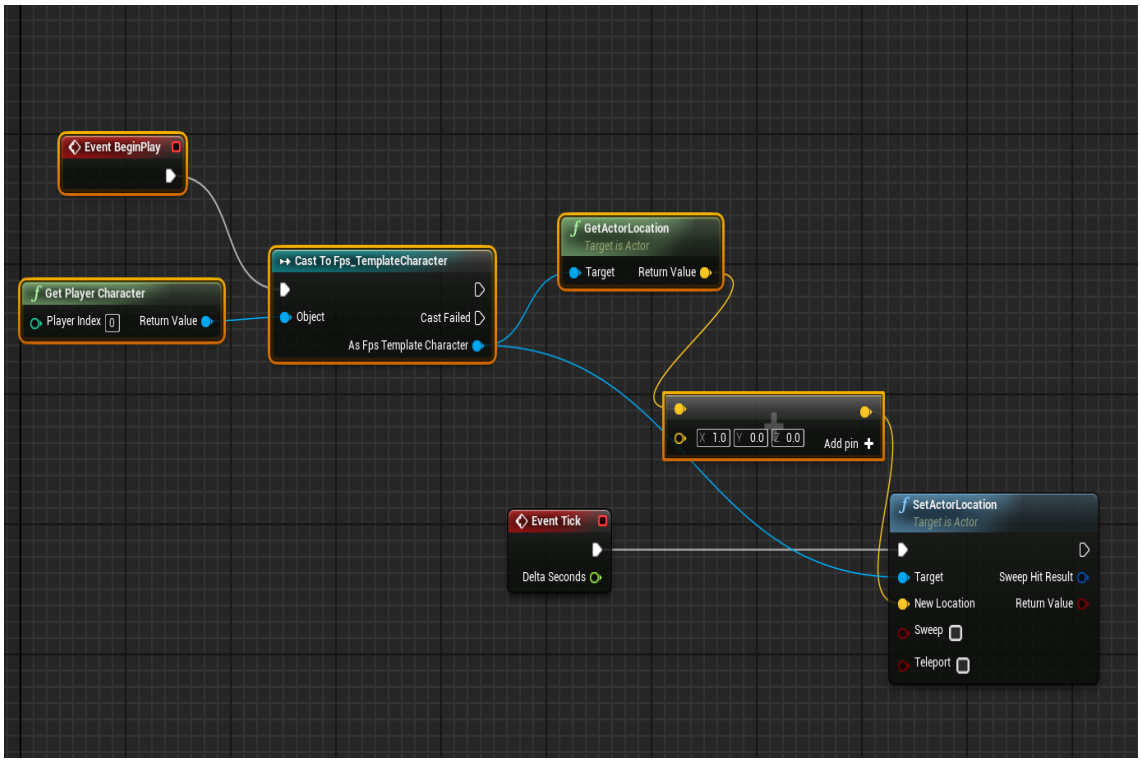
Kuva 1. Käyttöliittymän luokittelu tyylin mukaan (Andrews 2010; Fagerholt & Lorentzon 2009, 48–51).

Kuva 2. Erilaisia käyttöliittymän esitystapoja pelikuvassa

Käyttöliittymän toteutukseen eri pelimoottorit tarjoavat vaihtoehtoisia ratkaisuja. Joissakin pelimoottoreissa käyttöliittymä voidaan toteuttaa moottorin omilla työkaluilla. Yleensä tällaisten työkalujen käyttö on suoraviivaista ja kehittäjän kannalta helppoa. Osa moottoreista turvautuu käyttöliittymän toteutuksessa kokonaan ulkopuolisten järjestelmien integrointiin. Ulkoisia käyttöliittymän luontityökaluja on paljon, mutta yksi ehdottomasti suosituimmista on Autodeskin Scaleform (Chapple 2014). Vaikka pelimoottori sisältäisikin oman ratkaisunsa käyttöliittymän luontiin, voi moottoriin integroida myös ulkopuolisen käyttöliittymien luontityökalun. Ulkoinen työkalu valitaan usein sen monipuolisuuden ja suorituskyvyn vuoksi.

2.7 Skriptaus

Pelimoottoreissa logiikan ohjelmointi voidaan jakaa karkeasti kahteen eri kategoriaan: osassa pelimoottoreita käytetään skriptauskieltä ja osassa ohjelmointikieltä. Skriptaus ja ohjelmointi eroavat siinä, että skriptauskieli on tulkittavaa kieltä ja se ei tarvitse erillistä kääntämistä, ohjelmointikieli taas tulee kääntää aina koodin kirjoittamisen jälkeen tietokoneen ymmärtämään muotoon (Crowder 2013). Perinteisten kirjoitettavien kielien rinnalle useat pelimoottorit tarjoavat myös vaihtoehtoisen visuaalisen tavan sisällyttää pelilogiikka peliin. Tätä tapaa kutsutaan visuaaliseksi ohjelmoinniksi. Kuvasta 3 voidaan nähdä, kuinka Unreal Enginen Blueprint-ohjelmoinnissa pelaajan paikkaa siirretään x-akselilla eteenpäin jokaisella ruudunpäivityksellä.



Kuva 3. Pelaajan liikuttaminen Unreal Enginessä visuaalisella ohjelmoinnilla.

Käännettävät ohjelmointikielet ovat suoritusnopeudeltaan hieman nopeampia kuin tulkattavat skriptauskielet (Crowder 2013). Skriptauskielet ovat taas puolestaan nopeampia kirjoittaa, sillä niillä työskennellessä ei tarvitse odottaa ohjelmakoodin kääntymistä, vaan muutoksia pääsee testaamaan heti koodin kirjoittamisen jälkeen. Visuaalisia ohjelmointitapoja käyttäessä isojen kokonaisuuksien hahmottaminen ja hallinta voi käydä hankalaksi. Visuaalinen ohjelmointi on myös hitaampaa kuin käännettävä ohjelmointikoodi.

2.8 Tekoäly

Tekoälyä käytetään peleissä tuomaan pelimaailma eloon ja tuottamaan reaktioita pelaajan tekoihin (Graft 2015). Tekoälyn toteutus ei välttämättä vaadi erillistä työkalua, vaan sen voi toteuttaa myös puhtaasti loogisella ohjelmoinnilla. Useat pelimoottorit kuitenkin sisältävät valmiin pohjan tai työkalun tekoälyn nopeammalle toteutukselle tai vaihtoehtoisesti jonkin ulkopuolisen ladattavan työkalun. Esimerkiksi Unreal Engine sisältää tapahtumapainotteisen työkalun käytöspuiden toteuttamiseen (Unreal Engine 2016). Unity puolestaan ei sisällä valmista työkalua käytöspuun luontiin, mutta Unityn kauppa-alueelta voi ladata esimerkiksi ilmaisen

RAIN AI for Unity -lisäosan, jolla käytöspuun luonti onnistuu (Unity Asset Store 2015).

Tekoälyyn liittyy myös tekoälyn pelimaailman objekteille aiheuttama tarve liikkua ja liikkeen toteutus. Jos npc-hahmo näkee pelaajan ja haluaa päästä pelaajan luokse, tulee sillä olla jokin tieto siitä, mitä reittiä sen kannattaa kulkea. Älykästä liikkumista varten pelimoottoreissa käytetään navigaatioverkkoa ja reitinhakualgoritmeja. Reitinhakualgoritmeja on useita, mutta yksi suosituimmista pelimoottorikäytössä on A*-algoritmi, joka on muunnos Djikstran algoritmista (Wikipedia 2016e). Reitinhakualgoritmeja voi myös implementoida projekteihin valmiiden algoritmien pohjalta (Coke And Code 2016).

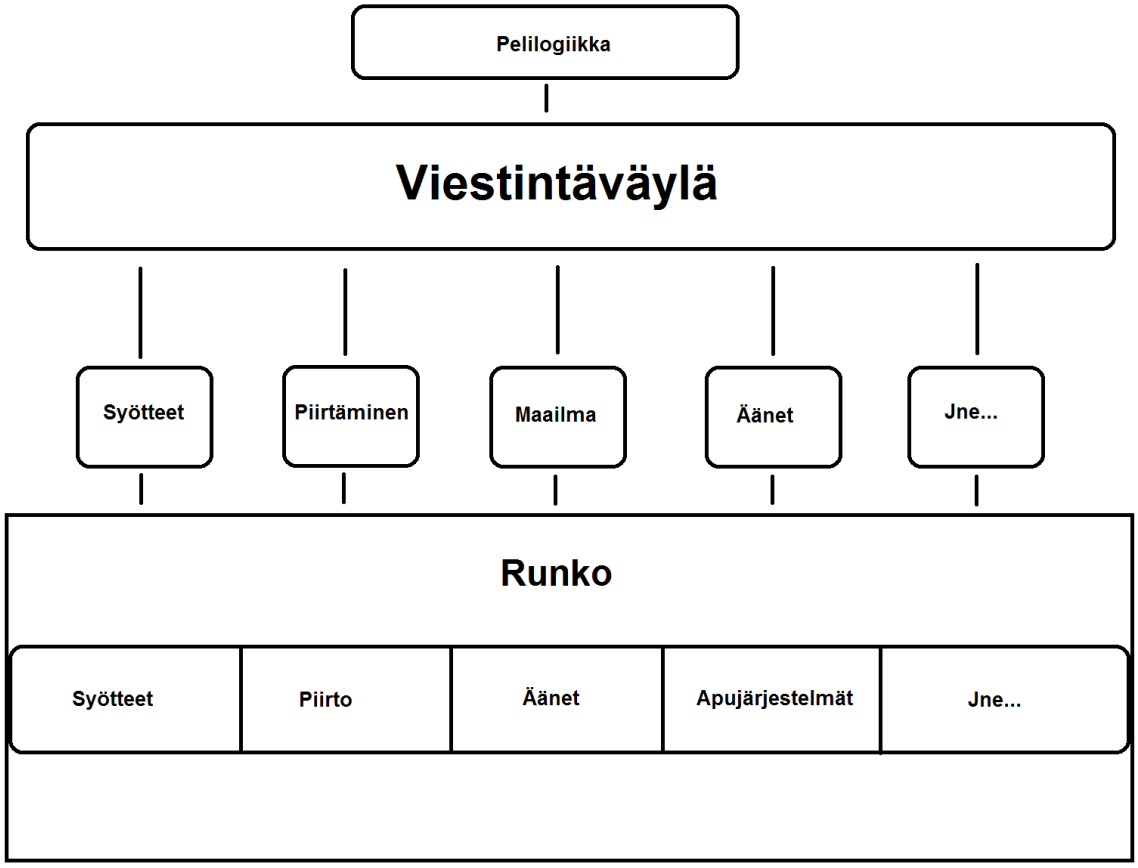
2.9 Lokitietojen kirjaus

Lokitietojen kirjaus ei näy pelaajalle suoraan valmiissa tuotteessa, mutta se on kehittäjälle erityisen tärkeä työkalu virheiden ja ongelmien etsimisessä. Lokijärjestelmän taustalla pelimoottoreissa on yleensä viestintäjärjestelmä, jonka kautta pelimoottorin komponentit viestivät toisilleen (Kissner 2015). Kehittäjä voi käyttää lokitietojen kirjausta esimerkiksi tarkistaakseen jonkin muuttujan arvon tai arvon muuttumisen suorituksen aikana. Pelimoottorit sisältävät myös paljon sisäänrakennettua tietojen välittämistä kehittäjälle lokitietojen kautta. Kehittäjälle voi moottorin luomista lokitiedostoista ilmetä jokin ongelmakohta, jota kehittäjä ei muuten huomaisi.

2.10 Resurssien hallinta

Pelisisällön hallinta tehokkaasti pelisession aikana on erityisen tärkeää. Pelit sisältävät paljon resursseja, mutta ne eivät välttämättä käytä niitä kaikkia yhtä aikaa. Tästä syystä pelimoottorin resurssien hallitsijan tulisikin osata tehokkaasti ladata resursseja oikeassa muodossa ja oikeaan paikkaan sekä myös purkaa nämä ladatut resurssit pois käytöstä. (Luukkonen 2015, 8.)

Pelimoottoreiden resurssienhallintajärjestelmä on siis vastuussa kaikista pelin sisällä olevasta sisällöstä. Myös muistinhallinta liittyy resurssienhallintaan. Hyvät muistin käyttötavat ja hyvä muistinhallinta ovat avainasemassa kun tavoitellaan nopeasti toimivaa peliä (Kissner 2015). Kuva 4 on Michael Kissnerin samaisesta blogista koostettu (2015), ja siitä voidaan nähdä, kuinka viestintäväylä yhdistyy pelimoottorin kaikkiin järjestelmiin. Resurssienhallinta on pelimoottoreissa pääosin sisäänrakennettua, mutta myös kehittäjän toimilla voi olla vaikutusta resurssien optimaaliseen käyttöön ja esimerkiksi muistivuotojen estämiseen.



Kuva 4. Esimerkki viestintäväylän sijoittumisesta pelimoottorin järjestelmiin nähden (mukaillen Kissner 2015).

3 Aikaisemmat tutkimukset aiheesta

Tutkimuksia joissa vertaillaan kaikkia juuri tähän opinnäytetyöhön valittuja moottoreita ei kirjoitushetkellä löydy. Vertailuja joistakin opinnäytetyössä käsitellyistä moottoreista löytyy blogikirjoituksista tai keskustelufoorumeilta. Internetin videopalveluista löytyvät vertailut keskittyvät pääosin vertailemaan moottoreilla jo tehtyjen pelien graafisia ominaisuuksia. Kattavien vertailujen puute ei ole yllättävää, sillä kaksi tässä opinnäytetyössä vertailtavista pelimoottoreista ovat erittäin uusia.

Lähimmäksi tässä opinnäytetyössä luotavaa vertailua pääsee blogikirjoitus, jonka on kirjoittanut Mark Masters (2014). Masters vertailee blogissaan Unityä, Source 2:a, Unreal Engineä ja CryEngineä. Mastersin mielestä Unity on hyvä valinta mobiilipelien tekemiseen, kun taas Unreal Enginen ja CryEnginen graafiset mahdollisuudet ovat paljon paremmat. Täytyy kuitenkin pitää mielessä, että tämäkin vertailu on jo kaksi vuotta vanha ja moottoreista on tullut uusia versioita sekä muut ominaisuudet ovat päivittyneet kustannuksia myöten.

4 Yleiskatsaus vertailtaviin moottoreihin

Tässä luvussa käydään lyhyesti läpi kunkin vertailtavan pelimoottorin historia ja tehdään katsaus muutamaan pelimoottorilla tehtyyn peliin, mikäli moottorilla on julkaistu pelejä. Samalla käydään myös läpi, onko pelimoottorin käytölle asetettu rajoitteita käyttöehdoissa ja onko moottorin käyttämisessä kustannuksia.

4.1 Unity

Unity on Unity Technologiesin kehittämä useaa eri julkaisualustaa tukeva pelimoottori. Ensimmäinen versio moottorista julistettiin vuonna 2005 Applen käyttöjärjestelmälle. Sittemmin Unity on päivittynyt tukemaan useampia laitealustoja ja

siitä on julkaistu viisi suurempaa versiota. (Wikipedia 2016f.) Tässä opinnäytetyössä käsitellään Unityn versiota 5.4, joka on julkaistu 28.7.2016.

Unityn monipuolisuus kehitysalustana näkyy myös sillä tehdyistä peleistä. Paljon näkyvyyttä saanut Blizzardin kehittämä korttipeli Heartstone: Heroes of Warcraft on kehitetty Unityllä ja pelin kehittäjät pitivät erityisesti iteraatioiden ja porttauksen helppoudesta (Unity Technologies, 2014). Campo Santos studion kehittämä Firewatch on hyvä esimerkki Unityn graafisesta kyvystä. Firewatchissa myös käytetään Unityn kauppa-alueelta saatavia lisäosia (Campo Santos Studio, 2016). Lisäosien käyttäminen on Unityllä kehitetyille peleille yleistä.

Unity sisältää neljä eri lisenssivaihtoehtoa, jotka sisältävät eri suuruisen kuukausiveloituksen. Personal-lisenssi on ilmainen ja se sisältää kaikki moottorin perustoiminnallisuudet. Jos kaupallisen henkilön tai yhtiön vuosittainen tuotto saavuttaa tai ylittää 100 000 Yhdysvaltain dollaria, tulee henkilön tai yhtiön hankkia Plus-lisenssi. Plus-lisenssi maksaa 35 Yhdysvaltain dollaria kuukaudessa yhtä henkilöä kohden. Jos vuosittainen tuotto saavuttaa tai ylittää 200 000 Yhdysvaltain dollaria tulee henkilön tai yhtiön hankkia Pro-lisenssi, joka maksaa 125 Yhdysvaltain dollaria yhtä henkilöä kohden. Lisäksi organisaatio voi hankkia Unityltä kustomoidun Enterprise-lisenssin, jonka hinta määräytyy sopimuksen mukaan. Kaikki lisenssivaihtoehdot ovat rojaltivapaita. (Unity Store 2016; Unity Technologies 2016b.)

Unityn nettisivut sisältävät oppimisvälilehden, jolta käyttäjä löytää apua ja oppaita moottorin opiskeluun. Opiskeluosio sisältää moottorin tutoriaalit, dokumentaation, yleisen tuen, suorat opetukset ja muut oppimisresurssit. Dokumentaatiossa on myös moottorin ohjelmointidokumentaatio. (Unity Technologies 2016c.) Apua käyttäjä voi saada myös Unityn foorumista ja vastausosiosta (Unity Technologies 2016d).

4.2 Unreal Engine

Unreal Engine on Epic Gamesin kehittämä pelimoottori, joka esiteltiin ensimmäisen kerran vuonna 1998 kun sitä käytettiin Unreal-nimisessä pelissä. Unreal oli ensimmäisen persoonan toimintapeli, mutta moottoria on sittemmin käytetty monien erilaisten pelityyppien toteuttamiseen. Unreal Enginestä on vuosien varrella ollut viisi suurempaa versiota. (Wikipedia 2016g.) Tässä opinnäytetyössä on käytössä Unreal Enginen versio 4.13. Unreal Enginen käyttöehdot kieltävät moottorin käyttämisen uhkapelaamiseen, ydinvoimaan tai lentoliikenteeseen liittyvissä sovelluksissa (Epic Games 2016a).

Vaikka Unreal Enginekin on monipuolinen moottori, tunnetaan se enemmän laajojen ja isojen studioiden työkaluna. Unreal Enginellä tehdyt Gears of War -sarjan pelit osoittavat hyvin moottorin kyvyn tuottaa näyttävää grafiikkaa konsoleilla. Mobiilipuolella etenkin kiinassa Unreal Engine 4:ää käytetään monissa peliprojekteissa (Epic Games 2016b).

Unreal Enginen lisenssiehtojen mukaisesti käyttäjä sitoutuu maksamaan Epicille 5 prosenttia moottorilla kehitetyllä sovelluksella saavutetuista tuloista, jotka kalenterineljänneksen aikana ylittävät 3000 Yhdysvaltain dollaria, muuten moottori on ilmainen käyttää. Unreal Engine ei sisällä eri ehdoilla olevia lisenssejä. (Epic Games 2016a.)

Unreal Enginen oppimisosio sisältää moottorin dokumentaation, videotutoriaaleja ja tietoa sisältävän wiki-osuuden. Dokumentaatio sisältää ohjelmointidokumentaation. Epic järjestää myös Unreal Engineä koskevia opetusstriimejä, jotka myös nauhoitetaan käyttäjien katsottavaksi myöhemmin. Muina apulähteinä käyttäjille toimii yhteisön foorumi ja vastaussivusto. (Unreal Engine 2016b.)

4.3 CryEngine

CryEngine on saksalainen Crytekin kehittämä pelimoottori. Ensimmäistä kertaa pelimoottoria käytettiin Crytekin kehittämässä pelissä nimeltä Far Cry. CryEnginestä on vuosien saatossa julkaistu viisi suurta versiota ja nykyinen versio on

nimeltään CryEngine V. (Wikipedia 2016h.) Tässä opinnäytetyössä käytetään moottorin versiota 5.2.1. CryEnginen käyttösopimus kieltää moottorin käyttämisen kaikkiin ei pelillisiin projekteihin sisältäen sotilaalliset projektit, uhkapelaamisen, peleihin liittymättömään simuloimisen, tieteen harjoittamisen, arkkitehtuurin, pornografian ja vakavat pelit (Crytek 2016a).

Kenties tunnetuin CryEnginellä kehitetty pelisarja on Crysis, jonka ensimmäinen osa sisälsi julkaisunsa aikaan ennennäkemätöntä peligrafiikkaa. Mobiilipelejä CryEnginellä ei voi kirjoitushetkellä tehdä. Indiepeli puolella CryEngineä ei ole ennen käytetty kovinkaan paljoa. CryEngineä käytetään tällä hetkellä monien pelien kehityksessä ja osa peleistä on indiekehitteisiä (CryEngine 2016a).

CryEngine on käyttäjälle täysin ilmainen ja moottorin lähdekoodi on käyttäjän saatavilla. Moottorilla tuotetuista peleistä ei tarvitse maksaa Crytekille rojaltimaksuja. CryEnginen kauppapaikalla myydyistä tuotteista 30%:ia tuotoista täytyy kuitenkin maksaa Crytekille. (Crytek 2016b.)

CryEngine sisältää kirjoitetun dokumentaation ja videotutoriaaliosion. Dokumentaatio sisältää myös ohjelmointidokumentaation. CryEnginen yhteisö kattaa avun muodossa foorumin ja vastaussivuston. (CryEngine 2016b.)

4.4 Lumberyard

Amazon Lumberyard on ilmainen AAA-tason pelimoottori. Lumberyard on vahvasti kytköksissä Amazonin verkkopalveluihin. Lumberyadin perusteknologia pohjautuu CryEngineen. Amazon kehittää kuitenkin Lumberyardia itsenäisesti eteenpäin, ja tästä syystä moottoreiden välille syntyy tulevaisuudessa eroja. Joitakin eroja moottoreissa oli jo Lumberyardin julkaisussa. (Nutt 2016.) Tässä opinnäytetyössä käytetään Lumberyardin 1.4 Beta-versiota. Lumberyard on moottorina niin uusi, että sillä ei ole kirjoitushetkellä vielä julkaistuja pelejä. Lumberyardilla voi kuitenkin tehdä tietokone-, konsoli- ja mobiilipelejä. Lumberyardin käyttöehdot kieltävät pelimoottorin käytön kriittisissä järjestelmissä, lääkintäjärjestel-

missä, automaattisen liikenteen hallinnassa, yksin kulkevissa ajoneuvoissa, lentoliikenteessä tai sen hallinnassa, ydinvoimalaitoksen hallinnassa, miehitetyissä avaruuslennoissa ja oikeaan toimintaan yhdistetyssä sotilaallisessa käytössä (Amazon Web Services 2016a).

Lumberyard on käyttäjälle täysin ilmainen ladata ja käyttää. Pelimoottori sisältää myös lähdekoodin ja se on käyttäjän muokattavissa. Amazonin tarjoamat AWS-palvelut ovat maksullisia, mikäli käyttäjä päättää käyttää niitä kehittämässään pelissä. (Amazon Web Services 2016b.)

Lumberyard sisältää kirjoitetun dokumentaation, joka sisältää oppaita alkuunpääsemiseen, kehittämiseen ja pelimoottorin käyttämiseen. Dokumentaatio sisältää myös ohjelmointidokumentaation lua-ohjelmointikielelle. (Amazon Web Services 2016c.) Dokumentaation lisäksi Lumberyard sisältää tutoriaaliosion, joka sisältää sekä kirjoitettuja että videotutoriaaleja (Amazon GameDev 2016).

4.5 Stingray

Stingray on Autodeskin omistama pelimoottori. Pelimoottori tunnettiin aiemmin nimellä Bitsquid, mutta Autodesk osti Bitsquidin ja on kehittänyt sitä eteenpäin Stingray-nimellä. Stingray on vahvasti kytköksissä Autodeskin kehittämiin muihin yleisesti peleissä käytettyihin sovelluksiin. Moottorin pyrkimyksenä on tavoittaa pienempiä kehittäjiä ja mahdollistaa ei-ohjelmoijien työskentely pelilogiikan parissa. (Collins 2015.) Tässä opinnäytetyössä käytetään Stingrayn versiota 1.5.

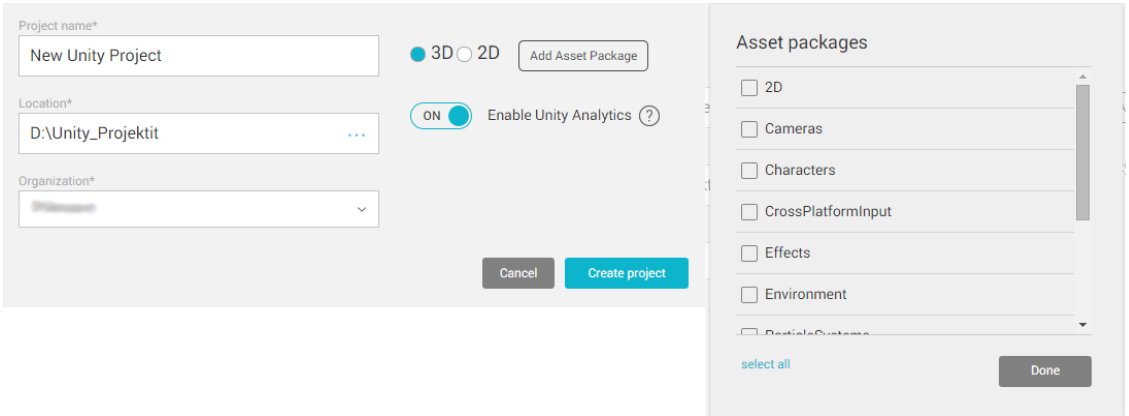
Samoin kuin Lumberyard on myös Stingray todella uusi pelimoottori. Stingraylla on kuitenkin jo julkaistu pelejä. Warhammer: End Times – Vermintide on Fatsharkin kehittämä ensimmäisen persoonan co-op-toimintapeli, joka sijoittuu Warhammerin ikoniseen fantasiamaailmaan (Fatshark 2016). Ylhäältäpäin kuvatun kolmannen persoonan toimintapeli Helldiversin kehittäjä valitsi Stingrayn sen helpon muokattavuuden takia (Autodesk 2016b).

Stingray on täysin rojaltimaksuton pelimoottori, mutta käyttölisenssi maksaa 30 Yhdysvaltain dollaria yhtä käyttölisenssiä kohden. Lisenssin voi myös ostaa kerralla yhdeksi, kahdeksi tai kolmeksi vuodeksi jolloin hinnat ovat 240, 455 ja 650 Yhdysvaltain dollaria. (Autodesk 2016c.) On myös huomionarvoista, että Stingray sisältyy Autodeksin Maya LT tilaukseen, jonka kuukausittainen tilaushinta on sama kuin Stingraylla (Autodesk 2016d).

5 Projektin aloitus ja editorin muokattavuus

5.1 Unity

Unityn asentamisen jälkeen uuden projektin aloitus on helppoa. Kuvasta 5 voidaan nähdä uuden projektin tekemisen yksinkertaisuus. Käyttäjä voi valita projektin nimen, tallennuspaikan, kehittäjäorganisaation, tehdäänkö 3D- vai 2D-projekti ja käytetäänkö Unityn analyysejä. ”Add Asset Package”-painikkeesta käyttäjä voi valita mitä sisältöpaketteja projektiin lisätään. Käyttäjä voi esimerkiksi valita käyttöön Characters-paketin, jolloin käyttäjällä on projektissa valmiina käytettävissä valmis pelaajahahmo. Kaikista näistä valinnoista huolimatta projektin vakiokenttä on aina tyhjä.

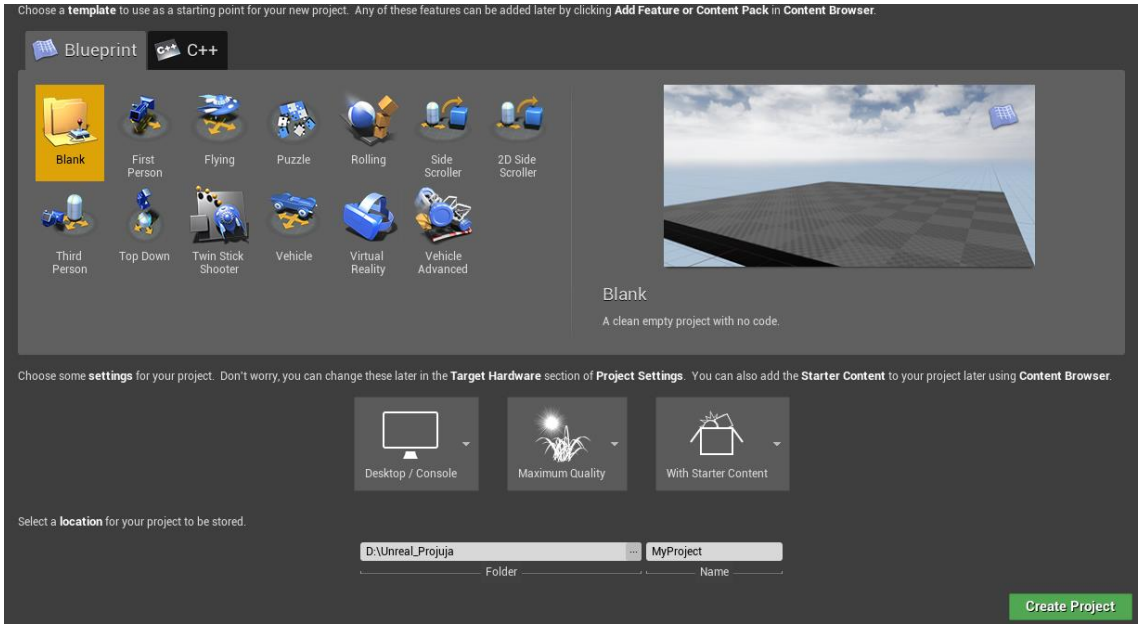


Kuva 5. Uuden projektin tekeminen Unityssä.

Unityn editorin kaikki ikkunat ovat käyttäjän vapaasti liikuteltavissa ja niiden koko on myös käyttäjän muutettavissa. Unityn käyttäjä voi myös laajentaa editorin ominaisuuksia ohjelmakoodilla (Unity Documentation 2016a).

5.2 Unreal Engine

Uuden projektin tekeminen Unreal Enginessä on pääpiirteissään helppoa. Kuvasta 6 näkyy käyttäjälle mahdolliset valinnat. Unreal Engine tarjoaa käyttäjälle paljon erilaisia pohjia projektin toteutukselle sekä blueprintpohjaisena, että myös koodipohjaisena. Valitusta pohjasta riippuen projektin aloituskenttä on hieman erilainen ja sisältää yleensä liikuteltavan valmiin hahmon, jonka pohjalta kehitystä on helppo lähteä viemään eteenpäin. Käyttäjä voi myös valita, tekeekö projektin tietokoneelle/pelikonsolille vai onko tavoitteena mobiilipeli, käytetäänkö grafiikoissa parhaita asetuksia vai ovatko ne skaalautuvat ja sisällytetäänkö projektiin Unrealin tarjoamia aloitussisältöjä.

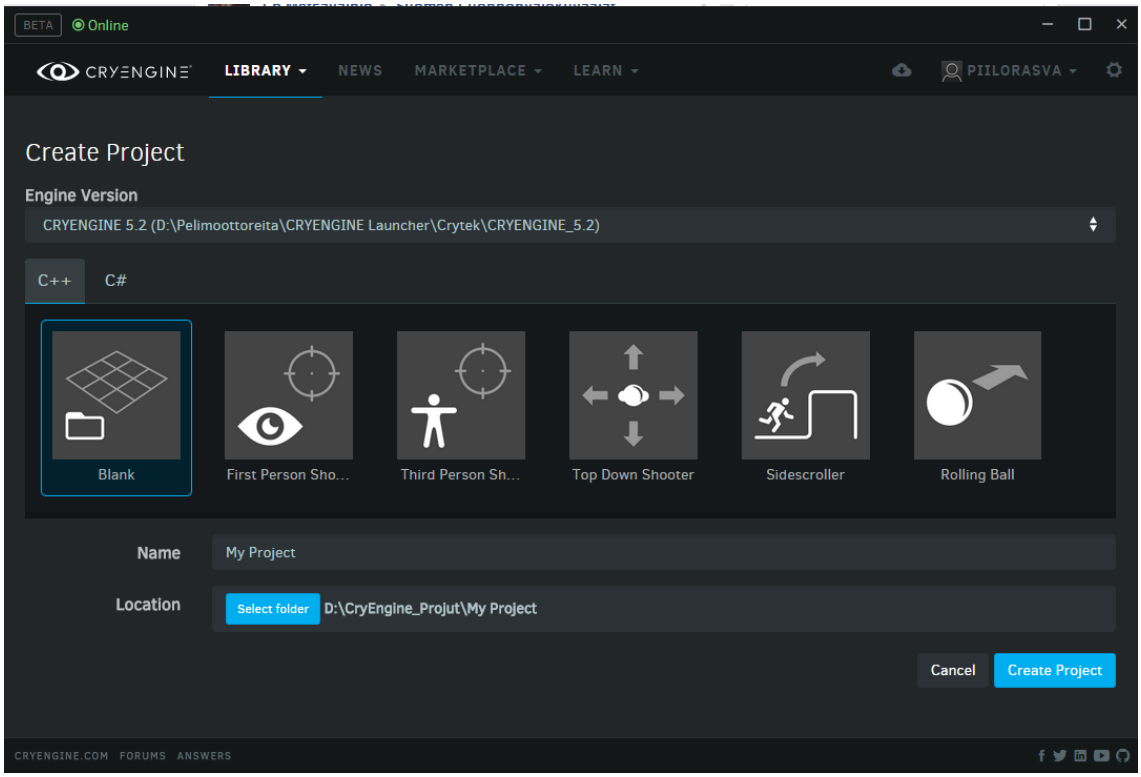


Kuva 6. Uuden projektin tekeminen Unreal Enginessä.

Myös Unreal Engine editorin ikkunoiden koko ja paikka ovat käyttäjän muokattavissa. Editoria voi myös muokata omiin käyttötarkoituksiinsa (Unreal Engine Documentation 2015). Editorin muokkaukselle ei löydy kuitenkaan niin hyviä ja kattavia ohjeita kuin Unityn editorin muokkaukselle.

5.3 CryEngine

CryEngine sisältää myös erityylisiä pohjia joista käyttäjä voi valita itselleen parhaiten sopivan. Valittavat pohjat ovat nähtävissä kuvassa 7. Valittavat pohjat ovat tulleet CryEngineen vasta viimeisimmässä päivityksessä, ja ne ovat vielä varsin rajoittuneita ja tarjoavat hyvin vähän toiminnallisuutta. Käyttäjä voi valita C++-projektin sijaan tekevänsä myös C#-pohjaisen projektin. C#-projektiksi käyttäjä voi valita tyhjän projektin tai vaihtoehtoisesti valmiin sivusta kuvatun pelin.



Kuva 7. Uuden projektin tekeminen CryEnginessä.

CryEngine V:n myötä CryEngineen tuli myös uusi editori. Uuden editorin ikkunoiden koko ja paikka ovat täysin käyttäjän muokattavissa. CryEnginen dokumentaatiosta ei löydy kohtaa, joka neuvoisi kuinka editoria laajennetaan. CryEnginen moottorin lähdekoodi on kuitenkin vapaasti saatavilla, joten todella edistyneeltä käyttäjältä editorin muokkauksen pitäisi onnistua.

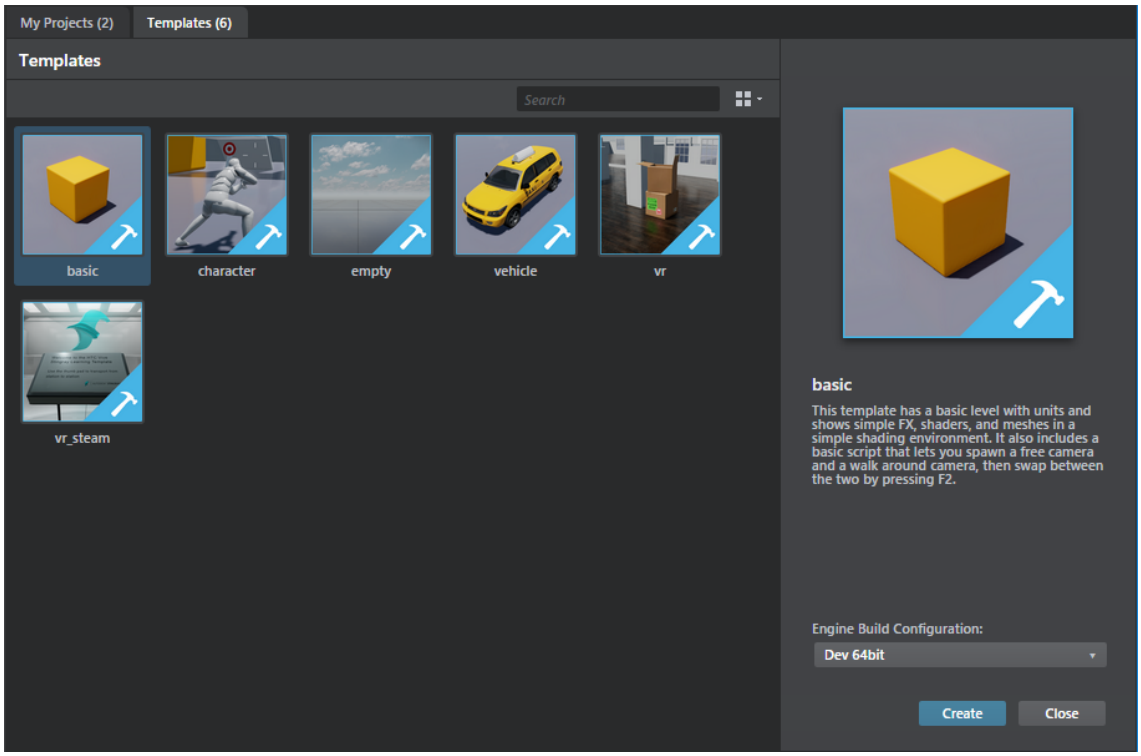
5.4 Lumberyard

Uuden projektin tekemiseksi Lumberyardilla käyttäjän täytyy ensin suorittaa setup assistant -ohjelma, joka tarkastaa ja ohjeistaa asentamaan moottorin toiminnalle välttämättömät ohjelmat. Setup assistantista voi avata projektien hallinnan, jonka kautta käyttäjä voi luoda uuden projektin. Uutta projektia luotaessa käyttäjä ei voi valita muuta kuin projektin nimen. Seuraavaksi projekti tulee asettaa oletusprojektiksi. Tämän jälkeen käyttäjän tulee komentokehoitteen kautta käyttää ”lmbr_waf configure” -komentoa, joka tekee tarvittavat asetukset oletusprojektiin. Tämän jälkeen peliprojekti pitää vielä kasata käytettäväksi. (Lumberyard Documentation 2016a.)

Editorin ikkunoiden koko ja paikka ovat täysin käyttäjän muokattavissa. Lumberyardin dokumentaatiosta ei löydy kirjoitushetkellä ohjeita editorin ominaisuuksien laajentamiseen. Moottorin lähdekoodi on kuitenkin täysin muokattavissa, joten edistyneeltä käyttäjältä muokkauksen pitäisi onnistua.

5.5 Stingray

Kuvasta 8 voi nähdä Stingrayn projektin luontiin liittyvän ikkunan ja tarjolla olevat pohjat. Stingrayn valmiit pohjat sisältävät hyvän kattauksen ominaisuuksia ja hyvän pohjan lähteä kehittämään omaa projektia. Pohjia soisi silti olevan enemmänkin. Pohjan lisäksi käyttäjän valittavissa on konfiguraation bittisyys, projektin nimi, tallennuspaikka ja kuvaus.



Kuva 8. Uuden projektin tekeminen Stingrayssa.

Singrayn editorin ikkunoiden koko ja paikka ovat täysin käyttäjän muokattavissa. Stingrayn editoria voi laajentaa lähdekoodin kautta, siihen ei kuitenkaan löydy ohjeita dokumentaatiosta, eikä sitä suositella tekemään kirjoitushetkellä tulevaisuuden yhteensopivuusongelmien vuoksi (Stingray Forum 2016).

5.6 Yhteenveto

Editorien käytettävyys ja ikkunoiden muokattavuus on kaikissa vertailun moottoreissa hyvällä tasolla ja ulkoasu pienen totuttelun jälkeen selkeä. Editorin laajennettavuudeltaan Unity on hyvän dokumentaation ja helppouden ansiosta paras, muissa moottoreissa editorin laajentaminen on hankalampaa tai sitä ei suositella.

Uuden projektin tekeminen on suoraviivaista kaikissa muissa moottoreissa paitsi Lumberyardissa. Lumberyard on vasta beta-vaiheessa, mikä näkyy joissakin asioissa joiden soisi olevan jo automatisoituja. Unreal Engine tarjoaa käyttäjälle monipuolisimmat aloituspohjat, joiden avulla erityylisiä pelejä on helppo alkaa tehdä. Unity tarjoaa käyttäjälle valmiita peliobjekteja, joiden avulla erilaisten pelien kehityksen aloitus nopeutuu, mutta ne vaativat käyttäjältä enemmän vaiheita

kuin Unreal Enginen pohjamalli. Stingrayn pohjat ovat myös erittäin hyviä, mutta niitä on harmillisesti vain muutama. CryEnginen tarjoamat pohjat tuntuvat ominaisuuksiensa puolesta vielä hieman keskeneräisiltä. Lumberyard ei tarjoa käyttäjälle mitään valmiita pohjia.

6 Peliobjektien tuonti moottoriin ja tuetut formaatit

Tässä luvussa luodaan katsaus kunkin vertailtavan moottorin tukemiin objektiformaatteihin sekä siihen, kuinka nämä objektit saadaan tuotua pelimoottorin sisään käytettäväksi pelimaailmassa. Yhteenveto-osiossa on taulukko (Taulukko 1), josta tuetut formaatit käyvät selkeästi ilmi. Moottorikohtaisesti arvioidaan työkalujen ja sisällön tuonnin helppokäyttöisyyttä, monipuolisuutta sekä dokumentaation kattavuutta.

6.1 Unity

Peliobjektien tuonti Unityyn onnistuu usealla eri tavalla. Käyttäjä voi tallentaa haluamansa tiedoston suoraan Unityn ”Assets”-kansioon, siirtää haluamansa tiedoston ”Assets”-kansioon tai raahata tiedoston Unityn projekti-ikkunaan. Unityn ”Assets”-kansiossa olevan tiedoston muokkaus päivittyy suoraan Unityn sisälle. Tuonnin yhteydessä Unity luo tuotuun objektiin liittyvän vakiona käyttäjälle näkymättömän meta-tiedoston, joka sisältää tuotuun objektiin liittyvää tietoa siitä, kuinka sitä käytetään pelissä. Turvallisin tapa poistaa objekteja pelistä on poistaa ne suoraan Unityn projektikansiosta. (Unity Documentation 2016b.) Neljäs tapa tuoda objekteja Unityyn on klikata hiiren oikeaa painiketta pelimoottorin ”Assets”-kansiossa ja valita ”Import new asset”.

Animoitujen objektien tuonti Unityyn onnistuu samalla tavalla kuin muidenkin objektien tuonti. Animointitiedostot voivat olla irrallisia objektista, jolloin ne täytyy tuoda moottoriin erillisinä tai ne voivat tulla moottoriin suoraan peliobjektin mukana. Unityyn tuodusta animaatiosta voi myös leikata useita animaatioita omiksi

tiedostoiksi ja samaa animaatiota voi tarvittaessa käyttää useaan erilliseen objektiin. (Unity Documentation 2016c.)

Fysiikkalaskuihin käytettävä collider-objekti Unityssä voidaan luoda tuodulle objektille objektin omasta muodosta tai vaihtoehtoisesti colliderina voidaan myös käyttää alkukantaisempia muotoja (Unity Documentation 2016d). Unity ei sisällä käyttäjän valittavissa olevia algoritmeja colliderin luonnille objektin omasta muodosta. Käyttäjä voi kuitenkin luoda colliderin yksinkertaisemmasta objektista monimutkaisemmalle objektille.

Staattisten ja animoitujen objektien tuonti Unityyn on todella helppoa ja suoraviivaista. Tuontivaiheessa käyttäjän tarvitsee vain tietää, että tiedostomuoto jota ollaan tuomassa, on tuettu. Tuodun tiedoston tuontiasetuksia ei tarvitse valita heti tuontivaiheessa, vaan niitä voi muuttaa jälkikäteen. Unityn dokumentaatio kaikkien peliobjektien ja muiden tiedostojen tuonnille on todella hyvä ja kattava. Dokumentaatio käy läpi tarvittavat perusteet ja sisältää hyvät selitykset kaikille tuontiin liittyville vaihtoehdoille.

6.2 Unreal Engine

Unreal Enginessä sisällön tuonti moottoriin tapahtuu pitkälti samalla tavalla kuin Unityssä. Unityyn verrattuna Unreal sisältää kuitenkin erillisen ”Import”-painikkeen, joka ajaa kuitenkin saman asian kuin muutkin tavat. Muita tapoja ovat tiedostojen vetäminen suoraan projektikansioon tai editorin ”Content”-ikkunaan ja ”Import”-valinnan käyttäminen, kun ”Content”-ikkunaa on klikattu hiiren oikealla painikkeella (Unreal Engine Documentation 2016a). Unreal Enginen sisällön tuontiasetukset ovat erittäin monipuoliset ja sisältävät paljon säätöjä.

Animoitujen objektien tuonti on Unreal Enginessä helppoa ja se onnistuu samalla tavalla kuin muidenkin sisältöjen tuominen moottoriin. Animaatiot voi tuoda objektin mukana yksittäisinä tiedostoina, leikata paloiksi yhdestä pidemmästä tiedostosta tai tuoda moottoriin kokonaan omina tiedostoinaan (Unreal Engine Documentation 2016b).

Staattisille objekteille collideria luotaessa Unrealissa voi käyttää primitiivisiä muotoja, erilaisia algoritmeja tai luoda colliderin suoraan objektin muodosta (Unreal Engine Documentation 2016c). Animoiduille objekteille colliderin teko on hieman erilaista. Animoitua objektia täytyy klikata hakemistossa hiiren oikealla napilla ja valita ”Create Physics Asset” (Unreal Engine Documentation 2016d). ”Create Physic Asset”-valinnan jälkeen käyttäjä voi muokata animoidun objektin fyysistä olemusta Unrealin ”Physics Asset”-työkalun kautta.

Kokonaisuutena Unreal Enginen sisällöntuonti ja siihen liittyvät työkalut ovat erittäin helppokäyttöiset. Moottori sisältää myös erinomaiset tuonti- ja muokkausvalinnat staattisille ja animoiduille objekteille. Unrealin sisällön tuontiin liittyvä dokumentaatio on myös varsin kattava. Dokumentaatio käy läpi kaikki perusteet ja sisältää myös hyvät selitykset kaikille tuontivalinnoille.

6.3 CryEngine

CryEngineen staattisten ja animoitujen objektien tuonnissa on kaksi vaihtoehtoa. CryEngine on luotu tukemaan vahvasti Autodeskin mallinnus- ja animointiohjelmistoja, ja tästä syystä muista ohjelmista tuodut mallit ja animaatiot on ollut hankala saada toimimaan ennen CryEnginen versiota 5.2. Paras tapa käyttäjälle tuoda objektit moottoriin on ollut käyttää CryEngine exporter lisäosaa Autodeskin sovelluksissa (CryEngine Documentation 2016a). CryEnginen versio 5.2 toi mukanaan moottoriin päivitetyn Fbx importer -työkalun, jolla onnistuu staattisten objektien tuonti (CryEngine 2016b) ja animoitujen objektien tuonti (CryEngine 2016c).

CryEnginessä colliderien luominen objekteille tapahtuu sisällönluontityökaluissa. Collider luodaan objekteille omana geometrianaan, ja se pitää nimetä tarkan nimeämiskäytännön mukaan. Collideria kutsutaan CryEnginessä proxyksi ja se tarvitsee myös oman materiaalin muiden materiaalien ohella. (CryEngine Documentation 2016b.)

CryEnginen versio 5.2 paransi sisällön tuonnin monipuolisuutta moottoriin merkittävästi uuden Fbx importer -työkalun myötä. Ennen tätä työkalua omien mallien saaminen moottoriin on ollut vaivalloista, ellei ole omistanut maksullisia Autodeskin sovelluksia. Autodeskin sovelluksia käsittelevä dokumentaatio on hyvällä tasolla, mutta Fbx importerista ei löydy kirjoitushetkellä kirjoitettuja ohjeita, vaan ohjeistus on muutaman Youtube-videon varassa.

6.4 Lumberyard

Lumberyardin pohjautuminen CryEngineen näkyy myös sisällöntuonnissa. Paras ja suositelluin tapa on käyttää Autodeskin sovelluksia ja Lumberyardin Setup Assistantilla asennettavia export-työkaluja (Lumberyard Documentation 2016b). Ongelmana joillekin Autodeskin sovelluksissa tuottaa varmasti näiden ohjelmistojen maksullisuus. Lumberyard sisältää myös kokeiluasteella olevan FBX Importer työkalun, jolla käyttäjä voi tuoda moottoriin staattisia FBX-tiedostoja (Lumberyard Documentation 2016c).

Colliderien luonti Lumberyardissa tapahtuu Autodeskin sovelluksissa samalla tavalla kuin CryEnginessä. FBX Importeria käyttäessä collider voidaan valita tuotavaksi mistä tahansa tuotavasta mallista (Lumberyard Documentation 2016c).

Lumberyardin tuki tuotaville staattisille ja animoiduille objekteille on tällä hetkellä huono ellei käyttäjä omista Autodeskin DCC-sovelluksia. Tulevaisuudessa FBX Importteria kuitenkin todennäköisesti päivitetään tukemaan myös animoituja objekteja, jolloin tilanne paranee merkittävästi. Dokumentaatio käsittelee tämän osion kohtalaisesti; collidereiden tekemiseen soisi olevan paremmat ohjeet.

6.5 Stingray

Autodeskin Stingrayn saa vahvasti kytkettyä Autodeskin sisällöntuotantosovelluksiin moottorin lisäosalla. Lisäosan asentamisen ja asetusten tekemisen jälkeen peliobjekteja ja niiden materiaaleja voi siirtää reaaliajassa pelimoottorin ja

sisällöntuotantosovellusten välillä. Tämä nopeuttaa sisällöntuotantoa, koska enää materiaaleja ja malleja ei tarvitse erikseen viedä sisällöntuotantosovelluksesta ja tuoda pelimoottoriin. (Stingray Help 2016b.) Stingrayhin voi kuitenkin tuoda sisältö myös perinteisemmällä tavalla. Stingray tukee natiivisti FBX-tiedostomuotoa staattisten ja animoitujen objektien tuomista varten. Sisältöä voi tuoda käyttämällä moottorista löytyvää ”Import”-painiketta, käyttämällä hiiren oikealla napilla avautuvaa valikkoa tiedostoselaimessa tai raahaamalla tiedoston suoraan moottorin tiedostoselaimeen. (Stingray Help 2016c.)

Collidereiden luomiseksi Stingrayssa tulee käyttää Unit editoria. Unit editorissa objektille luodaan fyysinen muoto. Fyysisenä muotona voi käyttää mallia, primitiivistä objektia tai generoitua mallia. (Stingray Help 2016d.)

Kokonaisuutena staattisten ja animoitujen objektien tuonti Stingrayhin on hyvällä tasolla. Reaaliaikainen päivitys Stingrayn ja Autodeskin sisällöntuotantosovellusten välillä on todella mielenkiintoinen ominaisuus ja tämän ominaisuuden soisi olevan myös muissa pelimoottoreissa tuettuna jollakin tapaa. Muussa kuin Autodeskin sovelluksissa luodun FBX-tiedoston tuonti moottoriin onnistuu vaivattomasti. Dokumentaatio objektien tuontiin ja hallintaan on Stingrayssä hyvä. Dokumentaatio käy hyvin läpi kaikki perusteet sisällön tuonnista. Lisädokumentaatio fysiikoiden asettamisesta olisi kuitenkin toivottavaa.

6.6 Yhteenveto

3D-tiedostotuen osalta pelimoottorit jakautuvat kahteen ryhmään. Unity, Unreal Engine ja Stingray tukevat natiivisti FBX-tiedostomuotoa staattisille ja animoiduille objekteille. CryEngine ja CryEngineen pohjautuva Lumberyard puolestaan käyttävät cgf- ,chr- ja skin-tiedostomuotoja. CryEngineen on kuitenkin jo julkaistu FBX Importer -työkalu, jolla FBX-tiedostomuodon voi muuntaa moottorin tukemiin muotoihin. Lumberyardissa tilanne ei ole aivan yhtä hyvä. Lumberyardiin on julkaistu myös FBX Importer -työkalu, mutta se tukee tällä hetkellä vain staattisten objektien muuntamista moottorin ymmärtämään muotoon. Taulukkoon 1 on koottu moottoreiden tukemat tiedostomuodot.

Taulukko 1. Pelimoottoreiden tukemat 3D-tiedostomuodot.

	Tuetut tiedostomuodot 3D-malleille
Unity	.FBX .dae .3DS .dxf .obj
Unreal Engine	.FBX
CryEngine	.cgf .chr .skin
Lumberyard	.cgf .chr .skin
Stingray	.FBX

Mallien tuonnin helppoudessa Unity, Unreal Engine ja Stingray ovat samalla tasolla. Näissä kolmessa moottorissa tuontiasetuksia on myös helppoa ja yksinkertaista muokata. Tuodun mallin uudelleen tuonnissa Stingray on pelimoottoreista paras, mikäli käyttäjä käyttää Autodeskin mallinnus- ja animointi-ohjelmistoja. CryEngine jää hieman jälkeen Unitystä, Unreal Enginestä ja Stingraysta. Erillisen työkalun käyttö lisää hieman mallin tuontiaikaa moottoriin. Lumberyardissa staattisten objektien tuonti on työvaiheiltaan samanlainen kuin CryEnginessä, mutta animoitujen mallien tuonnissa on helpointa, jos käyttäjä omistaa maksullisen Autodeskin ohjelman.

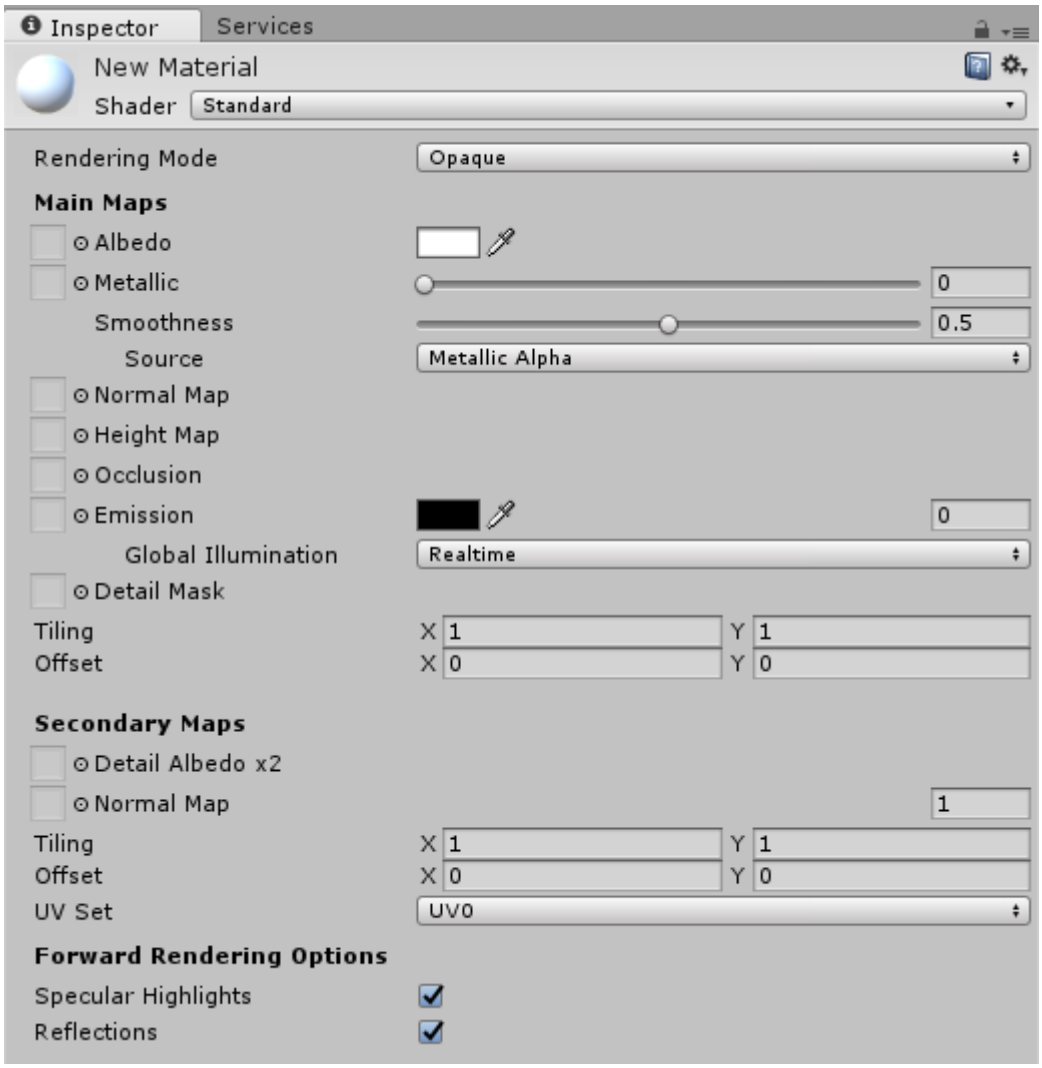
7 Materiaalien luonti ja muokkaus

Objektien tuonnin jälkeen malleille luodaan materiaali. Materiaali käyttää tekstuureita, numeroarvoja ja shaderia halutunlaisen ulkonäön luomiseksi objektille. Materiaalien luonnista ja muokkauksesta pelimoottoreiden kesken vertaillaan luonnin helppoutta, muokattavuuden laajuutta ja dokumentaation kattavuutta.

7.1 Unity

Materiaalin luominen Unityssä on helppoa. ”Assets”-ikkunassa hiiren oikella napilla aukeavasta valikosta valitaan ”Create” ja tästä aukeavasta valikosta ”Material”. Unityssä luodut materiaalit käyttävät vakiona Standard shaderia. Standard shader on Unityssä tarkoitettu yleiskäyttöiseksi shaderiksi ja se on ohjelmoitu mallintamaan valon käyttäytymistä reaalimaailmaan mukaisesti. Unityssä on myös muita sisäänrakennettuja shadereita erikoistuneisempiin käyttötarkoituksiin. (Unity Documentation 2016e.)

Unityssä materiaaleja voi muokata antamalla materiaalin ominaisuuksille tekstuureja, numeerisen valinnan tai ennalta määrätyn vaihtoehdon. Materiaalin muokattavat ominaisuudet vaihtelevat valitun shaderin mukaisesti. (Unity Documentation 2016e.) Materiaalin muokkausikkuna on selkeä ja kaikki muokattavat ominaisuudet ovat helposti nähtävillä (kuva 9).



Kuva 9. Unityn Standard shaderin muokattavat ominaisuudet.

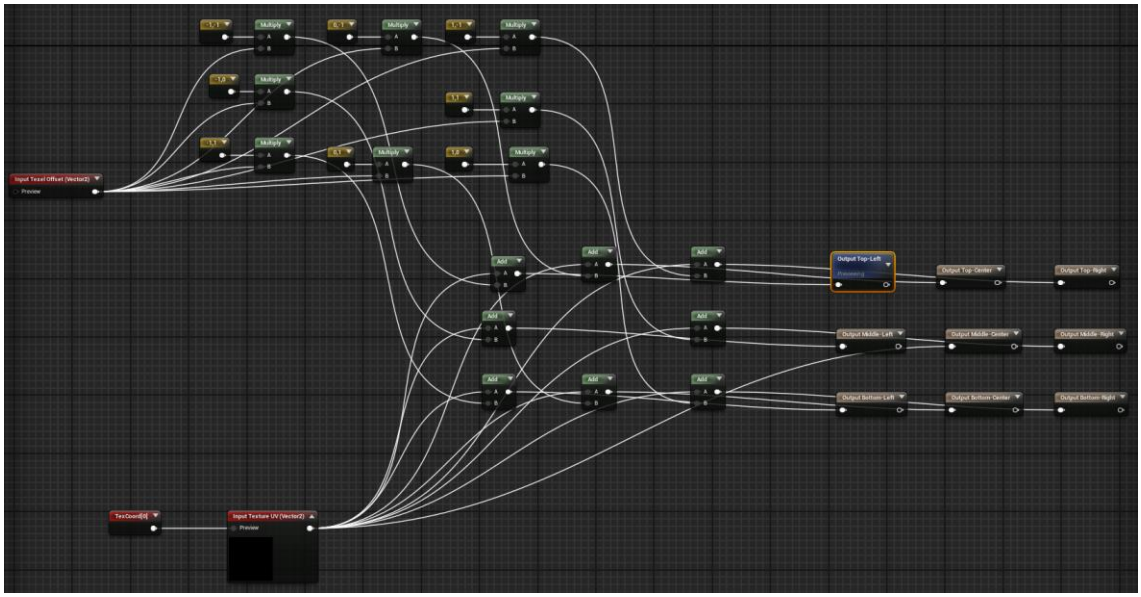
Unityssä käyttäjä voi myös kirjoittaa omia shadereita kolmella erillaisella tavalla. Kaikki tavat vaativat vähintään hieman ohjelmointiosaamista (Unity Documentation 2016e). Ohjelmoinnin tarve omia efektejä materiaaliin tehtäessä voi olla monelle käyttäjälle iso kynnys. Unityn dokumentaatio materiaalien teon, muokkauksen ja shadereiden ohjelmoinnin osalta on todella kattava. Dokumentaatio käy läpi kaikki perusteet ja sisältää myös syvällistä tietoa shadereiden toiminnasta.

7.2 Unreal Engine

Materiaalin luominen Unreal Enginessä on yhtä helppoa kuin Unityssä. Käyttäjän tarvitsee vain painaa vihreää ”Add new” -painiketta ja valita ”Material” tai vaihtoehtoisesti klikata hiiren oikealla napilla jonkin kansion sisällä content browserissa

ja valita avautuvasta valikosta ”Material”. Unreal Enginessä materiaalien muokkaus onnistuu visuaalisella ohjelmoinnilla ja se on node-pohjaista. Nodeina materiaalin muokkauksessa Unreal Enginessä käytetään materiaali-ilmaisuja, laskutoimituksia ja lukuarvoja. Käyttäjän rakentama nodeista rakentuva verkko käännetään shader-koodiksi ja se on myös käyttäjän nähtävillä. Unreal sisältää seitsemän valmista shaderia käytettäväksi ja materiaalin muokattavat ominaisuudet vaihtelevat valitun shaderin mukaisesti. (Unreal Engine Documentation 2016e.)

Materiaaleja varten Unreal Enginessä voi tehdä erillisiä materiaalifunktioita, joilla materiaaleihin voi lisätä monipuolista toiminnallisuutta. Funktion tekemisen jälkeen sitä voi käyttää monessa eri materiaalissa. Materiaalifunktio toteutetaan samalla visuaalisen ohjelmoinnin periaattella kuin materiaalin tekeminen. Reaaliajassa materiaalien muokkaukseen ilman materiaalin uutta kääntämistä Unreal Enginessä voi tehdä materiaali instansseja. Materiaali instanssien käyttämiseksi käyttäjä voi materiaalia muokatessa luoda nodeja, jotka toimivat parametreina materiaali instanssia muokattaessa. (Unreal Engine Documentation 2016e.) Materiaalifunktiot voivat olla isojakin kokonaisuuksia ja niissä voidaan tehdä paljon laskutoimituksia (Kuva 10).



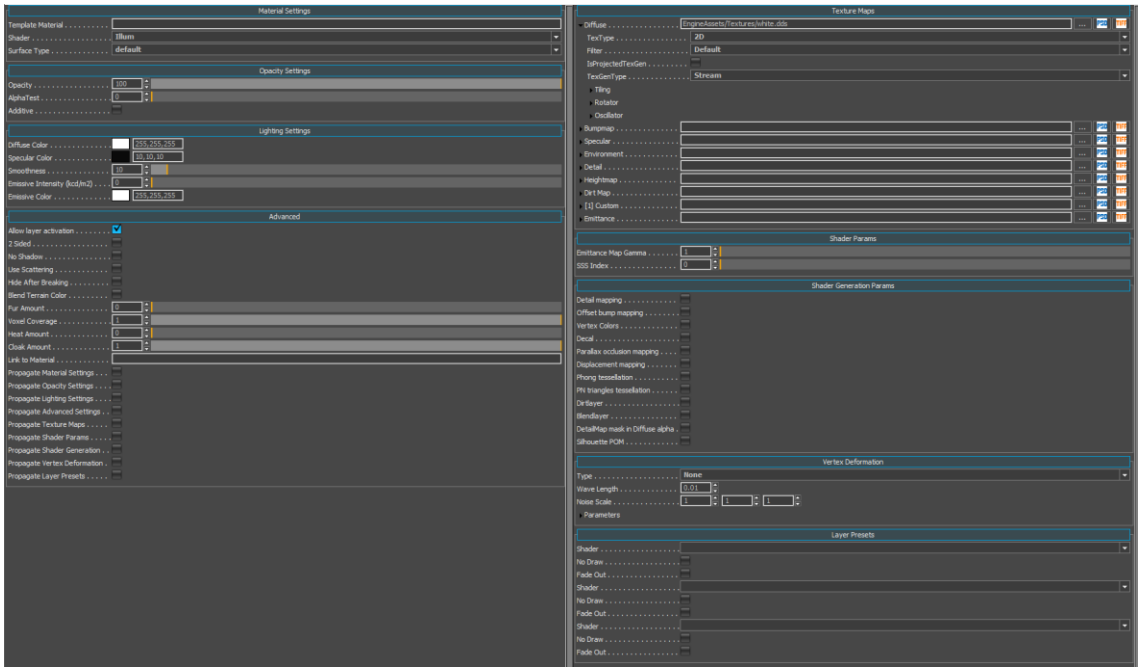
Kuva 10. Tekstuurin uv-koordinaatteja vertaileva materiaalifunktio Unreal Enginessä.

Dokumentaatioltaan Unreal Engine kattaa materiaalien luomisen ja käyttämisen todella kattavasti. Dokumentaatiossa on myös paljon esimerkkejä materiaalien ominaisuuksien käyttötarkoituksista ja käyttötapauksia materiaalifunktioille ja instansseille.

7.3 CryEngine

Materiaalin luomiseen ja käyttöön CryEnginessä on kaksi tapaa. Ensimmäinen tapa on käyttää DCC-ohjelmaa materiaalin luontiin ja muokata tätä luotua materiaalia CryEnginen Material Editorilla. Toinen tapa on käyttää ainoastaan CryEnginen Material Editoria materiaalin luontiin ja muokkaukseen. CryEngine sisältää 29 valmista shaderia käytettäväksi. Materiaalin muokattavat ominaisuudet vaihtelevat valitun shaderin mukaisesti. (CryEngine Documentation 2013a.) CryEngine käyttää fysiikkaan pohjautuvaa valaistusta shadereissa (CryEngine Documentation 2015).

CryEnginen Material Editorista löytyvät muokkausmahdollisuudet ovat erittäin monipuoliset (Kuva 11). Materiaalien muokkaus on perusteiltaan samanlaista kuin Unityssä, mutta muokattavia asioita on enemmän. Erityisen huomionarvoista on se, että videomateriaalin käyttö vaatii CryEnginessä GFxVideo-lisenssin (CryEngine Documentation 2015b).

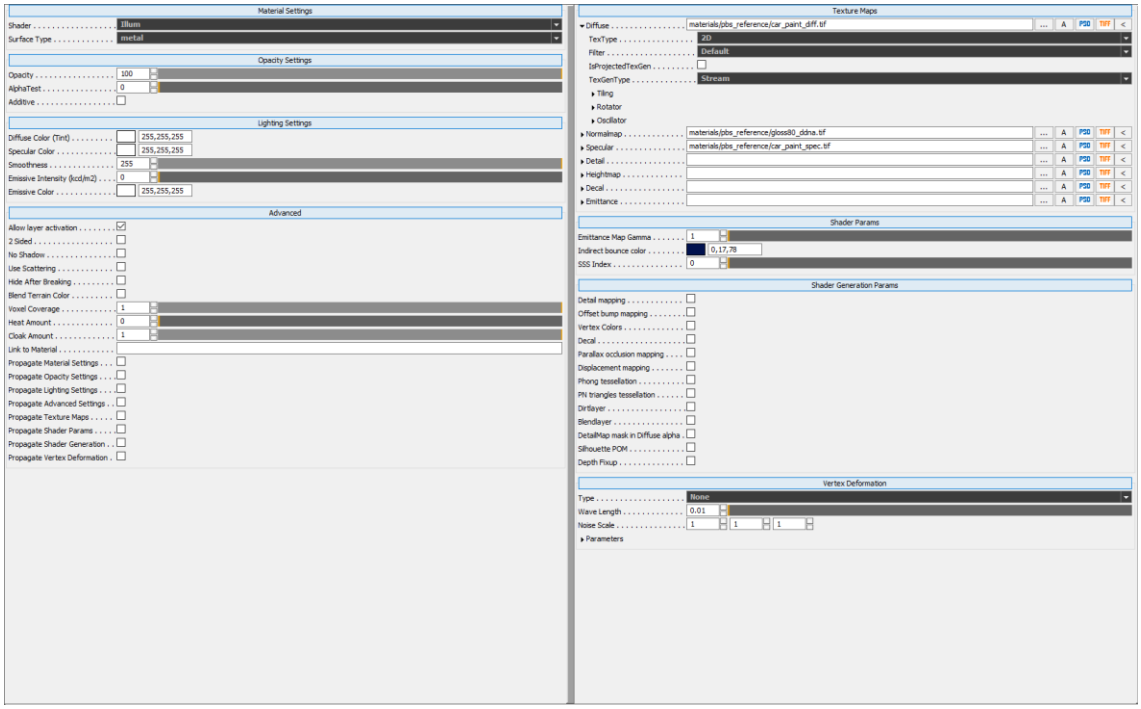


Kuva 11. CryEnginen Material Editorissa muokattavissa olevat ominaisuudet, kun käytössä on yleiskäyttöinen Illum-shader.

Materiaalien luonnin ja muokkauksen CryEnginen dokumentaatio käsittelee hyvin. Dokumentaatio kattaa kaikki perusteet materiaalin luonnin ja käytön osalta. Dokumentaatio sisältää myös paljon hyviä esimerkkejä eri tyyppisten tekstuurien käyttötarkoituksista sekä materiaalien ominaisuuksien vaikutuksista. Shadereiden muokkaukseen ja luomiseen dokumentaatio ei kirjoitushetkellä tarjoa käyttäjälle mitään apua. CryEngineen on kuitenkin mahdollista luoda omia shadereita (CryEngine Answers 2016).

7.4 Lumberyard

Lumberyardin materiaalien luonti ja muokkaus on käytännössä identtistä CryEngineen verrattuna. Lumberyardin Material Editor on ulkoasultaan ja käytettävyydeltään samanlainen kuin CryEnginen vastaava. (Lumberyard Documentation 2016d.) Ainoa ero Lumberyardin ja CryEnginen materiaalin ominaisuuksien muokkauksen välillä on editorin taustaväri (Kuva 11; Kuva 12). Valmiita shadereita Lumberyard sisältää hieman vähemmän kuin CryEngine (Lumberyard Documentation 2016e). Omien shadereiden luonti on Lumberyardissa mahdollista ja dokumentaatiossa on siihen kohtalaiset ohjeet (Lumberyard Documentation 2016f).

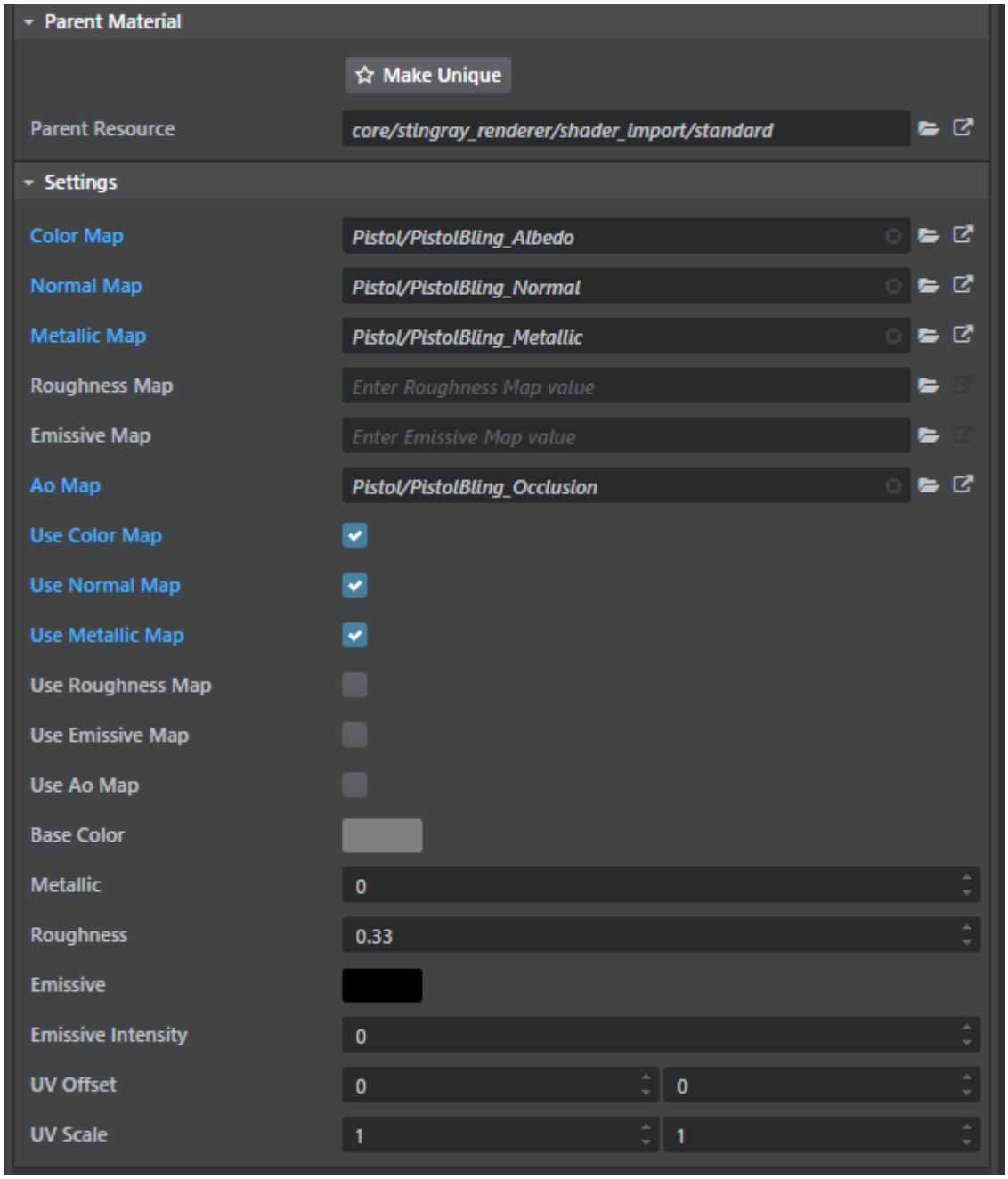


Kuva 12. Lumberyardin Material Editorin muokattavat ominaisuudet kun käytössä on yleiskäyttöinen Illum-shader.

Materiaalien muokkauksessa ja luonnissa Lumberyard ja CryEngine ovat hyvin samanlaisia ja ne sisältävät vastaavat toiminnallisuudet. Perusteet Lumberyardin dokumentaatiossa käydään läpi kohtalaisesti. Dokumentaation soisi kuitenkin sisältävän enemmän esimerkkejä materiaalien ominaisuuksien käytöstä. Käyttäjän omien shadereiden luomista varten dokumentaatiossa on oma osionsa.

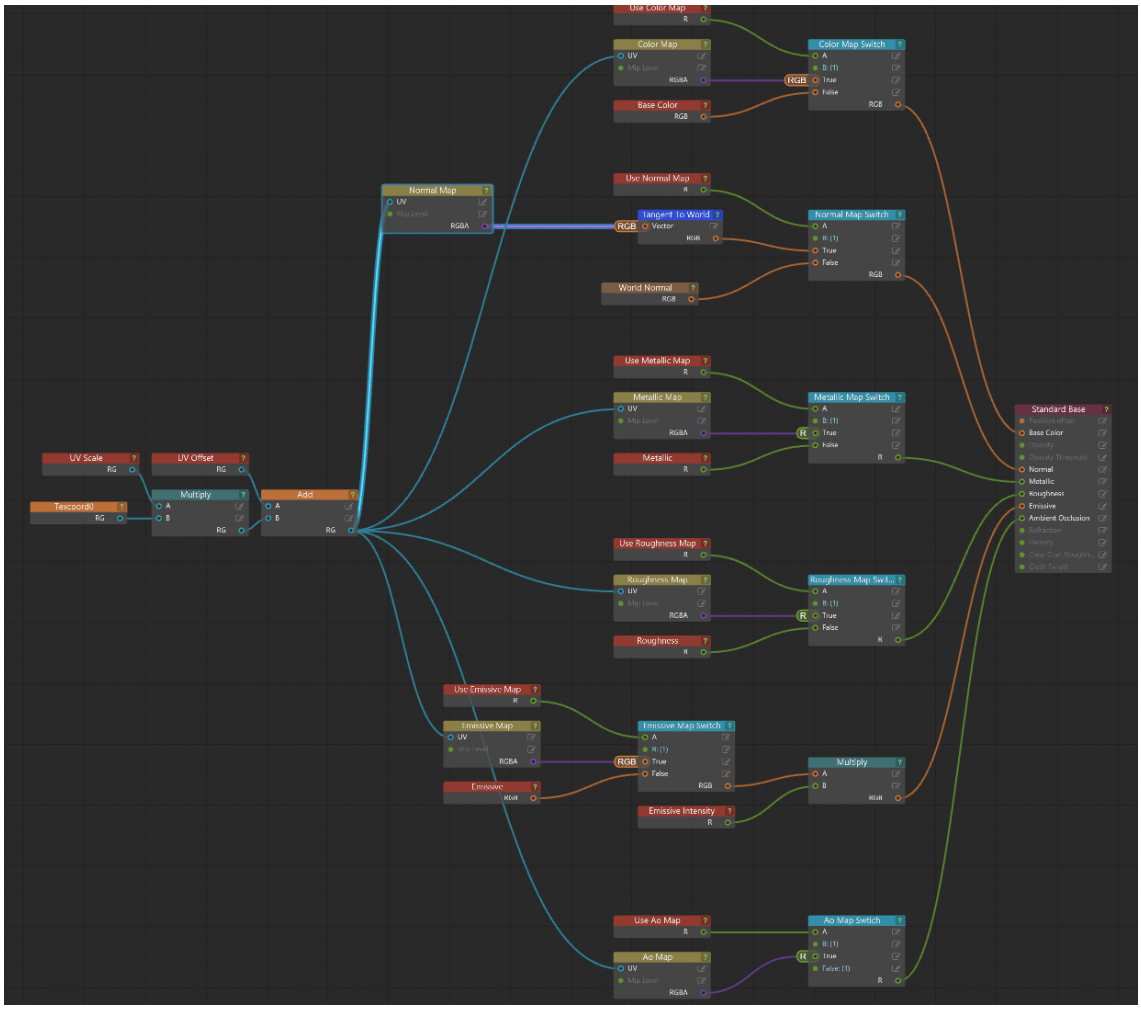
7.5 Stingray

Stingrayssa materiaalin luonti moottorin sisällä on suoraviivaista. Käyttäjä voi luoda materiaalin valitsemalla ”Create->Material” hiiren oikealla napilla avautuvasta valikosta moottorin tiedostoikkunassa. Luodun materiaalin voi asettaa käytettäväksi mille tahansa mallille. Materiaalin voi myös tuoda mallin mukana ulkoisesta DCC-ohjelmasta. (Stingray Help 2016e). Stingrayn materiaalien muokattavat ominaisuudet ovat hyvin samanlaiset kuin Unityssä (Kuva 13; Kuva 9).



Kuva 13. Stingrayssa luodun materiaalin ominaisuuksien muokkaus.

Stingrayn materiaaleilla on hierarkkinen vanhempimateriaali. Vanhempimateriaali on käytännössä shader, joka määrittelee materiaalin visuaalisen ulkoasun. Stingray sisältää useita valmiita materiaaleja. Vanhempimateriaalia muokataan node-pohjaisen käyttöliittymän kautta ja käyttäjä voi myös tehdä omia vanhempimateriaaleja. (Stingray Help 2016f.) Stingrayn vanhempimateriaalin muokkaus on hyvin samankaltaista kuin Unreal Enginen materiaalin muokkaus (Kuva 14; Kuva 10).



Kuva 14. Stingrayn vanhempimateriaalin muokkaus on node-pohjaista.

Stingrayn materiaalidokumentaatio on kohtalainen. Se käsittelee materiaalin luonnin ja materiaalin perusominaisuudet hyvin. Myös omien vanhempimateriaalien luonti on käsitelty, mutta hieman pintapuolisesti. Dokumentaation myös soisi sisältävän enemmän esimerkkejä erilaisten materiaalien luomiseen. Stingrayn dokumentaatio ei käsittele sitä, kuinka PBR-materiaaleja voidaan luoda ja tuoda Autodeskin mallinnusohjelmista. Tutoriaalit löytyvät kuitenkin ohjelmistojen omista dokumentaatioista (Autodesk Maya Documentation 2016).

7.6 Yhteenveto

Materiaalien luominen kaikkien vertailtavien pelimoottoreiden sisällä on helppoa, suurimmat erot löytyvät luotujen materiaalien muokkauksesta. Materiaalin muokkaus on Unityssä, CryEnginessä, Lumberyardissa ja Stingrayssa perusteiltaan

hyvin samanlaista. CryEnginessä ja Lumberyardissa muokattavia ominaisuuksia on kuitenkin materiaalilla enemmän. Unreal Engine on ainut vertailtavista moottoreista, jonka materiaalin muokkaus on täysin node-pohjaista. Materiaalien muokkauksen paremmuuteen vaikuttaa ennen kaikkea henkilökohtainen mielipide. Kaikki vertailtavat moottorit käyttävät PBR-shadereita, jotka helpottavat yhtenäisen ulkonäön luomista.

Dokumentaation osalta Unity, Unreal Engine ja CryEngine ovat erittäin hyviä ja ne sisältävätkin perusteiden lisäksi myös syvällisempää tietoa ja hyviä esimerkkejä. Stingrayn ja Lumberyardin dokumentaatiot käyvät perusteet hyvin läpi, mutta eivät sisällä kovinkaan syvällistä tietoa. Lumberyardia käyttäessä voi kuitenkin hyödyntää myös CryEnginen dokumentaatiota, koska työkalut moottoreissa ovat niin samanlaisia. Uusi käyttäjä pääsee kuitenkin alkuun kaikkien moottoreiden dokumentaatiolla.

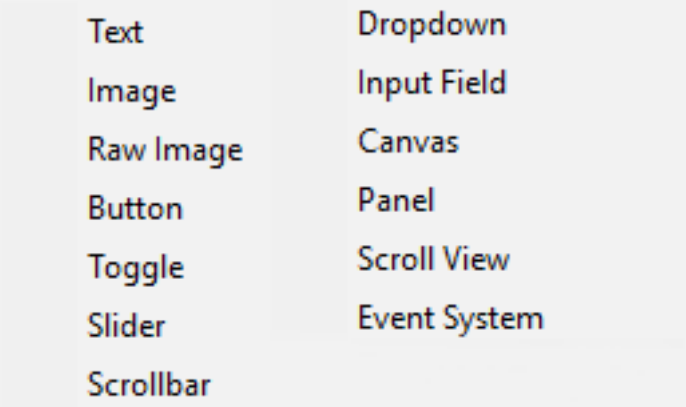
8 Käyttöliittymän toteutus

Jossakin vaiheessa kehitystä kehitettävään peliin on suunniteltava ja toteutettava graafinen käyttöliittymä. Graafista käyttöliittymää käytetään informaation kertomiseen pelaajalle. Tässä osiossa luodaan katsaus kunkin pelimoottorin käyttöliittymänluontityökaluun, ja mikäli moottorista ei itsessään tällaista työkalua löydy, luodaan katsaus moottorin tukemiin ulkoisiin työkaluihin. Samalla arvioidaan myös dokumentaation kattavuutta.

8.1 Unity

Unity sisältää sisäänrakennetun graafisen käyttöliittymän luontityökalun. Työkalulla voi luoda käyttöliittymän avaruudellisesti näyttötilassa tai maailmatilassa. Työkalu sisältää komponentteja, joilla saadaan rakennettua käyttöliittymän eri osiot sekä työkaluja, joilla käyttöliittymä saadaan skaalautumaan sopivaksi eri resoluutiota käyttäville laitteille. Kaikille interaktiivisille komponenteilla on tapahtuma, johon kehittäjä voi liittää oman toiminnallisuutensa. (Unity Documentation

2016f.) Kuvassa 15 näkyvät kaikki Unityn sisäänrakennetun käyttöliittymätyökalun komponentit.

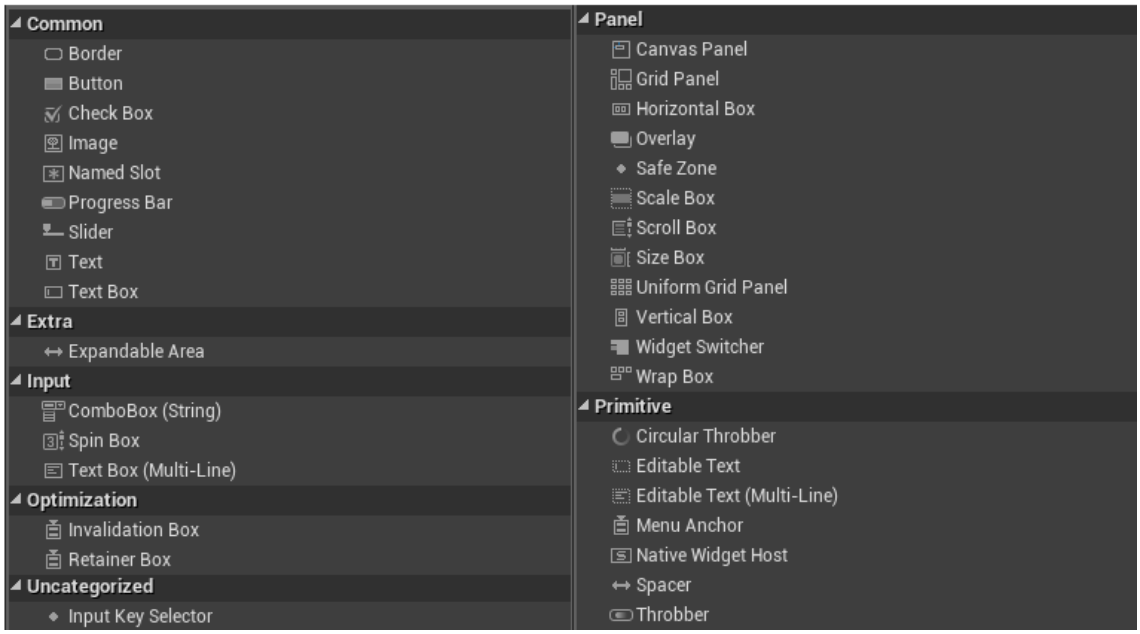


Kuva 15. Unityn graafisen käyttöliittymän kaikki mahdolliset komponentit.

Unityn dokumentaatio kattaa käyttöliittymän luonnin hyvin. Kirjoitettu dokumentaatio sisältää hyvän läpikäynnin käyttöliittymän perusteille, mutta ei opasta kuinka käyttöliittymän komponentteja hallitaan C#-koodin kautta (Unity Documentation 2016f). Unityn tutoriaaliosasto sisältää osion, joka käsittelee hyvin käyttöliittymän luonnin ja interaktiivisuuden koodin kautta (Unity Tutorials 2016a).

8.2 Unreal Engine

Unreal Engine sisältää UMG-nimisen (Unreal Motion Graphics) työkalun, jota käytetään graafisen käyttöliittymän luontiin pelimoottorin sisällä. UMG rakentuu kahdesta osasta. Suunnitteluosiossa käyttöliittymän komponentit asetellaan oikeille paikoilleen. Kaaviosivulla käyttöliittymän komponentteihin voi ohjelmoida toimintalogiikkaa visuaalisena ohjelmointina. (Unreal Documentation 2016f.) Kuvassa 16 on nähtävillä kaikki Unreal Enginen UMG-komponentit.



Kuva 16. Unreal Enginen kaikki UMG-komponentit.

Unreal Enginen UMG:n peruskäytön ja visuaalisen ohjelmoinnin dokumentaatio on erittäin kattava. Dokumentaatio ei kuitenkaan sisällä hyviä ohjeita UMG:n hyödyntämiseen C++-koodin kautta. C++-koodin hyödyntämiseen on kuitenkin hieman ohjeita Unreal Enginen käyttäjien tekemien ohjeiden muodossa (Unreal Engine Wiki 2015a; Unreal Engine Wiki 2015b).

8.3 CryEngine

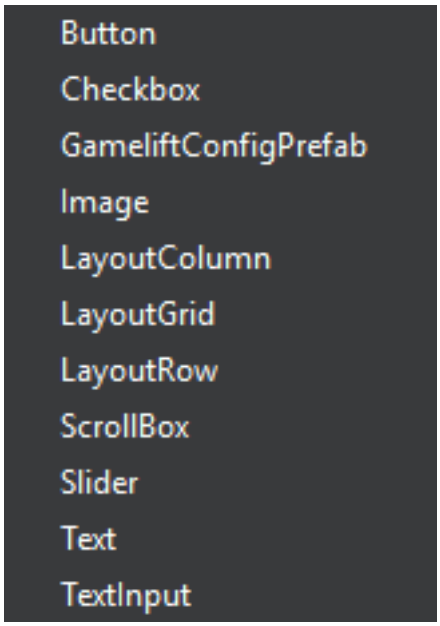
CryEngine sisältää käyttöliittymän luomista varten ohjelmointikirjaston vain C#-projektissa. Sisäänrakennetun työkalun sijaan C++-projekteissa CryEngine käyttää Autodeskin Scaleform alustaa käyttöliittymän luontiin ja hallitsemiseen. Scaleform on flash-pohjainen väliohjelmisto. Scaleformia hyödyntäen luotu käyttöliittymä koostuu kolmesta isommasta osasta: käyttöliittymäelementeistä jotka luodaan flashilla, käyttöliittymätoiminnoista jotka määrittelevät kuinka käyttöliittymä käyttäytyy ja käyttöliittymän tapahtumanhallintajärjestelmästä, joka hallitsee tapahtumien käsittelyn pelimoottorin ja käyttöliittymän välillä. (CryEngine Documentation 2014a.)

Käyttöliittymän luomiseen Flashilla käyttäjä tarvitsee erillisen ohjelmiston. CryEnginen tutoriaaleissa käytetään Adoben maksullista Flashia (CryEngine Documentation 2014b) tai ilmaisia Vectoria Giotto- ja FlashDevelop-ohjelmistoja (CryEngine Documentation 2014c). On myös huomattavaa, että CryEnginellä käyttöliittymää luotaessa kehittäjän tulee osata xml-kieltä käyttöliittymän elementtien luomiseen (CryEngine Documentation 2014a).

Kokonaisuutena käyttöliittymän luonti Scaleformia hyödyntäen on paljon sekavampaa ja monivaiheisempaa kuin esimerkiksi Unityn ja Unreal Enginen sisäänrakennettujen työkalujen käyttäminen. CryEnginen dokumentaatio on käyttöliittymien luonnin osalta kohtuullinen ja se käy perusteet kohtalaisesti läpi. Dokumentaatio kaipaisi kuitenkin paljon enemmän esimerkkejä ja valmiita ratkaisuja, joita kehittäjä voisi soveltaa omiin projekteihin. C#-projekteissa käytettävän käyttöliittymän dokumentaatio on huono sisältäen hyvin vähän esimerkkejä tai perusohjeita (CryEngine Documentation 2016c). Käyttöliittymän luomista Flashilla hankaloittaa myös se, että CryEngine ei käytä uusinta Scaleformin versiota. Tämä rajoittaa käyttöliittymän logiikan ohjelmoinnin Actionscript 2 -ohjelmointikieleen uudemman Actionscript 3:n sijaan. (CryEngine Forum 2012.) Actionscript 2:lle on hankala löytää hyviä tutoriaaleja ja jotkut Flash-ohjelmistot eivät enää tue sitä.

8.4 Lumberyard

Lumberyardissa on moottoriin sisäänrakennettu työkalu käyttöliittymän toteuttamiselle. Työkalulla voi toteuttaa avaruudellisesti näyttötilassa olevia interaktiivisia käyttöliittymiä sekä ei-interaktiivisia käyttöliittymiä, joita voi käyttää tekstuureina pelimaailmassa sijaitsevissa objekteissa. Käyttöliittymän elementtejä voi hallita Lumberyardin visuaalisella flowgraph-ohjelmoinnilla tai lua-ohjelmointikielellä. (Lumberyard Documentation 2016g.) Kuvassa 17 näkyvät kaikki Lumberyardin sisältämät käyttöliittymäelementit.



Kuva 17. Lumberyardin UI Editorin elementit.

Lumberyardin dokumentaatio kattaa UI Editorin perusteet hyvin ja dokumentaatio käy myös elementtien ominaisuuksia hyvin läpi. Luodun käyttöliittymän logiikan ohjelmoinnin dokumentaatio on puolestaan vielä vajaa. Dokumentaatio sisältää vain läpikäynnin visuaalisessa ohjelmoinnissa käytettäviin nodeihin. Tutoriaaliosio sisältää vain kolme kirjoitettua ohjetta siitä, kuinka luotua käyttöliittymää voidaan käyttää lua-kielen avulla (Lumberyard Tutorials 2016). Käyttöliittymien luonti ja elementtien asettelu on Lumberyardissa helppoa. Logiikan ohjelmointi ja käyttöliittymän integrointi itse peliin on hankalaa vajaan dokumentaation takia.

8.5 Stingray

Autodeskin Stingray pelimoottori sisältää tuen Autodeskin Scaleform Studio -ohjelmistolle ja Stingray hyödyntää Scaleform Studiota suoraan käyttöliittymän toteuttamiseen. Scaleform Studiolla voi tehdä käyttöliittymiä myös ilman pelimoottoria, mutta silloin kaikki Scaleform Studion ja Stingrayn yhtenäiset ominaisuudet eivät ole käytettävissä. Luotuja käyttöliittymiä voi käyttää tekstuureina pelimaailman objekteissa. Luotuihin käyttöliittymiin kehittäjä voi ohjelmoida logiikan visuaalisella ohjelmoinnilla käyttäen Stingrayn Flow-ohjelmointia tai vaihtoehtoisesti kehittäjä voi käyttää lua-ohjelmointikieltä. (Stingray Help 2016g.)

Stingrayn oma dokumentaatio kattaa käyttöliittymän luonnin kohtalaisesti. Dokumentaatio ohjeistaa käyttäjän Scaleform Studion dokumentaatioon, joka käsittelee Scaleformin käyttämisen laajemmin (Stingray Help 2016g; Scaleform Help 2016). Työkaluna Scaleform Studio on monipuolinen ja perusteiltaan yksinkertainen. Integraatio Stingrayhin on toteutettu luontevasti ja perusteiden oppimisen jälkeen uusia käyttöliittymiä on nopea tehdä.

8.6 Yhteenveto

Käyttöliittymä voidaan luoda pelimoottoreissa kahdella eri tavalla: pelimoottori joko sisältää sisäänrakennetun työkalun käyttöliittymän luomiselle tai tuen kolmannen osapuolen sovellukselle. Unity ja Unreal Engine ovat työkaluiltaan hyvin lähellä toisiaan. Molempien moottoreiden työkaluissa on samat perustoiminnallisuudet, mutta Unreal Enginessä käyttöliittymän logiikan ohjelmoinnissa on enemmän vaihtoehtoja. Lumberyardin ratkaisu on toteutukseltaan samanlainen kuin Unityssä ja CryEnginessä, mutta on vielä käytettävyydeltään rajoittuneempi. CryEngine ja Stingray molemmat käyttävät Flash-pohjaista Scaleformia käyttöliittymien luomiseen. Stingrayn tuki on kuitenkin huomattavasti parempi, koska moottori sisältää suoran tuen Scaleform Studiolle. Käyttöliittymän toteuttamiseksi CryEnginelle kehittäjä tarvitsee erillisen ulkopuolisen ohjelmiston tai ohjelmistoja.

Dokumentaatioltaan Unity ja Unreal Engine ovat todella hyviä ja perusominaisuuksien läpikäynnin lisäksi molemmat sisältävät käyttöliittymiin liittyviä tutoriaaleja. CryEnginen dokumentaatio on vajaa Scaleformin tarjoamiin ominaisuuksiin nähden ja käyttöliittymän rakentamisen aloittaminen vaatii enemmän opettelua kuin muilla moottoreilla. Lumberyardin dokumentaatio on moottoreista huonoin, koska se sisältää hyvin vähän tietoa siitä, kuinka käyttöliittymän logiikka ohjelmoidaan. Stingrayn Scaleform Studion dokumentaatio on kohtuullinen, dokumentaatiota kuitenkin täydentää Scaleform Studion oma dokumentaatio.

9 Logiikan ohjelmointi

Tässä osiossa luodaan katsaus vertailtavien pelimoottoreiden tarjoamiin logiikan ohjelmoinnin tapoihin ja ohjelmointikieliin. Tavoista ja kielistä vertaillaan niiden käytettävyyttä sekä aloittajaystävällisyyttä. Samalla luodaan katsaus pelimoottorin dokumentaation kattavuuteen.

9.1 Unity

Unity käyttää Mono-kehitysympäristöä ja tulkittavia ohjelmointikieliä kehitettävän pelin logiikan ohjelmointiin. Natiivisti Unity tukee C#- ja JavaScript-pohjaista UnityScript-ohjelmointikieltä. Unityssä voi kuitenkin käyttää monia muita .NET-kielillä toteutettuja DLL-tiedostoja. (Unity Documentation 2016g.)

Skriptitiedostojen käyttäminen Unityssä on komponenttipohjaista. Käytännössä tämä tarkoittaa sitä, että scriptitiedoston luotuaan käyttäjä voi lisätä tiedoston käytettäväksi niin monelle peliobjektille kuin käyttäjä haluaa. Luodut tiedostot on natiivisti peritty Unityn MonoBehaviour-luokasta. MonoBehaviour tarjoaa käyttäjälle pohjan, joka sisältää pelilogiikan kannalta tärkeimmät toiminnallisuudet. Editorissa näkyvien ja muokattavien muuttujien kirjoittaminen on Unityssä helppoa. (Unity Documentation 2016g.)

Unityn logiikan ohjelmoinnin dokumentaatio on erittäin kattava. Dokumentaatio käy kattavasti läpi Unityn perusrakenteet skriptauksesta. Unity myös tarjoaa käyttäjälle runsaasti oppaita logiikan toteuttamiseen (Unity Tutorials 2016b; Unity Live Training 2016). Logiikan ohjelmointi Unityssä on erittäin nopeaa, koska Unity käyttää tulkittavia kieliä. Tulkittavien kielien johdosta ohjelmakoodia ei tarvitse kääntää, mikä säästää kehittäjältä paljon aikaa ja mahdollistaa nopeat koodimuutokset.

9.2 Unreal Engine

Unreal Engine tarjoaa käyttäjälle kaksi eri vaihtoehtoa pelilogiikan ohjelmointiin (Unreal Engine Documentation 2016g). Ensimmäinen tapa on visuaalinen ohjelmointi, jota kutsutaan Unreal Enginessä Blueprint-ohjelmoinniksi. Blueprint-ohjelmointi on node-pohjaista skriptauskieltä, joka on helppo oppia ja joka tarjoaa hyvän työkalun suunnittelijoille ohjelmointilogiikan toteuttamiseen. (Unreal Engine Documentation 2016h.) Toinen tapa toteuttaa logiikka on C++-ohjelmointi. Unreal Enginen C++-ohjelmointi eroaa hieman perinteisestä C++-ohjelmoinnista, sillä se sisältää Unreal Engineen liittyviä ominaisuuksia jotka helpottavat kielen käyttämistä ja oppimista. Ohjelmointia helpottaa myös Unreal Enginen Class Wizard -työkalu, jonka avulla käyttäjä voi luoda uusia pohjaluokista periytyviä luokkia. Pohjaluokat tarjoavat käyttäjälle tarvittavia perustoimintoja. (Unreal Engine Documentation 2016g.)

Blueprint- ja C++-ohjelmointia voi käyttää myös yhdessä. Ohjelmoija voi käyttää C++-kieltä perusluokan tekemiseen, joka tarjoaa tarvittavat toiminnallisuudet luokan käyttämiseen. Luokkaa voi sen jälkeen laajentaa ja käyttää Blueprint-ohjelmoinnilla. Unreal Enginessä Blueprint tarkoittaa myös peliobjektia, joka on luotu C++-luokasta tai suoraan valikosta. Valikosta luotaessa Blueprint-objekteilla ei ole muokattavaa C++-toiminnallisuutta. (Unreal Engine Documentation 2016g.)

Unreal Enginen ohjelmointidokumentaatio on erinomainen. Dokumentaatio sisältää erittäin kattavan läpikäynnin ohjelmoinnin eri osa-alueisiin ja se sisältää paljon koodiesimerkkejä. Bluperint-ohjelmointi käydään dokumentaatiossa myös todella kattavasti läpi ja Blueprint-ohjelmointia käytetään paljon esimerkeissä. Unreal Enginessä C++-ohjelmoinnin aloittaminen on helppoa, mikäli tietää ohjelmoinnin perusteet. Negatiivisena puolena C++-kielessä on, että koodin kirjoittamisen jälkeen koodi täytyy kääntää. Kääntäminen voi kestää pahimmillaan useita minuutteja.

9.3 CryEngine

CryEngine tarjoaa käyttäjälle neljä erilaista tapaa toteuttaa ohjelmalogiikka. Flow Graph -ohjelmointi on CryEnginen visuaalinen ohjelmointityökalu, jonka suurin etu on sen helppokäyttöisyys (CryEngine Documentation 2016d). Flow Graph -ohjelmointi on perusteiltaan vastaavaa kuin Unreal Enginen Blueprint-ohjelmointi. Monimutkaisemman logiikan ja järjestelmien ohjelmoinnissa CryEnginessä voi käyttää C++-kieltä (CryEngine Documentation 2016e). C++-kielen sijaan CryEnginessä voi myös käyttää Mono-kehitysympäristön päälle rakennettua CE#-kehikkoa, jolloin ohjelmointi onnistuu C#-kielellä (CryEngine Documentation 2016f). C++-kielen rinnalla CryEnginessä voi käyttää tulkattavaa lua-ohjelmointikieltä. Lua-kielen kautta käyttäjä voi käsitellä tapahtumia, pelilogiikkaa ja C++-kielellä luotuja funktioita. (CryEngine Documentation 2014d.)

Ohjelmointi CryEnginessä voi olla erittäin sekavaa sekä kokemattomalle, että myös kokeneelle ohjelmoijalle. Iso osa sekavuudesta johtuu huonosta dokumentaatiosta. Dokumentaatio sisältää todella vähän esimerkkejä yksittäisistä ohjelmointitavoista tai ohjelmointitapojen käyttämisestä rinnakkain. Toisin kuin Unreal Enginessä, CryEnginen C++-kieli ei sisällä avustuksia tai luokan luontityökalua. Tämä hidastaa ohjelmointia ja sen oppimista. C#-ohjelmointi on moottorissa uusi ominaisuus, tämän johdosta C#-kielellä ei voi vielä toteuttaa kaikkea samaa toiminnallisuutta kuin C++-kielellä. Flow graph -ohjelmointi on periaatteiltaan ja käytännön toteutukseltaan hyvin samanlaista ja helposti opittavaa kuin Unreal Enginen Blueprint-ohjelmointi.

9.4 Lumberyard

Lumberyardin pohjautuminen CryEngineen tulee ilmi myös logiikan ohjelmoinnissa. Lumberyard sisältää saman visuaalisen ohjelmoinnin mahdollistavan Flow Grap -työkalun, joka on myös CryEnginessä (Lumberyard Documentation 2016h). Tämän lisäksi Lumberyardissa logiikkaa voi ohjelmoida käännettävällä C++-kielellä tai tulkattavalla lua-kielellä (Lumberyard Documentation 2016i). Lumberyard on kuitenkin luopumassa CryEnginessä käytetystä peliobjektien

luontitavasta (Lumberyard Documentation 2016j), ja siirtyy käyttämään komponenttipohjaisia peliobjekteja (Lumberyard Documentation 2016k). Lumberyard sisältää lua-kielellä ohjelmointia varten sisäänrakennetun editorin (Lumberyard Documentation 2016l).

Komponenttipohjaisten peliobjektien luonti on suurin eroavaisuus Lumberyardin ja CryEnginen välillä. Lumberyardin komponenttipeliobjektit mahdollistavat ominaisuuksien lisäämisen peliobjektiin nopeasti ja komponenttiobjektin logiikan ohjelmointi onnistuu lua-kielellä. Käyttäjä voi luoda omia komponentteja peliobjektien käyttöön C++-kielellä. (Lumberyard Documentation 2016k.)

Lumberyardin logiikan ohjelmoinnin dokumentaatio on kohtalainen. Dokumentaatio käsittelee perusasiat jokaisesta tavasta toteuttaa logiikka, mutta se ei sisällä paljoa esimerkkejä. Lumberyardia varten on tutoriaalisivusto, mutta senkin sisältö kattaa ohjelmoinnin pintapuolisesti (Lumberyard Tutorials 2016). Ohjelmointitavan valitsemista hankaloittaa Lumberyardin uusi komponenttipohjainen peliobjektien luonti. Varmaa kuitenkin on, että uusi komponenttipohjainen toteutustapa korvaa vanhan peliobjektien luontitavan kokonaan jossakin vaiheessa (Lumberyard Documentation 2016k).

9.5 Stingray

Stingrayssa logiikan ohjelmointiin on kaksi vaihtoehtoa. Ensimmäinen vaihtoehto on visuaalinen node-pohjainen ohjelmointi, jota Stingrayssa kutsutaan Flow-ohjelmoinniksi. Toinen vaihtoehto on käyttää lua-kieltä logiikan skriptaukseen. Flown etuina ovat helppo tapahtumiin reagointi, helppokäyttöisyys, nopeiden prototyyppien tekeminen ja peliobjektien pitäminen itsenäisinä. Laajojen kokonaisuuksien tekeminen pelkällä Flow-ohjelmoinnilla on kuitenkin haastavaa. Lua-skriptauksen etuina ovat laajemmat ominaisuudet, helpompi logiikan ohjelmointi, tehokkuus ja parempi virheiden etsiminen ja hallinta. Lua on kuitenkin hankalampaa oppia kuin Flow. (Stingray Help 2016h.) Stingrayta voi laajentaa C++-koodilla ja C++-koodilla luotuja ominaisuuksia voi käyttää Flow- ja lua-ohjelmoinnin kautta. C++-ohjelmointia varten ei löydy kuitenkaan dokumentaatiota ja sitä ei pidetä pääasiallisena kehitystapana. (GameFromScratch 2016.)

Stingrayn logiikan ohjelmoinnin dokumentaatio on hyvä. Dokumentaatio sisältää kattavan läpikäynnin käytettäviin ohjelmointitapoihin ja siihen miksi nämä ohjelmointitavat on pelimoottoriin valittu (Stingray Help 2016i). Ohjelmointitutoriaaleja Stingrayta varten ei löydy paljoa. Stingrayn dokumentaatio sisältää videotutoriaali-välilehden ja Autodeskillä on Youtube-kanava, joka käsittelee Stingray pelimoottoria (Stingray Help 2016j). Ohjelmointitutoriaalien määrä etenkin dokumentaatiossa on kuitenkin vähäinen.

9.6 Yhteenveto

Ohjelmointitavat vertailtavissa pelimoottoreissa jakautuvat kolmeen ryhmään. Unity, CryEngine, Lumberyard ja Stingray tukevat tulkattavia skriptauskieliä. Unreal Engine, CryEngine, Lumberyard ja Stingray sisältävän helppokäyttöisen visuaalisen ohjelmoinnin mahdollistavan työkalun ja Unreal Engine, CryEngine ja Lumberyard mahdollistavat ohjelmoinnin käännettävällä C++-kielellä.

Unityssä ohjelmoinnin aloittaminen on tehty käyttäjän kannalta helpoksi. Helpoksi aloittamisen tekee Unityssä käytetty tapa toteuttaa ohjelmointilogiikka. Helppoudesta suurimmassa vastuussa on hyvä dokumentaatio, joka kattaa ohjelmoinnin sekä aloittelijan, että edistyneen käyttäjän näkökulmasta. Unreal Enginen visuaalinen ohjelmointi on helppoa ja mahdollistaa nopeiden testien toteuttamisen. Unreal Enginen C++-kielen avustuksen nopeuttavat ohjelmointia, mutta ohjelmakoodin kääntämiseen voi kulua kehittäjältä yllättävän paljon aikaa. CryEnginessä logiikan ohjelmointi on visuaalisen ohjelmoinnin osalta helppoa ja yhtä suoraviivaista kuin Unreal Enginessä. CryEnginen dokumentaatio on kuitenkin huonompi kuin Unityssä tai Unreal Enginessä ja hankaloittaa etenkin lua- ja C++-kielellä tehtävää ohjelmointia. Lumberyardin visuaalinen ohjelmointi ja C++-ohjelmointi ovat käytännössä identtisiä CryEngineen verrattuna. Uusi komponenttipohjainen peliobjektijärjestelmä ja siihen liittyvä lua-skriptaus todennäköisesti helpottavat ohjelmointia tulevaisuudessa, mutta tällä hetkellä Lumberyardin ohjelmointidokumentaatio on huono. Stringrayn visuaalinen ohjelmointi on samalla tasolla CryEn-

ginen, Unreal Enginen ja Lumberyardin kanssa. Visuaalisen ohjelmoinnin periaatteet ovat samat ja dokumentaatio kohtalainen. Stingrayn lua-kielinen skriptaus on käyttäjälle suoraviivaista sisäänrakennetun editorin ja tulkattavan kielen takia. Stingrayssa ohjelmoinnin dokumentaatio on hyvä. Taulukkoon 2 on koottu vertailtavien pelimoottoreiden tukemat ohjelmointikielet ja tavat.

Taulukko 2. Pelimoottoreiden tukemat ohjelmointikielet.

	Visuaalinen ohjelmointi	Ohjelmointikielet
Unity	Ei	C#, UnityScript
Unreal Engine	Kyllä	C++
CryEngine	Kyllä	C#, C++, Lua
Lumberyard	Kyllä	C++, Lua
Stingray	Kyllä	C++, Lua

10 Valaistus, jälkikäsittely ja efektit

Tässä osiossa luodaan katsaus vertailtavien pelimoottoreiden valaistukseen, pelikuvan jälkikäsittelyyn ja efektien luonnin mahdollisuuksiin. Valaistuksesta, jälkikäsittelystä ja efekteistä arvioidaan niiden helppokäyttöisyyttä, monipuolisuutta ja dokumentaation kattavuutta.

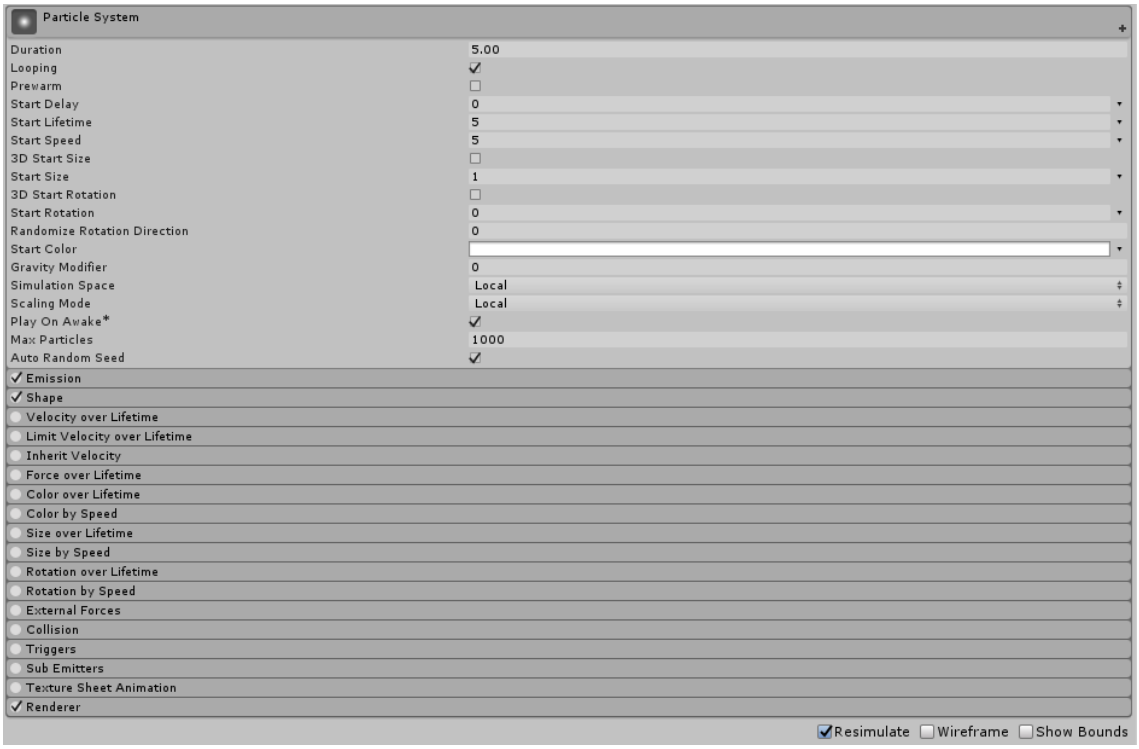
10.1 Unity

Unity sisältää kuusi erityyppistä valonlähdettä. Perusvalonlähteitä ovat kaikkiin suuntiin valoa tuottava pistevalo, keilana valoa tuottava kohdevalo ja tasaisesti yhteen suuntaan koko pelikenttään valoa tuottava suuntavalo. Näiden lisäksi va-

loa pelimaailmaan tuovat valoa hohtavat aluevalot, peliobjektien hohtavat materiaalit ja kentälle määritelty ympäröivä valaistus. Pistevalo, kohdevalo ja suuntavalo voivat olla joko reaaliaikaisia, leivottuja tai sekoitus näiden väliltä. Reaaliaikaiset valot eivät tuota ollenkaan epäsuoraa valaistusta. Leivotut valot tuottavat myös epäsuoran valaistuksen, mutta ne eivät tuota valaistusta tai varjoja liikkuviin objekteihin. Leivottujen valojen tuottama valaistus leivotaan valokarttaan, josta arvot voidaan suoraan lukea ilman niiden laskemista. Sekoitus näiden kahden välillä vaikuttaa myös liikkuviin objekteihin tuottaen kuitenkin epäsuoraa valaistusta. (Unity Documentation 2016h.)

Unityssä kuvan jälkikäsittely perustuu pelikuvan renderöivään kameraan liitettäviin komponentteihin. Komponentit käsittelevät kameran tuottamaa renderöintitekstuuria ja kuvaa käsitteleviä komponentteja voi olla kamerassa useita. Unity sisältää sisäänrakennettuna paljon valmiita kuvan jälkikäsittelykomponentteja. Käyttäjä voi myös luoda jälkikäsittelyefektejä itse. (Unity Documentation 2016i.)

Muita visuaalisia efektejä Unityssä voi lisätä pelimaailmaan ja peliobjekteihin komponentteina. Visuaalisina komponentteina voi lisätä partikkeliefektejä, linjan renderöijiä, linssiheijastuksia, valokehiä ja kuvan heijastavia projektoreja. (Unity Documentation 2016j.) Unityn partikkeliefektien muokkaus on moduulipohjaista (Unity Documentation 2016k). Kuvassa 18 näkyy kaikki Unityn partikkelien muokkauksen moduulit. Moduulien muokkaus ja päälle kytkentä on yksinkertaista.



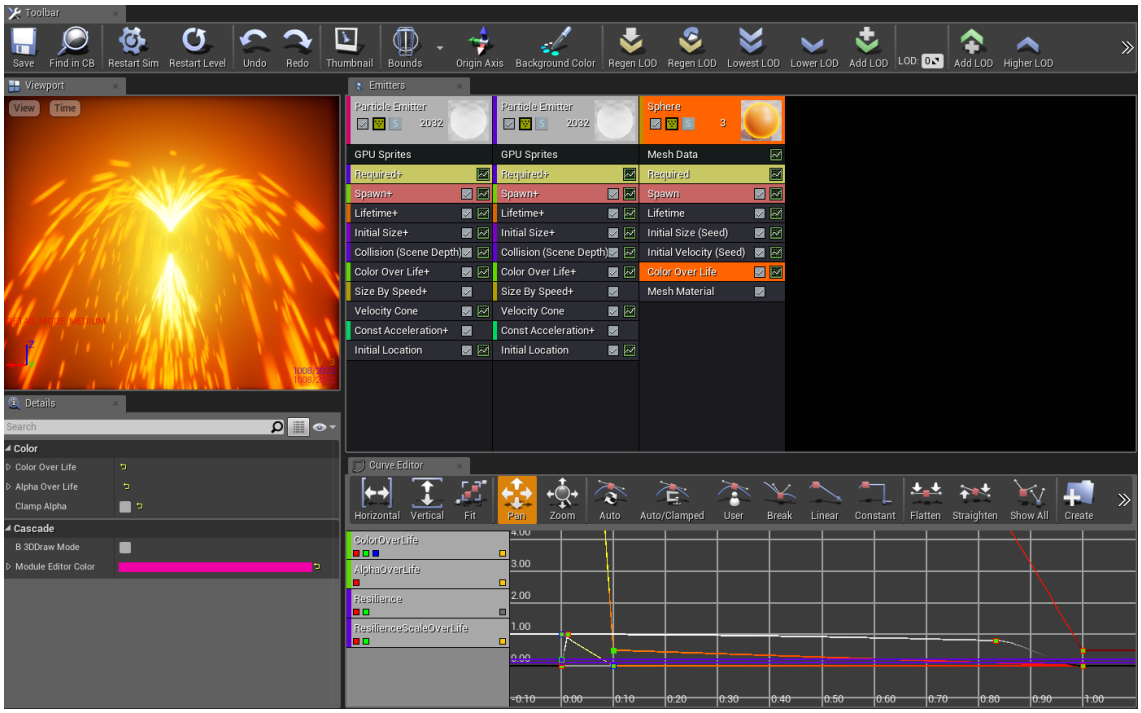
Kuva 18. Partikkeliefektien muokkaus Unityssä.

Unityn valaistuksen, jälkikäsittelyn ja efektien dokumentaatio on todella hyvä. Valaistuksen dokumentaatio käy läpi kattavasti erilaisten valotyyppien toimintaperiaatteet ja globaalin epäsuoran valaistuksen. Dokumentaatio sisältää myös hyvää tietoa eri valaisutyylien tehovaatimuksista. Valojen käyttäminen Unityssä on suoraviivaista. Helpoimmillaan käyttäjän tarvitsee vain raahata halutunlainen valo valikosta oikealle paikalle pelimaailmassa ja valita onko valo reaaliaikainen vai leivottu. Leivottuja valoja käyttäessä on huomattavaa, että valaistuksen leipominen hyvällä tarkkuudella ja isoissa kentissä voi viedä todella paljon aikaa. Dokumentaatio käy läpi kaikkien efektien ominaisuudet ja sisältää paljon esimerkkikuvia efektien vaikutuksista. Käyttäjälle valmiita efektejä on tarjolla runsaasti ja niiden käyttäminen on kehittäjän kannalta yksinkertaista. Valmiin efektin lisääminen komponttina on yksinkertaisimmillaan tehty muutamalla hiiren klikkauksella. Unityn partikkelijärjestelmällä luotujen partikkeliefektien muokkaus on moduulipohjaisuuden ansiosta helppoa. Partikkelien ominaisuudet on jaoteltu loogisesti osioihin, ja osioita voi tarvittaessa poistaa kokonaan käytöstä tai ottaa käyttöön.

10.2 Unreal Engine

Unreal Engine sisältää 4 erilaista valotyyppiä perusvalaistuksen toteuttamiseen. Suuntavalo, pistevalo ja kohdevalo ovat perustoiminnoiltaan samanlaisia kuin Unityssä. Neljäntenä tyyppinä Unreal Engine sisältää taivasvalon. Taivasvalolla pelimaailmaan voidaan luoda pelimaailman mukaisesti säätyvä valo. Kaikille neljälle valotyypille Unreal Enginessä voi asettaa liikkuvuuden kolmesta eri vaihtoehdosta. Staattiset valot ovat kokonaan leivottuja ja niiden tieto leivotaan Unreal Enginessä valomassaksi kutsuttuun valokarttaan. Paikallaan olevaksi merkityn valon kirkkautta ja väriä voi säätää. Paikallaan olevan valon varjot ja valon tuottama epäsuora valaistus leivotaan kuitenkin valokarttaan. Liikkuvat valot ovat kokonaan dynaamisia ja niiden tiedoista ei leivota mitään valokarttaan. (Unreal Engine Documentation 2016i.) Staattista valaistusta voidaan Unreal Enginessä tuottaa myös hohtavilla materiaaleilla (Unreal Engine Documentation 2016j).

Unreal Enginessä jälkikäsittely toteutetaan pelimaailmaan sijoitettavilla PostProcessVolume-objekteilla tai liittämmällä PostProcess-komponentti pelikuvan tuottavan kameran peliobjektiin. PostProcessVolume sisältää 16 lopulliseen kuvaan vaikuttavaa muokattavaa ominaisuutta. Pelikenttä voi sisältää useita PostProcessVolumeja ja niitä voi sijaita myös päällekkäin. Päällekkäin olevien PostProcessVolumien arvoja interpoloidaan lineaarisesti käyttäjän asettamien asetuksien mukaisesti. Käyttäjä voi luoda omia kuvaefektejä joita voidaan liittää PostProcessVolumeihin. Näitä kuvaefektejä kutsutaan Unreal Enginessä Post Process Materiaaleiksi ja niiden luonti tapahtuu node-pohjaisesti samalla periaatteella kuin peliobjektien materiaalien luonti. (Unreal Engine Documentation 2016k.) Partikkeliefektejä Unreal Enginessä luodaan Cascade nimisellä järjestelmällä. Partikkeliefektien luominen on modulaarista. Modulaarisuus tarkoittaa tässä tapauksessa sitä, että partikkeliefektiä luotaessa se sisältää vain muutaman välttämättömimmän toiminnon ja käyttäjän on itse lisättävä moduuleita tuottaakseen haluamansa efektin. (Unreal Engine Documentation 2016l.) Kuvassa 19 näkyy Unreal Enginen partikkeliefektien muokkaustyökalu. Työkalussa on näkyvissä partikkeliefekti, joka sisältää kolme partikkelilähdettä joiden sisällä on erinäisiä toimintaan vaikuttavia moduuleja.



Kuva 19. Partikkeliefektien muokkaus Unreal Enginessä.

Unreal Enginen valaistuksen, jälkikäsittelyn ja efektien dokumentaatio on todella hyvä. Dokumentaatio kattaa kaikki perusteet ja sisältää myös syvällistä tietoa. Unreal Engine sisältää myös paljon tutoriaaleja, jotka opastavat eri työkalujen käytössä. Käyttäjälle on tarjolla myös ilmaisia esimerkkiprojekteja, jotka sisältävät paljon esimerkkejä pelimoottorin eri osa-alueista. Valaistuksen ja valojen käyttäminen on Unreal Enginessä yhtä suoraviivaista kuin Unityssä. Staattisten ja paikallaan olevien valojen kanssa Unreal Enginessä on kuitenkin sama ongelma kuin Unityssä; paljon tällaisia valoja käyttäessä valaistustietojen leipominen valokarttaan voi viedä todella paljon aikaa. Unreal Enginessä partikkeliefektien luominen on monipuolista. Partikkeleiden ominaisuuksiin vaikuttavia moduuleja on paljon ja niiden arvojen säätäminen on suoraviivaista.

10.3 CryEngine

CryEnginessä pääasiallisina valonlähteinä toimivat aurinkoa simuloiva päivänaikajärjestelmä ja valoentiteetti. (CryEngine Documentation 2013b). CryEnginen valoentiteetin voi asettaa toimimaan eri tyyppisinä valoina sen arvoja säätämällä. Valoentiteettiä voi käyttää pistevalona, kohdevalona ja aluevalona. Valoentiteetti

on aina dynaaminen valo. (CryEngine Documentation 2015c.) CryEnginen valaistus on dynaamista ja vokselipohjaista (CryEngine Documentation 2016g). Hyvän valaistuksen saavuttamiseksi käyttäjän on kuitenkin asetettava kenttään Environment Probe -objekteja (CryEngine Documentation 2014e).

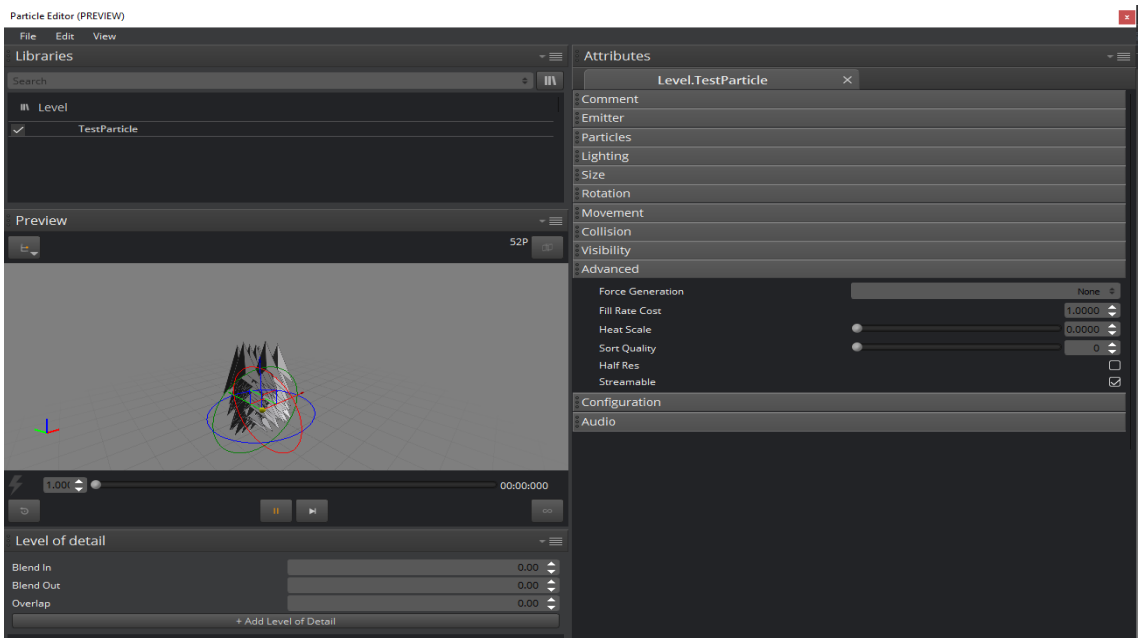
CryEngine sisältää Flow Graph -ohjelmoinnin kautta hallittavia visuaalisia efektejä ja jälkikäsittelysäätöjä, koko kuvaan vaikuttavia värisäätöjä, muokattavia linssiheijastuksia ja vesiefektejä (CryEngine Documentation 2013c). CryEngine sisältää myös työkaluja sääolosuhteiden simuloimiseen (CryEngine Documentation 2015d). Reunojen pehmennystä varten CryEnginessä käyttäjä joutuu käyttämään konsolikomentoja (CryEngine Documentation 2015e). CryEnginen partikkelieditorilla luotuja partikkeleita kutsutaan partikkelientiteeteiksi. Partikkelientiteettejä voi liittää peliobjekteihin ja niitä voi hallita Flow Graph -ohjelmoinnilla. CryEnginen partikkelit ovat komponenttipohjaisia. (CryEngine Documentation 2014f.)

CryEnginen valaistuksen, jälkikäsittelyn ja efektien dokumentaatio on vaihteleva. Valaistus käsitellään kohtuullisesti, mutta valaistuksen käsittely tuntuu pintapuoliselta. Lisäksi valaistuksen dokumentaatio on hajanaista ja se on jakautunut moneen eri paikkaan dokumentaatiossa. Jälkikäsittelyn dokumentaatio on huono sisältäen todella pintapuolisen läpikäynnin jälkikäsittelyn mahdollisuuksiin ja vähän esimerkkejä. Lisäksi dokumentaatiota ei ole päivitettyä moneen vuoteen. Osa jälkikäsittelystä täytyy myös tehdä konsolikomentojen kautta, mikä voi olla uudelle kehittäjälle vaivalloisen tuntuista. Myös partikkeliefektien dokumentaatio on suurilta osin vanhentunutta ja kaikki ohjeet eivät päde uuden CryEnginen version partikkelieditoriin. CryEnginen suurin etu Unityyn ja Unreal Engineen verrattuna on täysin dynaaminen globaali valaistus, joka ei vaadi valaistustietojen leipomista valokarttaan. Tämä säästää kehittäjältä todella paljon aikaa ja valaistuksen asettelu on helpompaa, kun lopullisen valaistuksen näkee jo valoja asettaessa. CryEnginen valoentiteetin säätöjen saaminen kohdalleen voi olla kuitenkin aluksi hankalaa, koska säätöjä on niin paljon.

10.4 Lumberyard

Valaistuksen käyttäminen Lumberyardissa on perusteiltaan hyvin samanlaista kuin CryEnginessä. Käyttäjän käytettävissä on päivänaikajärjestelmä, valoentiteetti ja Environment Probe -objekti (Lumberyard Documentation 2016m). Lumberyardissa valoja voi kuitenkin luoda komponenttisysteemin kautta ja tämä järjestelmä tulee tulevaisuudessa syrjäyttämään kokonaan aiemman tavan luoda valoja (Lumberyard Documentation 2016n).

Jälkikäsittelyefektejä Lumberyardissa hallitaan XML-tiedostojen ja Flow Graph – tai lua-ohjelmoinnin kautta. XML-tiedostoissa voi asettaa halutut arvot kaikille efekteille. XML-tiedoston mukaiset efektit voi aktivoida ja poistaa käytöstä ohjelmoinnilla. Eri efektiasetuksia voi myös sekoittaa toisiinsa ja efektien tehokkuutta voi myös säätää kameran etäisyyden mukaisesti. (Lumberyard Documentation 2016o.) Lumberyard sisältää uuden kehitysvaiheessa olevan partikkelieditorin. Partikkelia muokattaessa käyttäjän näkyvillä on kaikki partikkelin muokattavat ominaisuudet. Uuden partikkelin luonnin jälkeen kuitenkin vain osa ominaisuuksista on käytössä, ja käyttäjä voi ottaa käyttöön ominaisuudet jotka partikkeliefekti tarvitsee. (Lumberyard Documentation 2016p.) Kuvassa 20 näkyy Lumberyardin partikkelieditori. Oikealla puolella kuvaa näkyvät kaikki partikkeliefektin muokattavat osa-alueet.



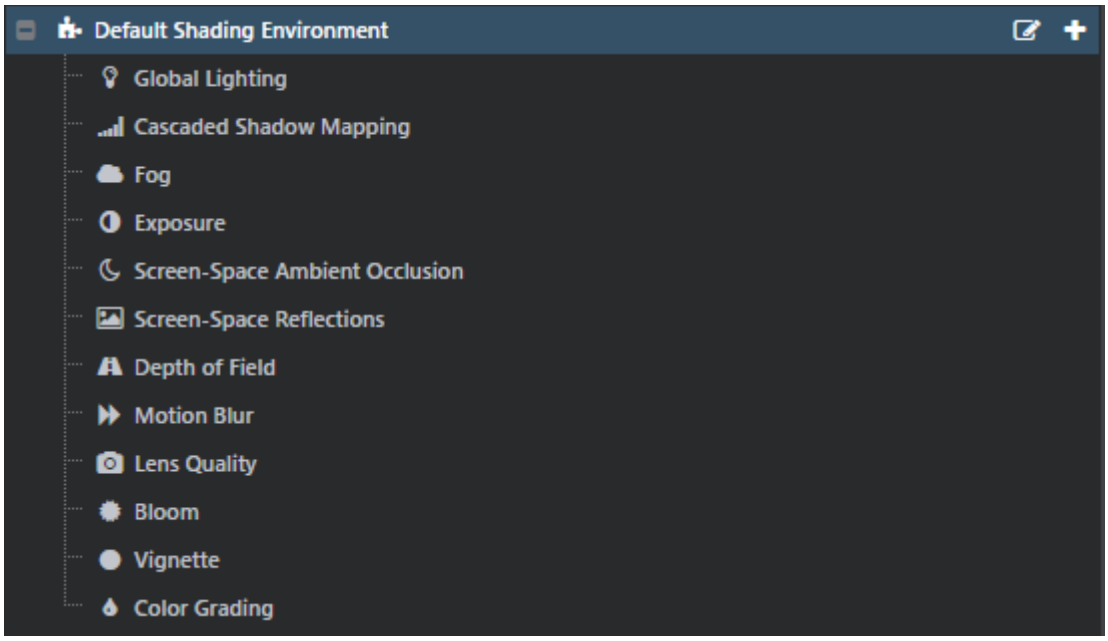
Kuva 20. Lumberyardin partikkelieditori.

Lumberyardin dokumentaatio kattaa valaistuksen, jälkikäsittelyn ja efektit kohtalaisesti. Valaistuksen dokumentaatio käy läpi perusteet, mutta sisältää vain vähän esimerkkejä. Jälkikäsittelyn kaikkia osa-alueita ei käydä dokumentaatiossa läpi. Vaihtoehdot ovat kuitenkin nähtävillä moottorin oletusefektit sisältävässä XML-tiedostossa. Uuden partikkelieditorin dokumentaatio on hyvä. Partikkelieditorin eri osa-alueet käydään hyvin läpi ja dokumentaatio sisältää hyviä ohjeita partikkelien tehokkaaseen käyttöön. Lumberyadin pohjautuminen CryEngineen on valaistuksen kannalta hyvä asia. Reaaliaikaisen valaistuksen käyttö nopeuttaa kehitystyötä myös Lumberyardissa, ja valojen käyttäminen komponenttipohjaisena on asetuksien säädön kannalta helpompaa kuin CryEnginessä.

10.5 Stingray

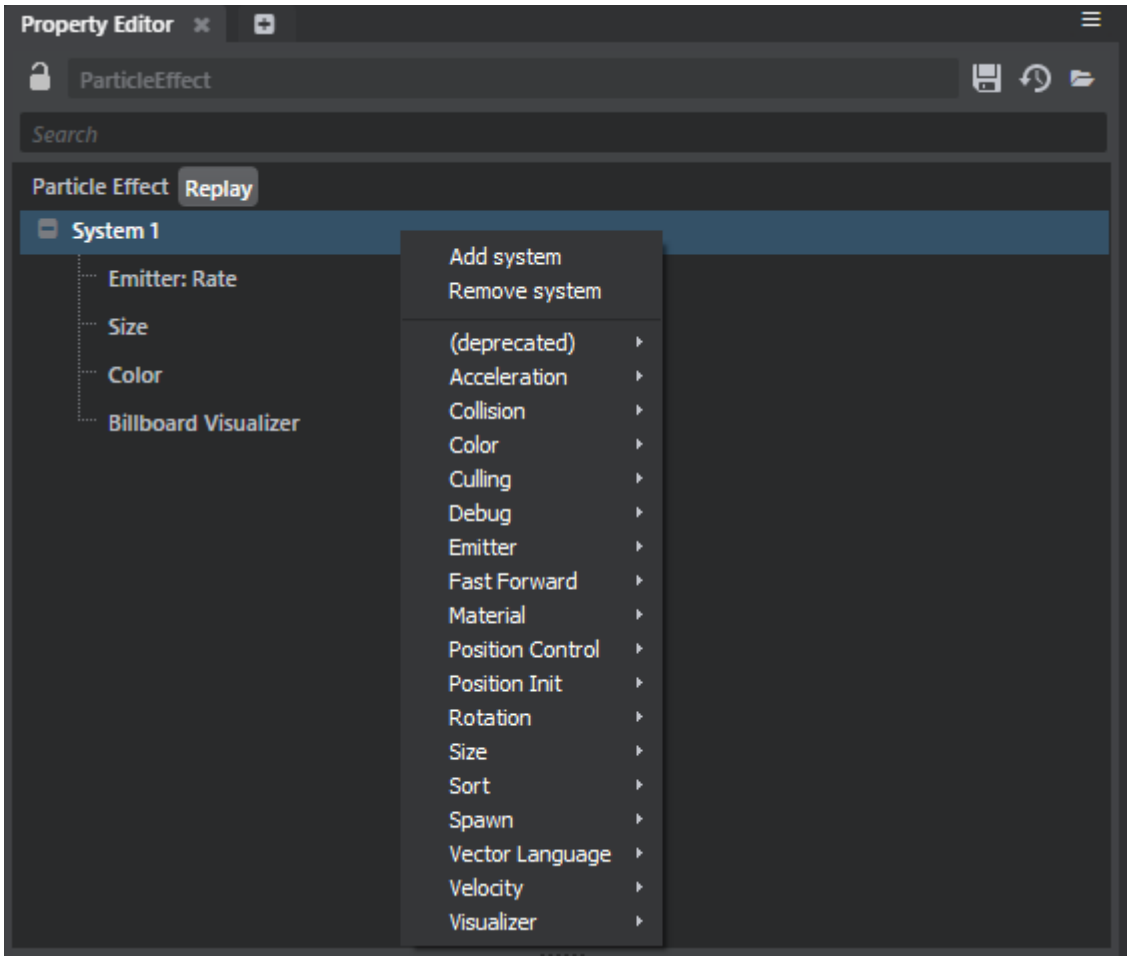
Stingrayssa valonlähteinä voi käyttää suuntavaloa, kaikkiin suuntiin hohtavaa valoa, kohdevaloa ja laatikkovaloa. Suuntavalo, kaikkiin suuntiin hohtava valo ja kohdevalo ovat perusteiltaan Stingrayssa samanlaisia kuin Unityssä ja Unreal Enginessä. Laatikkovalo tuottaa valoa laatikon yhdeltä sivulta ja sillä voi simuloida esimerkiksi ikkunan läpi tulevaa valoa. Käyttäjä voi valita onko valo dynaaminen vai leivotaanko kaikki sen tuottama valaistustieto valokarttaan. Hohtavien materiaalien tuottama valo on aina valokarttaan leivottua. Valojen leipominen Stingrayssa on helppoa, mutta se voi viedä paljon aikaa. (Stingray Help 2016k.)

Jälkikäsittelyefektit asetetaan ja säädetään Stingrayssa Shading Environment -entiteetin kautta. Näitä entiteettejä voi olla pelikentässä vain yksi, mutta entiteetin arvoja voi muuttaa reaaliajassa ohjelmakoodilla. (Stingray Help 2016l.) Kuvassa 21 on nähtävillä kaikki Stingrayssa säädettävät jälkikäsittelyefektit.



Kuva 21. Stingrayn mahdolliset jälkikäsittelyefektit.

Partikkelien luominen ja muokkaus on Stingrayssa komponenttipohjaista. Partikkeliefektiä luotaessa se sisältää vain välttämättömimmät toiminnot ja käyttäjä voi itse lisätä tarvitsemansa komponentit efektiin. Partikkeliefektiin voi myös lisätä erillisiä partikkelisysteemejä. (Stingray Help 2016m.) Kuvassa 22 näkyy kaikki partikkeliefekteissä käytettävissä olevat komponentit.



Kuva 22. Kaikki partikkeliefektin komponentit Stingrayssa.

Stingrayn valaistuksen, jälkikäsittelyn ja efektien dokumentaatio on hyvä. Valaistuksesta käsitellään perusasiat, mutta kokonaisuutena valaistuksen käsittely on hieman pintapuolinen. Dokumentaatio ei sisällä paljoa esimerkkejä valojen käytöstä. Jälkikäsittelyn dokumentaation tilanne on samanlainen. Dokumentaatio käsittelee perusteet ja selittää eri jälkikäsittelyvaihtoehtojen vaikutuksen. Vaikutuksien kuvaukset ovat kuitenkin erittäin lyhyitä eikä dokumentaatio sisällä esimerkkejä vaikutuksista. Myös partikkeliefektit katetaan samalla tavalla. Dokumentaatio käsittelee perusasiat, mutta ei sisällä vinkkejä tai esimerkkejä. Käytettävyydeltään valaistuksen luonti on Stingrayssa samalla tasolla kuin Unityssä ja Unreal Enginessä. Valojen lisääminen kentälle ja niiden arvojen säätäminen on yksinkertaista ja nopeaa. Staattisen ja epäsuoran valaistuksen leipominen valokarttaan voi kuitenkin viedä runsaasti kehitysaikaa.

10.6 Yhteenveto

Valaistuksen käyttämisen osalta pelimoottorit voi jakaa kahteen ryhmään. Unityssä, Unreal Enginessä ja Stingrayssa valotyypit ja valojen asettelu kenttään on hyvin samanlaista. Nämä kolme pelimoottoria mahdollistavat myös staattisen ja epäsuoran valaistuksen leipomisen valokarttoihin. On kuitenkin huomattavaa, että valojen leipominen hyvällä tarkkuudella vaatii todella paljon aikaa. CryEngine ja Lumberyard sisältävät puolestaan täysin reaaliaikaisen valaistuksen, joka ei vaadi leipomista. Tämä helpottaa etenkin artistien työtä, sillä kaikki muutokset valaistukseen ovat nähtävillä välittömästi lopullisessa muodossaan. CryEnginen ja Lumberyardin valoentiteetti sisältää todella paljon muokattavia ominaisuuksia, joka saattaa hidastaa valojen asettelua ja säätöä ainakin aluksi. Unityssä, Unreal Enginessä ja Stingrayssa valojen säätöjä on vähemmän, mutta valotyyppejä enemmän.

Jälkikäsittelyn toteutukseltaan pelimoottorit tarjoavat hyvin erilaisia ratkaisuja. Unityssä jälkikäsittely toteutetaan kameraan liitettävillä yksittäisillä komponenteilla, Unreal Enginessä käyttäjä asettaa pelikentälle alueita joiden asetukset määräävät jälkikäsittelyn ominaisuudet, CryEnginessä jälkikäsittelyefektejä hallitaan visuaalisen ohjelmoinnin kautta, Lumberyardissa asetuksia hallitaan XML-tiedostolla, ja Stingrayssa on kenttäkohtainen säätö asetuksille. Monipuolisimmat asetukset löytyvät Unitystä ja Unreal Enginestä. Molemmat sisältävät paljon valmiita ominaisuuksia ja molempiin moottoreihin voi jälkikäsittelyefektejä tehdä myös itse. Stingrayssa asetuksien säätö on tehty helpoksi, mutta vain yksien asetuksien teko kenttää kohden hankaloittaa asetusten muuttamista kentän aikana. Lumberyard sisältää monipuoliset säädöt, mutta säätöjen käyttäminen ja muuttaminen on hieman vaivalloista ulkoisen XML-tiedoston lukemisen ja siihen kirjoittamisen takia. CryEnginen tilanne jälkikäsittelyn suhteen on huonoin pääosin huonon dokumentaation takia. CryEnginen dokumentaatio käy jälkikäsittelyä läpi todella vähän ja dokumentaatio tuntuu vanhentuneelta.

Pelimoottoreiden partikkelijärjestelmät ovat hyvin samankaltaisia. Kaikkien moottoreiden järjestelmät sisältävät pitkälti samat toiminnallisuudet vaikkakin järjes-

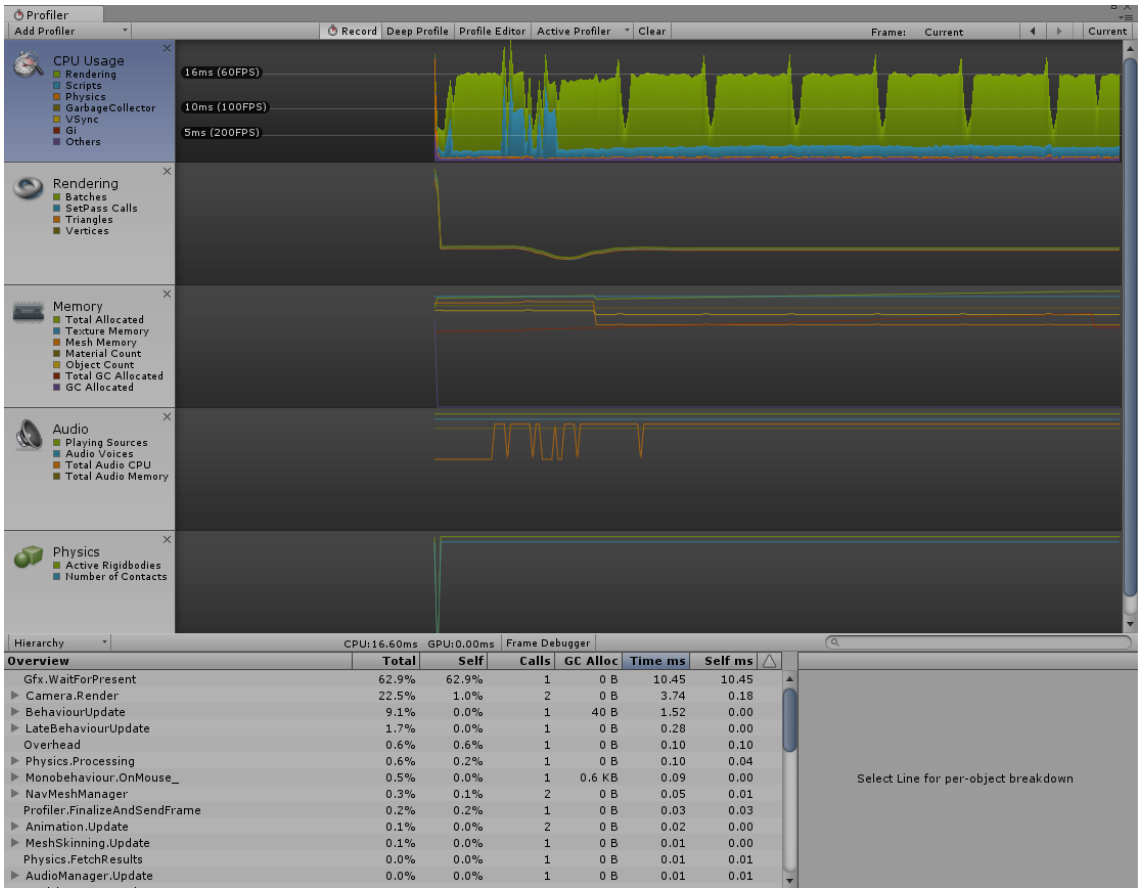
telmien käyttäminen voi hieman erota toisista. Partikkelijärjestelmien dokumentaatio on pelimoottoreissa hyvä lukuun ottamatta CryEngineä, jonka dokumentaatio tuntuu vanhentuneelta.

11 Profilointi ja kehittäjänäkymät

Pelin kehityskaaren aikana on tärkeää jatkuvasti tarkkailla pelin suorituskykyä ja mahdollisia ongelmakohtia suorituskyvyssä. Suorituskykyä voidaan pelimoottoreissa tarkkailla siihen tarkoitetuilla sisäänrakennetuilla työkaluilla. Tästä suorituskyvyn tarkkailusta käytetään nimitystä profilointi. Pelimoottorit sisältävät myös pelimaailmasta erilaisia visuaalisia näkymiä, jotka voivat helpottaa havainnollistamaan pelimaailman monimutkaisuutta. Vertailtavien pelimoottoreiden kesken profiloinnista ja kehittäjänäkymistä arvioidaan niiden monipuolisuutta, helppokäyttöisyyttä ja dokumentaation kattavuutta.

11.1 Unity

Unityn profilointityökalun saa avattua pelimoottorin sisällä omana ikkunanaan. Profilointi-ikkunasta käyttäjä näkee tarkkaa tietoa kehitettävän pelin suorituskyvystä ja tiedot ovat reaaliaikaisia. Profilontityökalu kerää tietoa suorittimesta, graafisesta suorittimesta, grafiikan piirtämisestä, muistista, äänistä, fysiikoista ja tietoverkoista. Profilointia voi myös suorittaa sovellukselle, jota ajetaan toisessa laitteessa kuten puhelimessa. Käyttäjä voi halutessaan profiloida kirjoittamaansa koodia funktiokohtaisesti mahdollistaen erittäin tarkan suorituskyvyn seurannan. (Unity Documentation 2016l.) Kuvassa 23 on nähtävillä Unityn profilointityökalun ulkoasu.



Kuva 23. Unityn Profiler työkalun ikkunan ulkoasu.

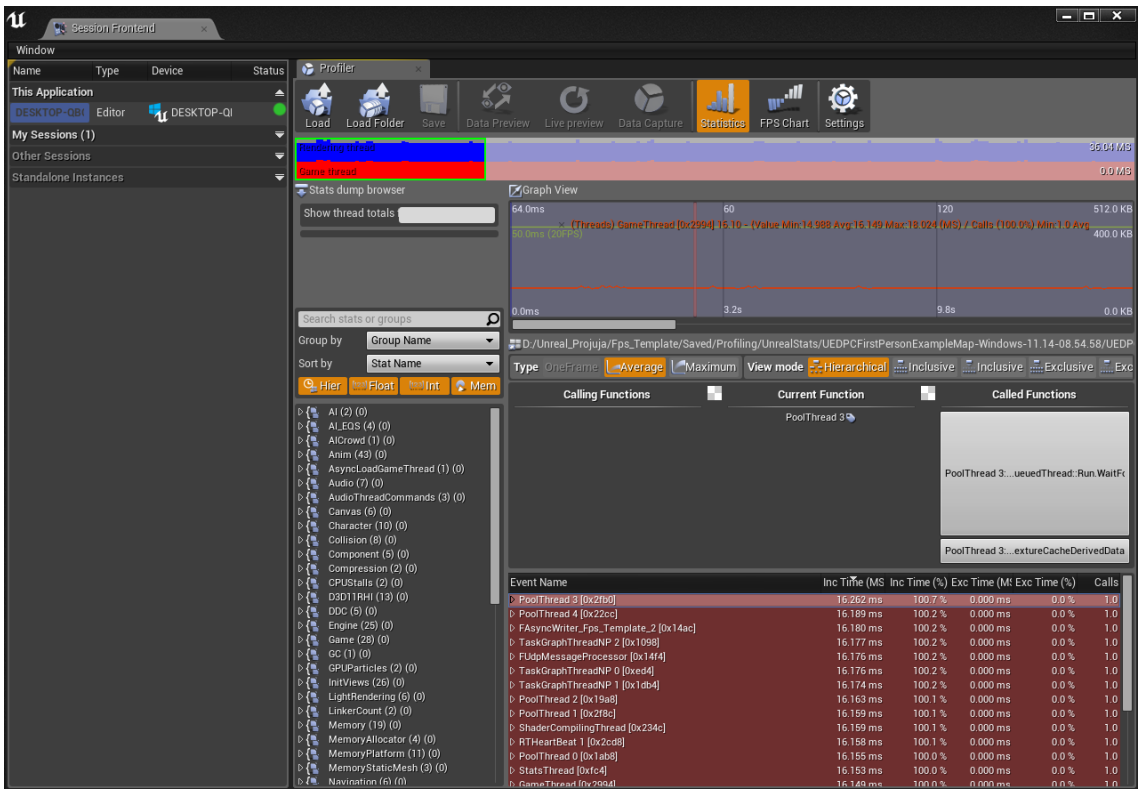
Unityssä näköportin piirtotyylin voi vaihtaa joidenkin asioiden havainnollistamisen helpottamiseksi. Piirtotyyleiksi voi valita muun muassa lopullista ulkoasua vastaavan varjostetun tyylin, pelkästään kehikkomaisen rakenteen näyttävän tyylin, ideaalisen tekstuurien koon värillisesti näyttävän tyylin tai pelkästään objektien värin ilman valaistusta näyttävän tyylin. (Unity Documentation 2016m.)

Unityn dokumentaatio kattaa profiloinnin ja eri tyyppiset kehittäjänäkymät hyvin. Profilointityökalun käyttäminen ja sen käyttämisen hyödyt selostetaan dokumentaatiossa kattavasti ja kaikki työkalun ominaisuudet käydään läpi. Profilointityökalun peruskäyttö on myös käyttäjän kannalta erittäin helppoa. Työkalun aukaiseminen tapahtuu valikosta kahdella hiiren klikkauksella, jonka jälkeen työkalu on jo käytettävissä ja sen voi antaa olla auki omana ikkunanaan tai liitettynä johonkin moottorin toiseen ikkunaan. Profilointityökalua ei tarvitse käynnistää erikseen, vaan se kerää tietoja aina, kunhan se vain on valikosta aukaistu ja nauhoitus on asetettu päälle. Kehittäjänäkymistä dokumentaatio käy perusteet läpi,

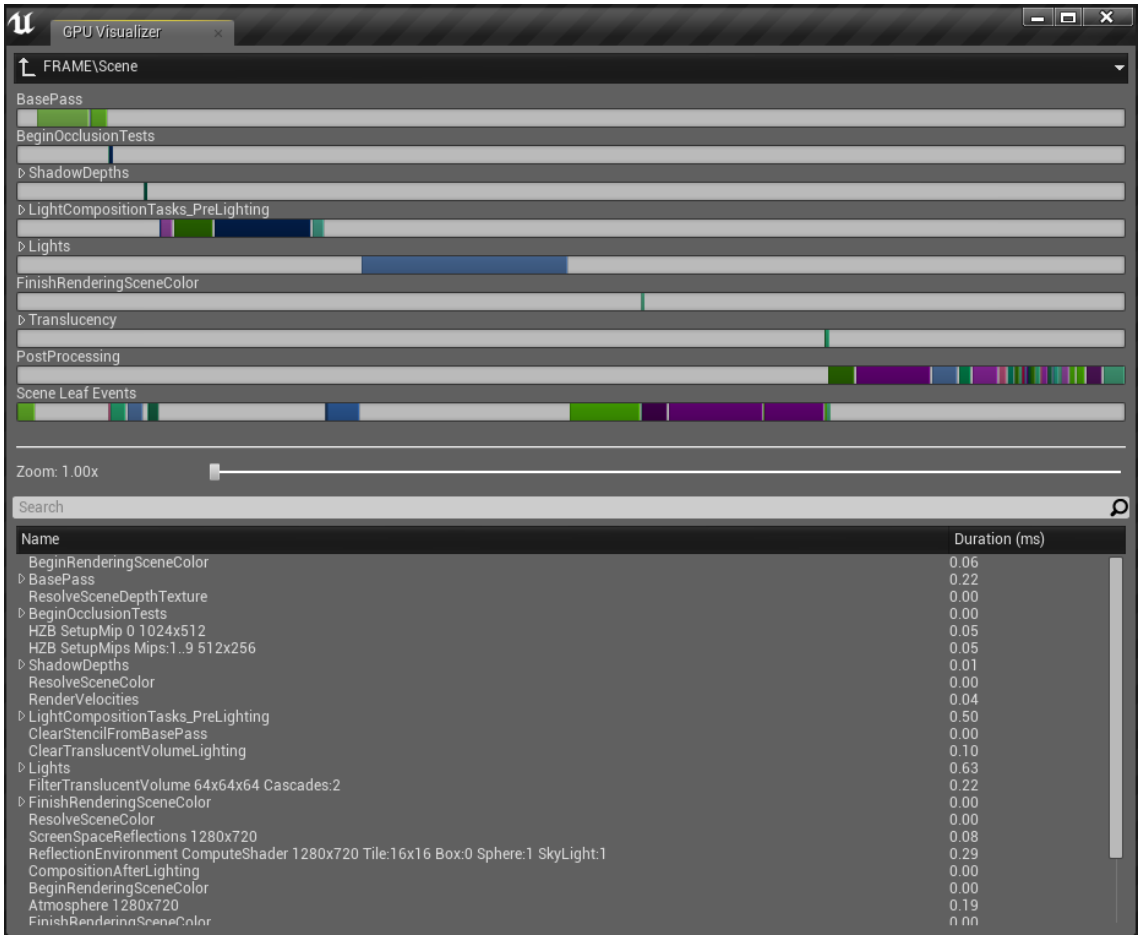
mutta se ei sisällä esimerkkikuvia jotka havainnollistaisivat eri vaihtoehtojen hyödyllisyyttä käyttäjälle.

11.2 Unreal Engine

Unreal Enginessä profilointityökalujen käyttö tapahtuu pääasiassa kehitettävän pelin suorituksen aikana konsolikomennoilla. Konsolikomentoja käyttämällä käyttäjä saa näkyviin tietoa pelin eri osa-alueiden suorituskyvystä ja muista tiedoista. Näytönohjaimen profilointikomento aukaisee näytönohjaimen suorituskykytiedot omana ikkunanaan pelin päälle. Näytönohjaimen tiedot tallennetaan vain yhdeltä piirrettävältä ruudulta ja ikkunan tiedot eivät päivity reaaliaikaisesti. Prosessorin tarkka profilointi tapahtuu liittämällä moottorin erillinen profilointityökalu suoritettavaan peliin jolloin tietoja voidaan kerätä ja lukea reaaliajassa. Vaihtoehtoinen tapa on tallentaa suoritettavasta pelistä tiedostoon lyhyt osio pelistä, joka voidaan myöhemmin profilointityökalulla avata ja pelin suorituskykyä tarkkailla sitä kautta. (Unreal Engine Documentation 2016m.) Unreal Engine mahdollistaa suorituskyvyn profiloinnin erittäin monipuolisesti (Kuva 24; Kuva 25).



Kuva 24. Unreal Enginen profilointityökalu.



Kuva 25. Näytönohjaimen profilointinäkymä Unreal Enginessä.

Unreal Engine sisältää 14 erilaista näkymätilaa, jotka voivat helpottaa hahmottamaan pelimaailman monimutkaisuutta. Unreal Engine sisältää näkymätiloja muun muassa valaistuksen visualisoinnille, valaistuksen monimutkaisuudelle ja varjostimien monimutkaisuudelle. (Unreal Engine Documentation 2016n.) Näkymätiloja voi vaihtaa editoria käyttäessä editorin valikosta tai vaihtoehtoisesti peliä suorittaessa konsolikomennoilla (Unreal Engine Documentation 2016n; Unreal Engine Documentation 2016m).

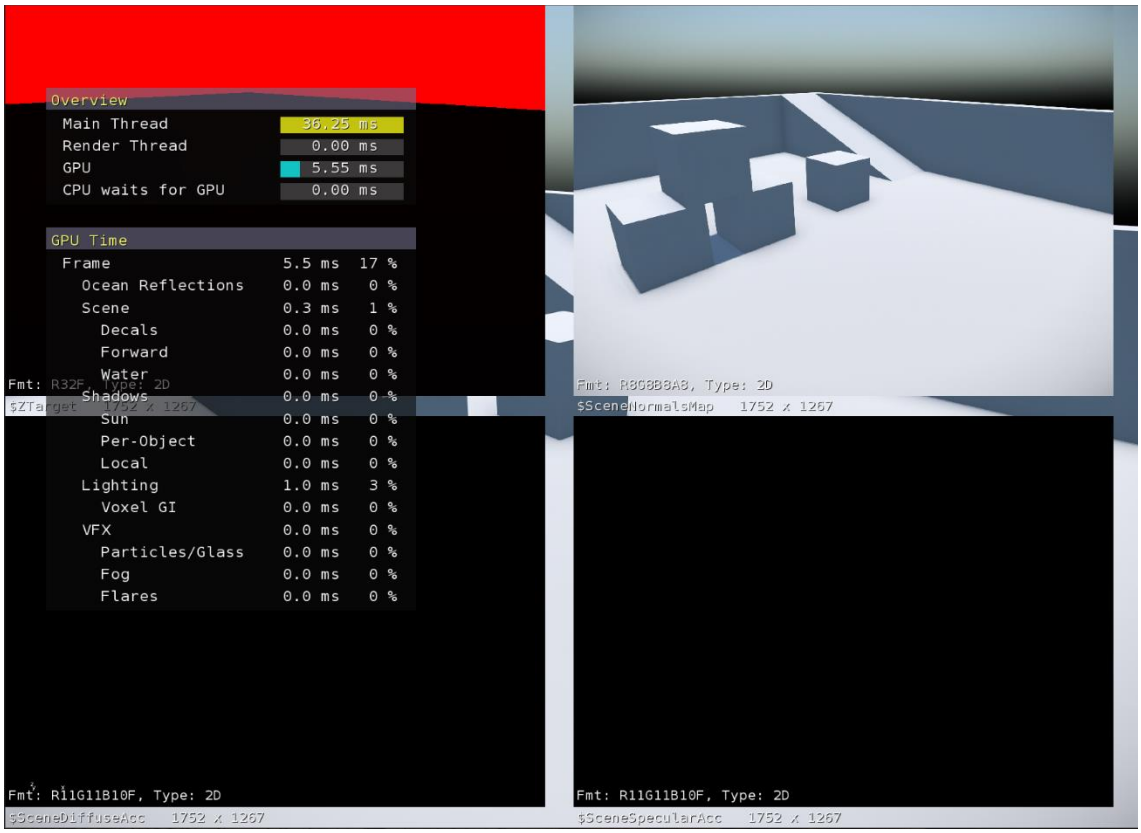
Unreal Enginen profilointia ja kehittäjänäkymiä käsittelevä dokumentaatio on erinomainen. Dokumentaatio sisältää kattavat ohjeet siitä, kuinka moottorin profilointityökaluja käytetään ja mitä niillä voidaan saavuttaa. Dokumentaatio sisältää myös erinomaista tietoa optimoinnin periaatteista ja optimointikohteista. Konsolikomentojen käyttäminen profilointia varten voi olla uudelle kehittäjälle aluksi

hankalaa. Konsolikomentojen käyttö mahdollistaa kuitenkin reaaliaikaisen testauksen erilaisilla grafiikka-asetuksilla. Kehittäjänäkymien dokumentaatio on hyvä ja dokumentaatio selittää jokaisen näkymän kohdalta sen tarkoituksen.

11.3 CryEngine

CryEnginessä profilointi ja sen hallinta tapahtuu kokonaisuutena konsolikomentojen kautta. Konsolikomennoilla käyttäjä voi valita pelikuvan päällä reaaliaikaisesti näytettäväksi eri tyyppisiä tietoja kehitettävästä pelistä. Konsolikomentoja jotka liittyvät profilointiin CryEngine sisältää todella paljon. Komentojen avulla voi tarkkailla muun muassa muistinkäyttöä, prosessorin kuormitusta ja näytönohjaimen kuormitusta. CryEngine ei sisällä erillistä profilointityökalua, vaan kaikki profilointi tapahtuu konsolikomennoilla. (CryEngine Documentation 2015f.)

Näkymätilojen hallinta tapahtuu myös CryEnginessä konsolikomentojen kautta. Konsolikomentoja käyttämällä näköportin saa näyttämään käyttäjälle visuaalisesti muun muassa valaistustietoja, partikkeliefektien tietoja ja peliobjektien väri- ja heijastustietoja. Näkymätilojen vaihto ja hallinta onnistuvat ainoastaan konsolikomentojen kautta. (CryEngine Documentation 2015f.) Kuvassa 26 on nähtävillä CryEnginen näköportti, kun konsolikomentojen kautta on laitettu päälle renderöintiprofilointi ja jälkivarjostetun renderöinnin näkymä.



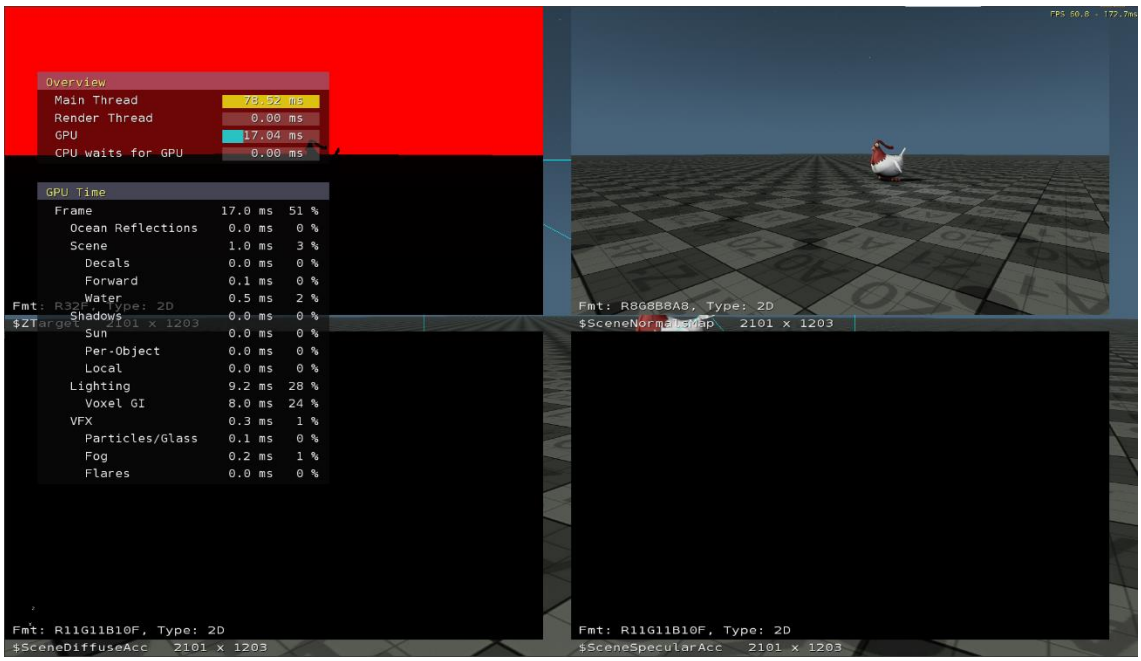
Kuva 26. Näkymä yhdestä CryEnginen profilointivalinnasta ja näkymätilasta.

CryEnginen profilointia ja kehittäjänäkymiä käsittelevä dokumentaatio on hyvä. Dokumentaatio sisältää hyvät perustiedot profiloinnista ja se käsittelee kaikki profilointivaihtoehdot hyvin sisältäen selityksen jokaisesta konsolikomennosta. Myös erilaiset näkymätilat käsitellään samalla tarkkuudella. Profiloinnin ja kehittäjänäkymien konsolikomentopohjaisuus voi tuntua uudesta kehittäjästä aluksi vaivalloiselta. Muistettavia komentoja onkin runsaasti ja kaikkien komentojen oppiminen vaatii aktiivisellakin käytöllä jonkin verran aikaa.

11.4 Lumberyard

Lumberyard sisältää kehitysvaiheessa olevan profilointityökalun. Profilointityökalu avataan omana suoritettavana tiedostonaan ja avattu työkalu voidaan liittää suoritettavaan sovellukseen. Profilointityökalu käyttää apunaan GridHub-nimistä yhdistymisverkkoa. Profilointityökalua voi käyttää prosessorin, muistin ja verkkoyhteyksien profilointiin. Profilointityökalua voi käyttää reaaliaikaisesti tai sillä voi

tarkkailla tallennettua profilointidataa. (Lumberyard Documentation 2016q.) Lumberyardissa toimivat myös CryEnginessä käyettävät konsolikomennot profilointinille ja kehittäjänäkymille, mutta näiden konsolikomentojen käyttöön ei löydy ohjeita Lumberyardin dokumentaatiosta (Kuva 27).



Kuva 27. Näkymä yhdestä Lumberyardin profilointivalinnasta ja näkymätilasta.

Lumberyardin profilointia ja kehittäjänäkymiä käsittelevä dokumentaatio on hyvä. Dokumentaatio sisältää hyvät ohjeet työkalun peruskäyttöön ja dokumentaatio selittää mitä työkalulla pystyy tekemään. On kuitenkin outoa, että Lumberyardin dokumentaatio ei sisällä ohjeistusta konsolikomennoilla suoritettavaan profilointiin tai konsolikomennoilla hallittaviin kehittäjänäkymiin, vaikka ne tuovat todella paljon lisäapua kehitykseen. Lumberyardin omalla profilointityökalulla profilointi ei onnistu yhtä monipuolisesti kuin kaikilla konsolikomennoilla.

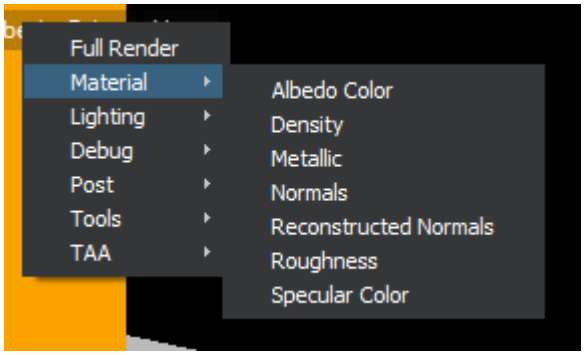
11.5 Stingray

Stingrayssa profilointitietojen näyttäminen ja hallinta on konsolikomentopohjaista (Stingray Help 2016n). Stingray sisältää konsolikomentoja valmiiden kokonaisuuksien näyttämiseksi tai käyttäjä voi itse luoda näytettäviä kokonaisuuksia joi-

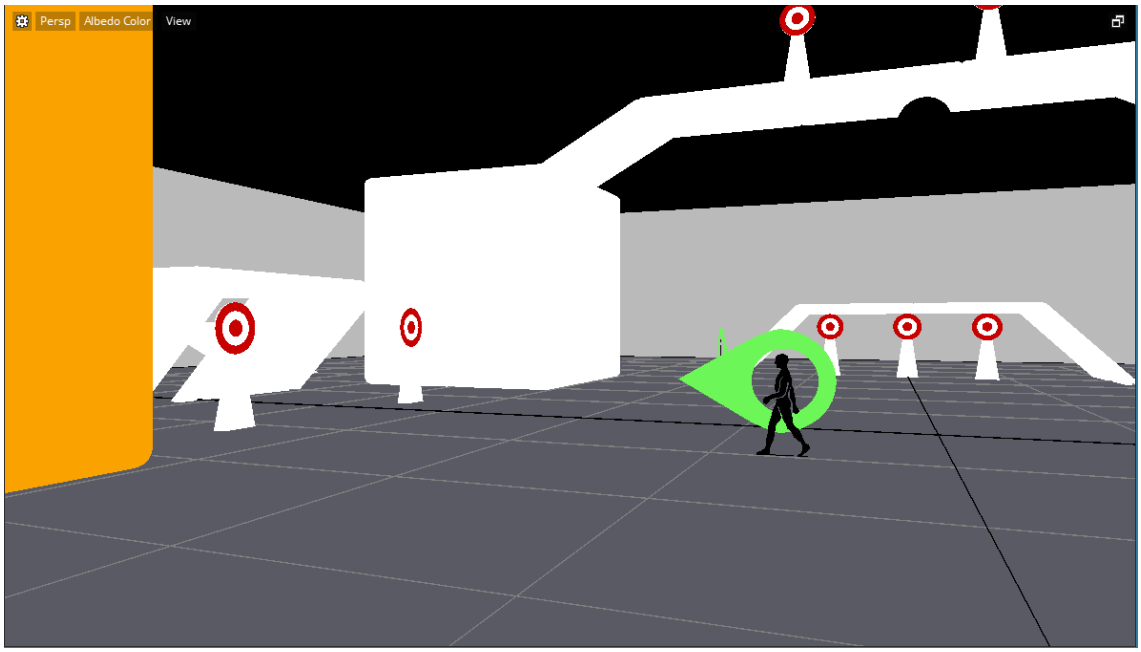
hin tieto otetaan moottorin seuraamista tiedoista tai käyttäjän seurattavaksi määritellyistä tiedoista (Stingray Help 2016n; Stingray Help 2016o). Kuvassa 28 on nähtävillä yksi Stingrayn valmiista profilointinäkymistä. Tästä näkymästä käyttäjälle selviää muistinkäyttö, renderöintiajat kohteittain, ruudunpäivitysnopeus ja prosessorin sekä näytönohjaimen kuormitus.

Kuva 28. Stingrayn yksi valmiista profilointinäkymistä.

Stingray sisältää kehittäjänäkymiä joiden välillä voi siirtyä editorin valikosta. Valittavina näkyminä on muun muassa erilaisia näkymiä väreille ja tekstuureille, valaistusnäkymiä ja jälkikäsittelynäkymiä. (Kuva 29.) Kuvassa 30 on nähtävissä näkymä, jossa näkyy vain kentässä olevien peliobjektien väritieto ilman valaistusta.



Kuva 29. Stingrayn kehittäjänäkymien valitseminen editorissa.



Kuva 30. Stingrayn kehittäjänäkymä jossa näkyy vain objektien väritieto.

Stingrayn profilointia ja kehittäjänäkymiä käsittelevä dokumentaatio on huono. Dokumentaatio sisältää vain hyvin pintapuolisen katsauksen profiloinnin mahdollisuuksiin ja se ei sisällä käyttöesimerkkejä jotka helpottaisivat profilointityökalujen käyttämisen aloituksessa. Dokumentaatio ei myöskään kunnolla selvennä mitä kaikkia tietoja käyttäjä voi profiloida ilman tietojen lisäämistä profiloitavaksi statistiikaksi. Stingrayn dokumentaatio ei sisällä omaa osiota erilaisten näkymien käytölle ja niiden käyttämisen hyödyille.

11.6 Yhteenveto

Profilointityökalujen käyttäminen vertailtavissa pelimoottoreissa on käytön monipuolisuuden ja helppouden kannalta hyvin vaihtelevaa. Käytettävyyden kannalta Unityn profilointityökalu on vertailtavista työkaluista paras. Unityn työkalu sisältää kaiken tarvittavan yhdessä ikkunassa ja käyttäjä ei tarvitse konsolikomentoja profiloinnin käyttämiseksi. Unreal Enginen työkalut ovat myös käytettävyydeltään hyvät, mutta eivät aivan yllä Unityn tasolle. Unreal Enginen profilointityökalujen käyttö vaatii muutaman konsolikomennon opettelun ja suorituskyvyn profilointi reaaliajassa on hieman hankalampaa kuin Unityssä. CryEngine on profiloinnin monipuolisuudessa Unityn ja Unreal Enginen tasolla, mutta käytettävyyden kan-

nalta CryEngine on hieman huonompi kuin Unreal Engine. Profilointi on CryEnginessä kokonaan konsolikomentopohjaista ja tästä syystä käyttäjän tulee opetella paljon erilaisia konsolikomentoja kaikkien ominaisuuksien käyttämiseksi. Lumberyard sisältää kaikki samat konsolikomentotoiminnallisuudet kuin CryEngine, mutta tämän lisäksi Lumberyard sisältää myös kehitysvaiheessa olevan profilointityökalun, joka on jo kehityksen tässä vaiheessa monipuolinen. Työkalun liittäminen kehitettävään peliin on kuitenkin vielä tässä vaiheessa hieman monimutkaisempaa kuin esimerkiksi Unityn työkalun käyttäminen. Stingrayn profilointityökalujen käyttäminen on vertailtavista moottoreista hankalinta. Stingrayn työkalut ovat monipuoliset ja muokattavat, mutta konsolikomentojen käyttäminen työkalujen käyttämiseksi on hitaampaa kuin muissa moottoreissa ja komennot ovat hieman monimutkaisempia.

Kaikki vertailtavat pelimoottorit sisältävät erilaisia näkymätiloja kehitystyön helpottamiseksi. Unityssä, Unreal Enginessä ja Stingrayssa näkymätiloja voi vaihtaa suoraan editorin valikosta. Unreal Enginessä, CryEnginessä ja Lumberyardissa näkymätiloja voi vaihtaa konsolikomennoilla. Valikosta näkymätilan vaihtaminen voi olla uudelle kehittäjälle miellyttävämpää, mutta konsolikomentojen kautta näkymätilojen vaihtaminen on myös nopeaa pienen totuttelun jälkeen.

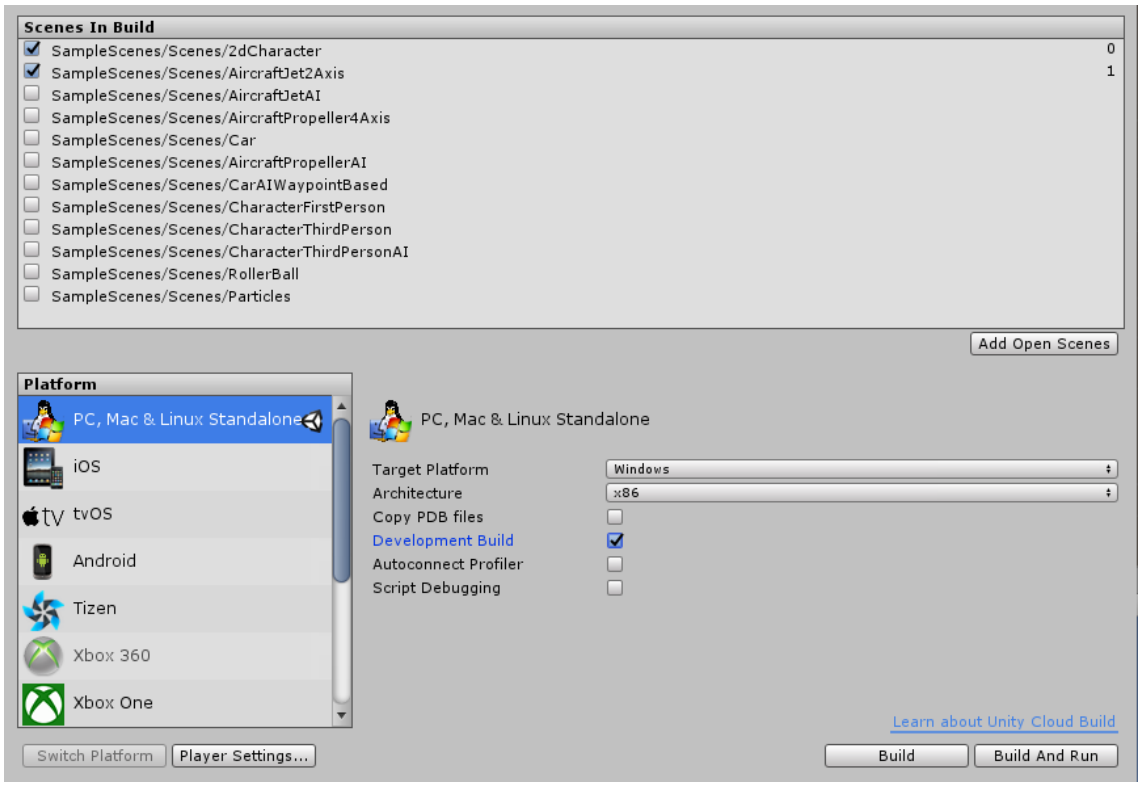
Unityn ja Unreal Enginen dokumentaatiot ovat kattavampia kuin muiden vertailtavien moottoreiden dokumentaatiot. Unity sisältää hieman vähemmän tietoa erilaisista näkymätiloista kuin Unreal Engine. CryEnginen dokumentaation kattavuus on lähellä samaa tasoa kuin Unityssä ja Unreal Enginessä. Lumberyardin dokumentaatio käsittelee Lumberyardin uuden profilointityökalun hyvin, mutta dokumentaatio ei sisällä ohjeistusta konsolikomentojen käyttöön, vaikka ne toisivat profilointiin paljon monipuolisuutta. Stingrayn dokumentaatio on vertailtavista moottoreista huonoin. Stingrayn dokumentaatio sisältää erittäin vähän ohjeistusta profiloinnin tehokkaaseen käyttämiseen eikä se myöskään sisällä kattavaa tietoa erilaisista näkymätiloista.

12 Tuetut alustat ja suoritettavan sovelluksen kasaaminen

Pelin valmistuttua ja sen kehityskaaren aikana kehitettävä peli kasataan kohdealustalla suoritettavaksi. Tässä osiossa luodaan katsaus pelimoottoreiden tukemiin kohdealustoihin, suoritettavan sovelluksen kasaamisen helppouteen sekä dokumentaation kattavuuteen.

12.1 Unity

Unity tukee kirjoitushetkellä 27:ää eri alustaa. Unityllä voi luoda työpöytä-, konsoli-, mobiili-, virtual reality -, selain- ja televisiopelejä. (Unity Tehcnologies 2016e.) Unity sisältää erillisen aukaistavan ikkunan, jonka kautta sovelluksen voi kasata. Tästä ikkunasta käyttäjä voi valita mitä kenttiä kasattavaan sovellukseen sisällytetään, mille alustalle peli kasataan ja halutaanko kasattuun peliin sisällyttää kehittäjätoimintoja. (Unity Documentation 2016n.) Kuvassa 31 on nähtävillä kasausvalikko. Kuvassa olevassa valikossa kasattaviksi kentiksi on valittu listasta kaksi ylintä kenttää, kohdealustaksi työpöytä ja kehitysversio.

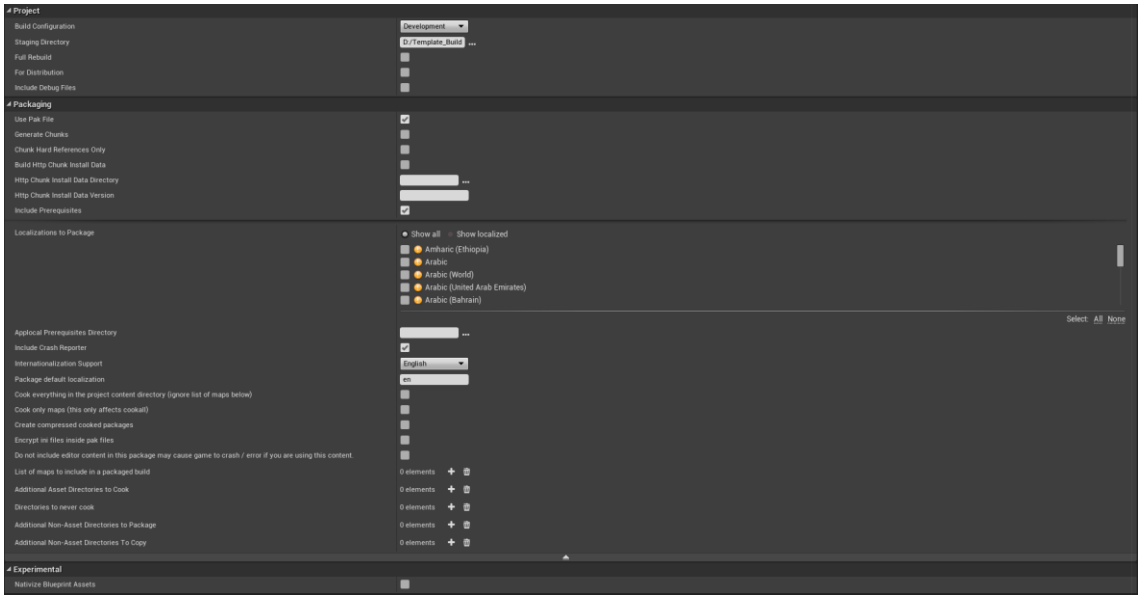


Kuva 31. Unityn kasausvalikko.

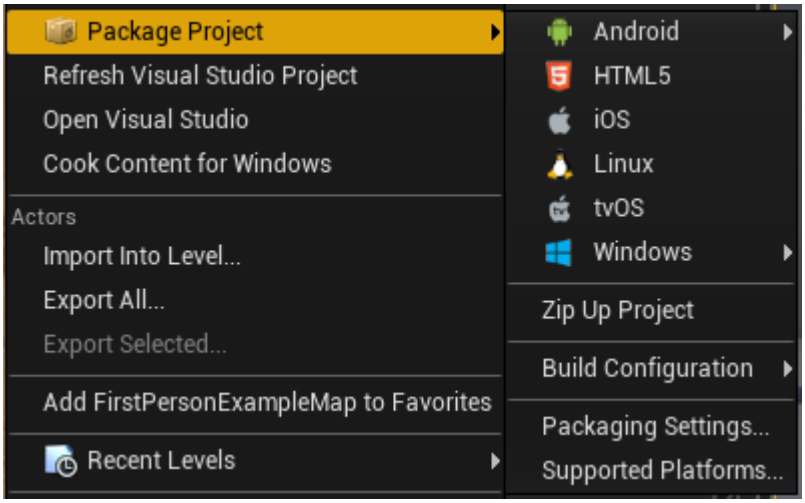
Unityn dokumentaatio ei käsittele pelin kasausta kovin laajasti. Dokumentaatio kattaa kuitenkin laajuudestaan huolimatta kasauksen hyvin, sillä pelin kasaus on Unityssä kokonaisuutena erittäin helppoa. Kehityksen aikana kasaamisen helppouden ja nopeuden takia onkin kannattaa tehdä suoritettavia sovelluksia usein.

12.2 Unreal Engine

Unreal Enginellä pelin voi kasata monelle yleiselle alustalle. Unreal Enginellä voi tehdä pelejä uuden sukupolven konsoleille, tietokoneille, mobiilille, virtual reality -laitteille, ja nettiselaimille. (Epic Games 2016c.) Unreal Enginen projektiasetuksissa on oma osionsa pelin kasausasetuksille. Kasausta varten tulee käyttäjän asettaa pelin oletuskenttä, joka ladataan pelin käynnistyessä. Kasausasetuksista käyttäjä voi valita muun muassa sisällytetäänkö peliin virheenetsintätiedostot, käytetäänkö tiivistettyjä tiedostoja, luodaanko kehittäjäversio vai julkaisuversio ja mitkä kentät kasaukseen halutaan sisällyttää. Itse kasaaminen tapahtuu päävalikossa sijaitsevasta valikosta. Tästä valikosta käyttäjä voi valita pelille kohdealustan. (Unreal Engine Documentation 2016o.) Unreal Enginen kasausasetukset ovat erittäin monipuoliset (Kuva 32) ja itse kasauksen tekeminen yksinkertaista (Kuva 33).



Kuva 32. Unreal Enginen pelin kasausasetukset.



Kuva 33. Unreal Enginen kasauksen kohdealustan valitseminen valikosta.

Unreal Enginen dokumentaatio kasauksen suhteen on erittäin hyvä. Dokumentaatio käsittelee ja selittää hyvin normaalit kasausasetukset, syvällisempiä asetuksia ei kuitenkaan käydä tarkasti läpi. Kokonaisuutena pelien kasaus Unreal Enginellä on helppoa ja testiversioiden kasaus kehityksen aikana on kohtuullisen nopeaa.

12.3 CryEngine

CryEnginellä voi kirjoitushetkellä kehittää pelejä Windows ja Linux pohjaisille tietokoneille, Oculus Riftille, Xbox Onelle ja Playstation 4:lle (CryEngine 2016d). Kehitettävän pelin kasaaminen suoritettavaksi kokonaisuudeksi tapahtuu CryEnginessä CryWAF nimisellä kasausjärjestelmällä. CryWAF pohjautuu yleiskäyttöiseen WAF-kasausjärjestelmään. CryWAFin monipuolinen käyttö vaatii konsolikomentojen ja json-kielen käyttämistä. CryWAF suoritetaan omana tiedostonaan moottorin ulkopuolella (CryEngine Documentation 2016h.)

CryEnginen kasausjärjestelmien käyttäminen käydään läpi dokumentaatiossa kohtalaisesti. Dokumentaatio käsittelee kasausjärjestelmän kohtuullisen kattavasti, mutta se tuntuu hankalalta seurata. Kasausjärjestelmä on myös monimutkainen Unityyn ja Unreal Engineen verrattuna ja sen käyttäminen vaatii käyttäjältä enemmän tietämystä järjestelmän toiminnasta.

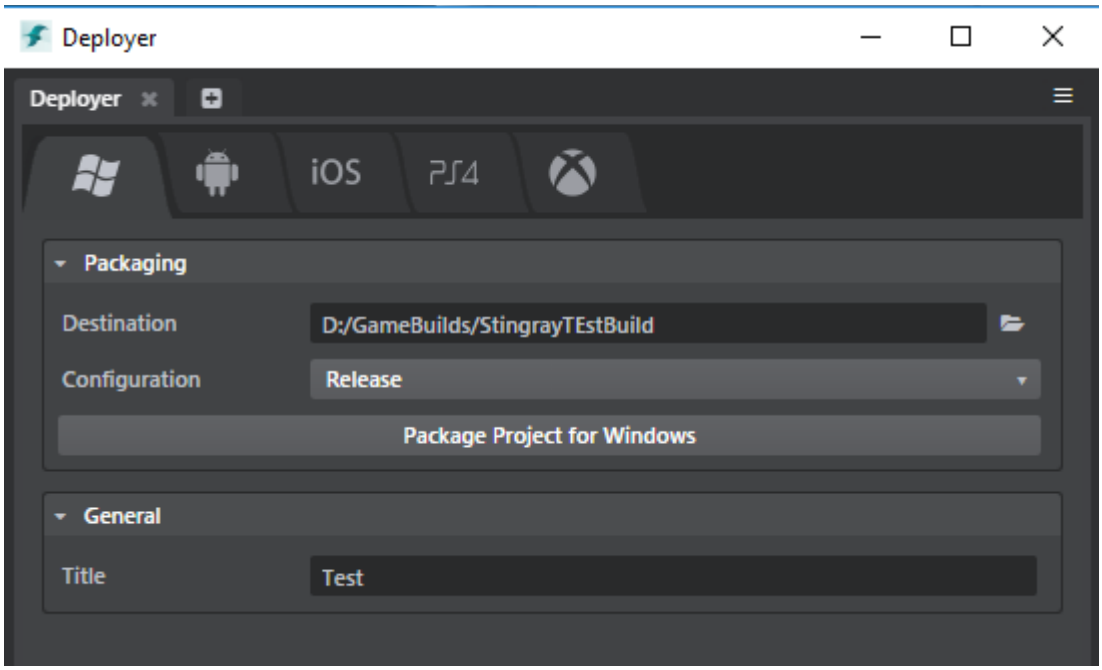
12.4 Lumberyard

Lumberyardilla voi kehittää pelejä tietokoneille, uuden sukupolven konsoleille, mobiilialustoille ja virtua reality -laitteille (Amazon Web Services 2016d). Suoritettavien pelien kasaukseen Lumberyard käyttää samaa WAF-järjestelmää kuin CryEngine. Järjestelmää käyttämällä käyttäjä voi tehdä erilaisia ominaisuuksia sisältäviä kasauksia. WAF-järjestelmän käyttäminen vaatii Lumberyardissa konsolikomentojen käyttämistä. (Lumberyard Documentation 2016r; Lumberyard Documentation 2016s.)

Lumberyardin dokumentaatio käsittelee kasauksen vaiheet hyvin. Dokumentaatio sisältää hyvän ohjeistuksen WAF-järjestelmän käyttämiseen ja ohjeita eri tarkoituksiin tehtävistä kasauksista. Dokumentaatio kattaa myös kasaamisen mobiilialustoille, mutta ei sisällä erillistä osiota konsolikehitykselle. Hyvästä dokumentaatiosta huolimatta WAF-järjestelmän käyttäminen on huomattavasti hankalampaa kuin Unityn ja Unreal Enginen tapa toteuttaa suoritettavan pelin kasaus. WAFin käyttäminen vaatii konsolikomentojen käyttämistä, joka voi olla kokemattomalle kehittäjälle iso kynnys.

12.5 Stingray

Stingraylla voi kehittää pelejä tietokoneille, mobiililaitteille, uuden sukupolven konsoleille sekä virtual reality -laitteille (Stingray Help 2016p). Pelin kasausta varten Stingray sisältää omana ikkunanaan toimivan Deployer-työkalun. Deployer-työkalun käyttäminen on yksinkertaista; käyttäjän tarvitsee vain valita kohdealusta, valita haluttu konfiguraatio, täyttää vaaditut kohdat ja painaa kasauspainiketta. Käyttäjä voi konfiguraatiota valitessaan valita kasattavaksi julkaisuversion, kehitysversion tai virheidenetsintäversion. (Stingray Help 2016q.) Deplyer-työkalu on ulkoasultaan ja käytettävyydeltään yksinkertainen (Kuva 34).



Kuva 34. Pelin kasaus Stingrayssa Deployer-työkalulla.

Stingrayn dokumentaatio pelin kasauksen suhteen on hyvä. Dokumentaatio käsittelee Deployer-työkalun käytön kattavasti ja sisältää erilliset osiot eri alustoille kasauksesta. Dokumentaatiosta ei kuitenkaan löydy kirjoitushetkellä tietoa siitä, kuinka käyttäjä valitsee kasaukseen sisällytettävät kentät. Kehittäjän kannalta pelin kasaus on Stingraylla erittäin helppoa selkeän työkalun takia. Kasaus on myös nopeaa, joten se on helppo tehdä useasti kehityksen aikana.

12.6 Yhteenveto

Pelin kasauksen helppoudessa vertailtavat pelimoottorit jakautuvat kahteen ryhmään. Unityssä, Unreal Enginessä ja Stingrayssa pelin kasaus on tehty käyttäjän kannalta erittäin helpoksi selkeillä työkaluilla. Näistä kolmesta moottorista Unreal Engine sisältää kuitenkin monipuolisimmat ominaisuudet ja säädöt kasaukselle tekemättä kasauksesta kuitenkaan käyttömukavuudeltaan hankalampaa. CryEnginen ja Lumberyardin tapa toteuttaa pelin kasaus on käyttäjälle hankalampi. Näissä moottoreissa kasaustyökalujen monipuolinen käyttö vaatii paljon konsolikomentojen käyttämistä ja syvempää tietämystä järjestelmien toiminnasta. Kaikkien vertailtavien pelimoottoreiden dokumentaatio käsittelee kasauksen vähintään hyvin sisältäen riittävät selitykset ja tiedot suoritettavan pelin kasaamiseksi.

Tuettuja alustoja vertailtaessa Unity on pelimoottoreista monipuolisin. Unityn tuki on erittäin kattava ja Unity tukee myös monia erilaisia virtual reality -laitteita. Unreal Engine sisältää Unityn jälkeen parhaimman tuen alustoille ja myös Unreal Engine sisältää hyvän tuen virtual reality -laitteille. Lumberyard ja Stingray ovat alustatueltaan samalla tasolla tarjoten hyvin samanlaisen tuen. CryEngine on vertailtavista pelimoottoreista ainut, joka ei kirjoitushetkellä sisällä tukea mobiililaitteille. Taulukkoon 3 on koottu taulukkomuodossa pelimoottoreiden tukemat alustat.

Taulukko 3. Vertailtavien pelimoottoreiden tukemat alustat.

	Tuetut alustat	Tuetut VR-laitteet
Unity	Android, iOS, Windows Phone, Tizen, Fire OS, PC, Mac, Linux PS4, Xbox One, Playstation Mobile, Playstation Vita, Wii U, WebGL, tvOS, Android TV, Samsung Smart TV	Oculus Rift, Gear VR, Playstation VR, Microsoft Hololens, HTC Vive, Google Daydream
Unreal Engine	PC, Playstation 4, Xbox One, Mac, Android, Linux, Steam OS, HTML 5	HTC Vive, Oculus Rift, Playstation VR, Google Daydream, Gear VR
CryEngine	PC, Linux, Xbox One, Playstation 4	Oculus Rift
Lumberyard	PC, OS X, Android, iOS, Xbox One, Playstation 4	Oculus Rift, HTC Vive, OSVR
Stingray	PC, iOS, Android, Playstation 4, Xbox One	Oculus Rift, HTC Vive

13 Pohdinta ja moottoreiden käyttösuositukset

Tämän opinnäytetyön tarkoituksena on helpottaa pelimoottorin valintaa hyvästä tarjonnasta. Pelimoottori on erittäin laaja kokonaisuus vertailtavia ominaisuuksia

ja onkin huomattavaa, että kaikkia ominaisuuksia ei voi sisällyttää yhteen opinnäytetyöhön. Tähän vertailuun valitut ominaisuudet ovat kuitenkin jokaisen kehitettävän pelin kannalta oleellisia ja pois on jätetty ominaisuuksia jotka voivat olla esimerkiksi enemmän yhtä peligenreä hyödyttäviä. On myös huomattava, että tämä vertailu on tehty pelimoottoreiden vakio-ominaisuuksia vertaillen ja useita tämänkin vertailun moottoreista voi muokata käytettävyydeltään ja toimintatavoiltaan erilaiseksi esimerkiksi moottorin kauppapaikalta ladattavilla sisällöillä ja työkaluilla. Tässä osiossa luodaan vielä lyhyt yleiskatsaus kaikkiin vertailussa mukana oleviin pelimoottoreihin ja pohditaan, kenelle tai minkälaiselle ryhmälle pelimoottoria voisi suositella.

13.1 Unity

Unity on erittäin suosittu pelimoottori, ja syyt on helppo nähdä. Unityn alustatuki on erittäin laaja, joten Unity todennäköisesti tukee alustaa, jolle peliä halutaan kehittää. Unityn toimintaperiaatteet ja työkalut ovat myös logiikaltaan ja käytettävyydeltään yksinkertaisia. Tämä helpottaa etenkin uutta kehittäjää, jolle pelimoottori ominaisuuksineen ei ole vielä kovin tuttu. Työkalujen helppokäyttöisyyttä edistää myös Unityn kattava ja erittäin hyvä dokumentaatio, joka käsittelee pelimoottorin ominaisuudet monipuolisesti ja selkeällä tyylillä. Unity on myös alkukustannuksiltaan kehittäjälle erittäin edullinen, mutta tiettyjen käyttöehtojen täyttyessä Unityn kuukausihinta kehittäjälle voi olla suurempi kuin muilla vertailtavilla moottoreilla. Tiedostotueltaan Unity on myös kattava ja tämän takia kehittäjä ei välttämättä tarvitse maksullisia kolmannen osapuolen sisällöntuotantosovelluksia.

Unityä on helppo suositella sekä uudelle että myös kokeneelle kehittäjälle. Unityn hyvä dokumentaatio ja kattava tutoriaaliosio ovat erittäin hyvä apu pelikehityksessä alkuun pääsemiseen. Myös kokeneille kehittäjille ja isommille projektiryhmille Unityn työkalujen käyttö on nopeaa. Mark Masters (2014) mainitsee blogikirjoituksessaa Unityn olevan hyvä pelimoottori mobiilipelien kehitykseen ja Unreal Enginen sekä CryEnginen soveltuvan paremmin graafisesti vaativampiin peleihin. Mastersin mielipidettä ei tämän vertailun pohjalta pysty kumoamaan, etenkin kun CryEnginellä ei mobiilipelejä voi tällä hetkellä kehittää.

13.2 Unreal Engine

Unreal Engine on perusteiltaan yllättävän samanlainen kuin Unity ja Unitystä Unreal Engineen siirtyminen tai toisinpäin on yllättävän helppoa. Unreal Engine sisältää hyvän tuen erilaisille alustoille, mutta se ei aivan yllä Unityn monipuolisuuden rinnalle. Unreal Engine toimintaperiaatteet ja työkalut ovat helppokäyttöisyydeltään Unityn tasolla. Pelilogiikan ohjelmoinnin tavat tuntuvat kuitenkin Unreal Enginessä hieman monipuolisemmilta visuaalisen ohjelmoinnin ja C++-kielen ansiosta. Käännettävä ohjelmointikieli kuitenkin hidastaa kehitystyötä aina ohjelmakoodin kääntövaiheessa. Unreal Engine sisältää erittäin hyvän ja kattavan dokumentaation. Dokumentaatiota täydentää monipuolinen videotutoriaaliosasto. Alkukustannuksiltaan Unreal Engine on kehittäjälle täysin ilmainen. Määrätyn liikevaihdon jälkeen viiden prosentin rojaltimaksu voi kuitenkin kasvaa yllättävän isoksi rahamääräksi, mikäli moottorilla kehitetyn tuotteen tuottama liikevaihto on suuri. Tiedostotueltaan Unreal Engine on Unityn tasolla ja tästä syystä myöskään Unreal Engine ei tarvitse välttämättä maksullisia kolmannen osapuolen sisällöntuotantosovelluksia.

Unreal Engineä on myös helppo suositella sekä uudelle että myös kokeneelle kehittäjälle. Unityn tapaan myös Unreal Enginen dokumentaatio ja tutoriaaliosio sisältävät hyvät lähtökohdat pelikehityksen aloittamiselle, mutta ne sisältävät myös edistyneempää ohjeistusta kokeneemmalle kehittäjälle. Unreal Enginen työkalut mahdollistavat myös suuren projektiryhmän toimimisen samassa projektissa. Mobiilipelituen osalta Unreal Engine on kehittynyt, mutta Unityn dokumentaatio ja työkalut mobiilikehitystä varten ovat vielä paremmat.

13.3 CryEngine

CryEngine on vertailtavista pelimoottoreista harmillisin tapaus. Pelimoottorin pohjateknologia tuntuu erittäin kehittyneeltä, mutta moottorin työkalujen käyttäminen ei tunnu yhtä käyttäjäystävälliseltä kuin Unityssä ja Unreal Enginessä.

Moottorin ja sen työkalujen käyttöä hankaloittaa myös CryEnginen vanhentuneelta ja vajaalta tuntuva dokumentaatio, joka käsittelee etenkin pelilogiikan ohjelmoinnin erittäin huonosti. CryEnginen alustatuki on huonoin vertailtavista pelimoottoreista ja se ei sisällä ollenkaan tukea mobiilipeleille. Ohjelmointi CryEnginessä C++- ja C#-kielillä on hankalaa etenkin uudelle kehittäjälle. C++-kieli ei sisällä Unreal Enginen C++-kielen tyylisiä avustuksia ja pelilogiikan ohjelmointidokumentaatio on erittäin huono. Visuaalinen ohjelmointi on CryEnginessä perusteiltaan yhtä helppoa kuin Unreal Enginessä, mutta dokumentaatio tämänkin osion suhteen on hieman vajaa. Käyttöliittymän toteuttamista varten CryEngine ei sisällä sisäänrakennettua työkalua C++-projekteissa vaan käyttäjän on käytettävä kolmannen osapuolen ohjelmistoja. Tiedostotuen osalta tuki FBX-muodolle on parantanut moottorin käytettävyyttä ilman maksullisia kolmannen osapuolen ohjelmistoja. CryEngine on kehittäjälle täysin ilmainen ilman rojaltimaksuja. Täysin reaaliaikainen valaistus on ominaisuus jota Unity ja Unreal Engine eivät kirjoitushetkellä sisällä.

CryEngineä on hankala suositella uudelle kehittäjälle tai pienelle projektiryhmälle ellei tarkoituksena ole tuottaa vain visuaalisesti hieno ympäristö ilman pelillisiä ominaisuuksia. Pelimaailman rakennustyökalut, optimointi ja reaaliaikainen valaistus ovat CryEnginessä erittäin hyvällä tasolla ja mahdollistavat hienojen pelimaailmojen luonnin nopeasti. Ohjelmointidokumentaation vajauksen takia ohjelmoinnissa alkuun pääseminen on kuitenkin CryEnginessä erittäin hankalaa ja voi viedä kehityksestä paljon aikaa. CryEngine soveltuu parhaiten isommalle ja kokeneemmalle projektiryhmälle jonka tavoitteena on kehittää visuaalisesti näyttävä peli ja jolla on myös kokemusta pelimoottorien ohjelmoinnista.

13.4 Lumberyard

Lumberyardin pohjautuminen CryEngineen on moottorin kannalta sekä hyvä että huono asia. Pohjateknologia on Lumberyardissakin kunnossa, mutta moottorin työkalut ja toimintatavat tuntuvat hieman hankalilta Unityyn ja Unreal Engineen verrattaessa. CryEngineen verrattuna Lumberyardin alustatuki on hieman parempi, mutta alustatuki ei pärjää Unityn tai Unreal Enginen monipuolisuudelle.

Lumberyardin C++-ohjelmointi on yhtä hankalaa kuin CryEnginessä, mutta Lumberyardia varten Amazon on kehittänyt komponenttijärjestelmän, jonka on tarkoitus tulevaisuudessa korvata kokonaan C++-ohjelmointi ja mahdollistaa kaikkien moottorin ominaisuuksien käytön lua-ohjelmointikielen kautta. Tulevaisuudessa tämä järjestelmä tulee toivottavasti helpottamaan ohjelmointia Lumberyardissa. Lumberyardin visuaalinen ohjelmointi on kuitenkin samalla tasolla kuin Unreal Enginessä ja CryEnginessä. Lumberyardin dokumentaatio ei ole erityisen hyvä. On kuitenkin muistettava, että moottori on erittäin uusi ja dokumentaatiota päivitetään jatkuvasti. Lumberyard ei tue kirjoitushetkellä animoituja objekteja FBX-tiedostomuodossa ja Lumberyard on ainut vertailtavista pelimoottoreista, josta tämä ominaisuus puuttuu. Tämän puutteen takia käyttäjän tulee käyttää maksullista kolmannen osapuolen ohjelmistoa animoitujen objektien tuonnin mahdollistamiseksi, vaikka itse pelimoottori onkin täysin ilmainen ja rojaltimaksuton. CryEnginestä peritty reaaliaikainen valaistus on käyttäjän kannalta Lumberyardissakin hyvä asia.

Lumberyardia on sen kehityksen tässä vaiheessa hankalaa suositella käytettäväksi etenkin, jos sitä käyttävä kehittäjä ei omista Autodeskin sisällöntuotantoohjelmistoja. Moni Lumberyardin työkaluista on vielä kehitysvaiheessa ja pelimoottorin dokumentaatio on myös vielä hieman vajaan oloinen. Tämä hankaloittaa Lumberyardilla alkuun pääsemistä. Lumberyardin integraatio Twitch-striimauspalveluun on mielenkiintoinen ominaisuus ja jotkut kehittäjät voivat antaa sille suurtakin painoarvoa pelimoottoria valittaessa. Lumberyardiin pätee siis sama suositus kuin CryEngineen; Lumberyard soveltuu parhaiten isolle ja kokeneelle projektiryhmälle. Pelimoottoreista kiinnostuneen kannattaa kuitenkin seurata Lumberyardin kehittymistä, sillä kaikki tähän mennessä Lumberyardiin julkaistut päivitykset ovat tehneet moottorista helppokäyttöisemmän ja monipuolisemman.

13.5 Stingray

Stingray on varsin uusi moottori ja sen omistaa paljon resursseja omaava Autodesk. Käytettävyydeltään ja toiminnoiltaan Stingray on yllättävän samankaltainen

Unityn ja Unrealin kanssa, mutta alustatuki on Stingrayssa hieman huonompi. Logiikan ohjelmoinnin toteutus Stingrayssa on verrattavissa sekä Unityyn että Unreal Engineen. Stingrayssa käytettävä tulkattava lua-ohjelmointikieli on helppoa ja nopeaa käyttää ja uudelle kehittäjälle kynnys aloittaa ohjelmointi tulkattavalla kielellä on todennäköisesti pienempi kuin käännettävällä C++-kielellä. Visuaalinen ohjelmointi on yhtä helppoa kuin muissa visuaalisen ohjelmoinnin sisältävissä vertailtavissa moottoreissa. Stingrayn dokumentaatio on kohtalainen. Dokumentaatio kattaa hyvin kaikki moottorin käytölle oleelliset perusteet, mutta tutoriaaleja ja syventäviä oppaita soisi olevan moottorille enemmän. Stingrayn tiedostotuki on hyvällä tasolla ja integraatio Autodeskin sisällöntuotantosovelluksiin on erittäin mielenkiintoinen ominaisuus, joka antaa pelimoottorille varmasti lisäarvoa, mikäli kehittäjä on tottunut käyttämään kyseisiä sisällöntuotantosovelluksia. Stingray on vertailtavista pelimoottoreista ainut, joka sisältää pakollisen kuukausimaksun. On kuitenkin huomattavaa, että Stingray sisältyy Autodeskin Maya LT -lisenssiin, joka on kuukausimaksultaan samanhintainen kuin pelkkä Stingrayn lisenssi.

Stingrayn suurin etu Unityyn ja Unreal Engineen verrattuna on sen hyvä integraatio Autodeskin sisällöntuotantosovelluksiin. Mikäli kehittäjä tai projektiryhmä maksaa jo valmiiksi kuukausimaksua Maya LT:stä, Mayasta tai 3dsMaxista voi Stringrayn valinta olla hyvinkin perusteltua. Muussa tapauksessa Stingraylla ei välttämättä ole käyttäjän kannalta selviä etuja Unityyn ja Unreal Engineen verrattuna. CryEngineen ja Lumberyardiin verrattuna Stingraylla on uuden kehittäjän helpompi päästä alkuun paremman dokumentaation ansiosta. Stingrayn dokumentaatio ei kuitenkaan yllä samalle tasolle kuin Unityssä ja Unreal Enginessä. Stingraysta on saatavilla kokeiluversio, joka mahdollistaa moottorin testauksen ennen kuukausimaksujen maksamista. Valinta Unityn, Unreal Enginen ja Stingrayn välillä voi riippua myös henkilökohtaisesta mieltymyksestä ja onkin hienoa, että kaikkia näitä moottoreita voi vähintään kokeilla ilmaiseksi.

Lähteet

Amazon Game Dev. 2016. Amazon GameDev Tutorials. https://gamedev.amazon.com/forums/tutorials. 9.11.2016.

Amazon Web Services. 2016a. AWS Service Terms. 29.7.2016. https://aws.amazon.com/service-terms/. 29.9.2016.

Amazon Web Services. 2016b. Lumberyard Details. https://aws.amazon.com/lumberyard/details/. 8.11.2016.

Amazon Web Services. 2016c. Amazon Lumberyad (beta). https://aws.amazon.com/documentation/lumberyard/. 9.11.2016.

Amazon Web Services. 2016d. Lumberyard. https://aws.amazon.com/lumberyard/. 19.11.2016.

Andrews, M. 2010. Game UI Discoveries: What Players Want. 23.2.2010. http://www.gamasutra.com/view/feature/132674/game_ui_discoveries_what_players_.php. 21.9.2016.

Audiokinetic. 2016. Wwise Products. https://www.audiokinetic.com/products/wwise/ 19.9.2016.

Autodesk. 2016a. FBX. http://www.autodesk.com/products/fbx/overview. 7.11.2016.

Autodesk. 2016b. The Making Of Helldivers. http://static-dc.autodesk.net/content/dam/autodesk/www/products/stingray/responsive-center/case-studies/Arrowhead%20Game%20Studios_Customer_Story.pdf. 29.9.2016.

Autodesk. 2016c. Stingray Subscribe. http://www.autodesk.com/products/stingray/subscribe. 8.11.2016.

Autodesk. 2016d. Maya LT Subscribe. http://www.autodesk.com/products/maya-lt/subscribe. 8.11.2016.

Autodesk Maya Documentation. 2016. Create A Stingray Physically Based Shader. 20.4.2016. https://knowledge.autodesk.com/support/maya-lt/learn-explore/caas/CloudHelp/cloudhelp/2016/ENU/MayaLT/files/GUID-13B0F0A6-39D5-4668-A722-DD743877D2E3-htm.html. 16.10.2016.

Barr, J. 2016. Lumberyard + Amazon Gamelift + Twitch For Games On AWS. 9.2.2016 https://aws.amazon.com/blogs/aws/lumberyard-amazon-gamelift-twitch-for-games-on-aws/. 19.9.2016.

Campo Santo Studio. 2016. About Firewatch. http://www.firewatchgame.com/about/. 28.9.2016

Chapple, C. 2014. 8 Top UI Tools. 28.11.2014. http://www.develop-online.net/tools-and-tech/8-top-ui-tools/0200621. 20.9.2016.

Coke And Code. 2016. Path Finding. http://www.cokeandcode.com/main/tutorials/path-finding/. 8.11.2016.

Collins, B. 2015. Autodesk Drops Its New Stingary Game Engine. http://blog.digitaltutors.com/autodesk-drops-new-stingray-game-engine/. 29.9.2016.

Crowder, T.J. 2013. Scripting Language Vs Programming Language, ylin vastaus. http://stackoverflow.com/questions/17253545/scripting-language-vs-programming-language. 22.9.2016.

CryEngine Answers. 2016. Is It Possible To Have Custom Shaders Without Altering The Engine's Source Code. 21.6.2016. https://answers.cryengine.com/questions/1266/is-it-possible-to-have-custom-shaders-including-ge.html. 14.10.2016.
CryEngine. 2016a. CRYENGINE Showcase GDC 2016. https://www.youtube.com/watch?v=wcnrt1pX5XA. 8.11.2016.
CryEngine. 2016b. CryEngine. https://www.cryengine.com/. 9.11.2016.
CryEngine. 2016b. 5.2 Highlight – FBX Static Import 2.0. https://www.youtube.com/watch?v=o4vpWF58qIE. 4.10.2016.
CryEngine. 2016c. 5.2 Highlight – FBX Anim Import. https://www.youtube.com/watch?v=TUDWb1CMETE. 4.10.2016.
CryEngine. 2016d. Platforms. https://www.cryengine.com/features/platforms. 18.11.2016.
CryEngine Documentation. 2016a. CRYENGINE Exporter in 3dsMax. 27.6.2016. http://docs.cryengine.com/display/CEMANUAL/CRYENGINE+Exporter+in+3dsMax. 4.10.2016.
CryEngine Documentation. 2016b. Physics Proxy. 13.1.2016. http://docs.cryengine.com/display/SDKDOC2/Physics+Proxy. 4.10.2016.
CryEngine Documentation. 2016c. Using C#'s UI System (Tutorial Series). 19.9.2016. http://docs.cryengine.com/pages/viewpage.action?pageId=25536751. 19.10.2016.
CryEngine Documentation. 2016d. Flow Graph Editor. http://docs.cryengine.com/display/CEMANUAL/Flow+Graph+Editor. 26.10.2016.
CryEngine Documentation. 2016e. C++ Programming Tutorials. 7.3.2016. http://docs.cryengine.com/pages/viewpage.action?pageId=23298338. 26.10.2016.
CryEngine Documentation. 2016f. CE# Framework. 19.9.2016. http://docs.cryengine.com/display/CEPROG/CE%23+Framework. 26.10.2016.
CryEngine Documentation. 2016g. Voxel-Based Global Illumination. 18.7.2016. http://docs.cryengine.com/display/SDKDOC2/Voxel-Based+Global+Illumination. 3.11.2016.
CryEngine Documentation. 2016h. CryEngine Build System. 10.3.2016. http://docs.cryengine.com/display/CEPROG/CRYENGINE+Build+System. 18.11.2016.
CryEngine Documentation. 2013a. Material Editor And Shaders. 15.7.2013. http://docs.cryengine.com/display/SDKDOC2/Material+Editor+and+Shaders. 13.10.2016.
CryEngine Documentation. 2013b. Lighting. http://docs.cryengine.com/display/SDKDOC4/Lighting. 3.11.2016.
CryEngine Documentation. 2013c. Advanced Effects. 26.7.2016. http://docs.cryengine.com/display/SDKDOC2/Advanced+Effects. 3.11.2016.
CryEngine Documentation. 2015a. Physically Based Shading. 3.8.2015. http://docs.cryengine.com/display/SDKDOC2/Physically+Based+Shading. 13.10.2016.
CryEngine Documentation. 2015b. Video Playback With GFxVideo. 5.1.2015. http://docs.cryengine.com/display/SDKDOC2/Video+Playback+with+GFxVideo. 13.10.2016.

CryEngine Documentation. 2015c. Light Entitites. 13.4.2015. http://docs.cryengine.com/display/SDKDOC2/Light+Entities. 3.11.2016.
CryEngine Documentation. 2015d. Setting Up Weather Effects. 19.10.2015. http://docs.cryengine.com/display/SDKDOC2/Setting+Up+Weather+Effects#SettingUpWeatherEffects-Earthquakes. 3.11.2016.
CryEngine Documentation. 2015e. Anti-Aliasing & Supersampling. 17.12.2015. http://docs.cryengine.com/pages/viewpage.action?pageId=1605684. 3.11.2016.
CryEngine Documentation. 2015f. Profiling. 10.6.2015. http://docs.cryengine.com/display/SDKDOC2/Profiling. 15.11.2016.
CryEngine Documentation. 2014a. User Interface. 4.6.2014. http://docs.cryengine.com/display/SDKDOC4/User+Interface. 19.10.2016.
CryEngine Documentation. 2014b. Creating A Crosshair For Top-Down Games. 4.6.2014. http://docs.cryengine.com/display/SDKDOC4/Creating+A+Crosshair+For+Top-Down+Games. 19.10.2016.
CryEngine Documentation. 2014c. Creating UI Using Vectorian Giotto And FlashDevelop. 17.11.2014. http://docs.cryengine.com/display/SDKDOC4/Creating+UI+Using+Vectorian+Giotto+and+FlashDevelop. 19.10.2016.
CryEngine Documentation. 2014d. Lua Scripting. 27.4.2016. http://docs.cryengine.com/display/SDKDOC4/Lua+Scripting. 26.10.2016.
CryEngine Documentation. 2014e. Environment Probes. 15.9.2014. http://docs.cryengine.com/display/SDKDOC2/Environment+Probes. 3.11.2016.
CryEngine Documentation. 2014f. Particles. 13.4.2014. http://docs.cryengine.com/display/SDKDOC2/Particles. 3.11.2016.
CryEngine Forum. 2012. Actionscript 3. 29.3.2012. https://www.cryengine.com/community/viewtopic.php?f=373&t=86991. 19.10.2016.
Crytek. 2016a. Limited License Agreement For The Use Of The CryEngine. https://www.cryengine.com/ce-terms. 28.9.2016.
Crytek. 2016b. Get CryEngine Today. https://www.cryengine.com/get-cryengine. 8.11.2016.
Enger, M. 2013. Game Engine: How Do They Work? 20.6.2013. http://www.giantbomb.com/profile/michaelenger/blog/game-engines-how-do-they-work/101529/. 4.9.2016.
Epic Games. 2016a. Unreal ® Engine End User License Agreement. https://www.unrealengine.com/eula. 28.9.2016.
Epic Games. 2016b. Check Out These Impressive Projects From China. 9.4.2016. https://www.unrealengine.com/blog/check-out-these-impressive-projects-from-china. 28.9.2016.
Epic Games. 2016c. FAQ. https://www.unrealengine.com/faq. 17.11.2016.
Fagerholt, E. Lorentzon, M. 2009. Beyond The HUD – User Interfaces For Increased Player Immersion In FPS Games. Chalmers University Of Technology. Department Of Computer Science And Engineering. Opinnäytetyö. http://publications.lib.chalmers.se/records/fulltext/111921.pdf. 21.9.2016.
Fatshark. 2016. About The Game. http://www.vermintide.com/about-the-game/. 29.9.2016.

Flanagan, J. 2014. Game Engine Analysis And Comparison. https://www.gamesparks.com/blog/game-engine-analysis-and-comparison/. 29.11.2016.

GameDev. 2015. How Can People Recognize What Engine A Game Uses? 14.12.2016. http://gamedev.stackexchange.com/questions/3154/how-can-people-recognize-what-engine-a-game-uses. 10.10.2016.

GameFromScratch. 2016. A Closer Look At The Stingray Engine. 13.4.2016. http://www.gamefromscratch.com/post/2016/04/13/A-Closer-Look-at-The-Stingray-Engine.aspx. 28.10.2016.

Geig, M. 2015. Integrating Unity Ads. 9.4.2016. https://blogs.unity3d.com/2015/04/09/integrating-unity-ads/. 19.9.2016

Graft, K. 2015. When Artificial Intelligence In Video Games Becomes…Artificially Intelligent. 22.9.2015. http://www.gamasutra.com/view/news/253974/When_artificial_intelligence_in_video_games_becomesartificially_intelligent.php. 24.9.2016.

Kalderon, E. 2011. Game Engines: What They Are And How They Work. 9.4.2011. https://nullpwd.wordpress.com/2011/05/09/game-engines-what-they-are-and-how-they-work/. 5.9.2016.

Kissner, M. 2015. Writing a Game Engine From Scratch – Part1: Messaging. 27.1.2015. http://www.gamasutra.com/blogs/MichaelKissner/20151027/257369/Writing_a_Game_Engine_from_Scratch__Part_1_Messaging.php. 7.9.2016.

Lumberyard Documentation. 2016a. Creating And Launching Game Projects. http://docs.aws.amazon.com/lumberyard/latest/userguide/configurator-projects.html. 1.10.2016.

Lumberyard Documentation. 2016b. 3ds Max Export Tools. http://docs.aws.amazon.com/lumberyard/latest/userguide/char-model-export-max.html. 5.10.2016.

Lumberyard Documentation. 2016c. Working With The FBX Importer. http://docs.aws.amazon.com/lumberyard/latest/userguide/char-fbx-importer.html. 5.10.2016.

Lumberyard Documentation. 2016d. Materials. http://docs.aws.amazon.com/lumberyard/latest/gettingstartedguide/materials.html. 14.10.2016.

Lumberyard Documentation. 2016e. Materials And Shaders. http://docs.aws.amazon.com/lumberyard/latest/userguide/mat-intro.html. 14.10.2016.

Lumberyard Documentation. 2016f. Developing A Custom Shader. http://docs.aws.amazon.com/lumberyard/latest/userguide/mat-shaders-custom-dev-intro.html. 14.10.2016.

Lumberyard Documentation. 2016g. UI System. http://docs.aws.amazon.com/lumberyard/latest/userguide/ui-editor-intro.html. 20.10.2016.

Lumberyard Documentation. 2016h. Flow Graph System. http://docs.aws.amazon.com/lumberyard/latest/userguide/fg-editor-intro.html. 27.10.2016.

Lumberyard Documentation. 2016i. Lumberyard For Programmers. http://docs.aws.amazon.com/lumberyard/latest/developerguide/Lumberyard-intro.html. 27.10.2016.

Lumberyard Documentation. 2016j. Entity System. http://docs.aws.amazon.com/lumberyard/latest/developerguide/entity-intro.html. 27.10.2016.

Lumberyard Documentation. 2016k. Component Entity System. http://docs.aws.amazon.com/lumberyard/latest/developerguide/component-entity-system-intro.html. 27.10.2016.
Lumberyard Documentation. 2016l. Lua Scripting. http://docs.aws.amazon.com/lumberyard/latest/developerguide/lua-scripting-intro.html. 27.10.2016.
Lumberyard Documentation. 2016m. Lighting The Scene. http://docs.aws.amazon.com/lumberyard/latest/gettingstartedguide/lighting.html. 4.11.2016.
Lumberyard Documentation. 2016n. Component Reference. http://docs.aws.amazon.com/lumberyard/latest/userguide/component-components.html. 4.11.2016.
Lumberyard Documentation. 2016o. Customizing Post-Processing Effects. http://docs.aws.amazon.com/lumberyard/latest/userguide/effect-groups-customizing-intro.html. 4.11.2016.
Lumberyard Documentation. 2016p. Particle Effects System. http://docs.aws.amazon.com/lumberyard/latest/userguide/particle-intro.html. 4.11.2016.
Lumberyard Documentation. 2016q. Profiler. http://docs.aws.amazon.com/lumberyard/latest/developerguide/profiler-intro.html. 15.11.2016.
Lumberyard Documentation. 2016r. Waf Build System. http://docs.aws.amazon.com/lumberyard/latest/userguide/waf-intro.html. 19.11.2016.
Lumberyard Documentation. 2016s. http://docs.aws.amazon.com/lumberyard/latest/userguide/game-build-intro.html. 19.11.2016.
Lumberyard Tutorials. 2016. Tutorials. https://gamedev.amazon.com/forums/tutorials. 20.10.2016.
Luukkonen, T. 2015. Resource Management In Game Engine Development. Kajaanin Ammattikorkeakoulu. Business Information Technology. Opinnäytetyö. https://www.theseus.fi/bitstream/handle/10024/101246/Luukkonen_Timo.pdf?sequence=1. 24.9.2016.
Masters, M. 2014. Unity, Source 2, Unreal Engine 4, or CryEngine – Which Game Engine Should I Choose?. 2014. http://blog.digitaltutors.com/unity-udk-cryengine-game-engine-choose/. 13.9.2016.
Microsoft. 2016a. Why .Net. https://www.microsoft.com/net/intro. 7.11.2016.
Microsoft. 2016b. Kinect Hardware. https://developer.microsoft.com/en-us/windows/kinect/hardware. 18.9.2016.
Microsoft Support. 2016. What Is A DLL. https://support.microsoft.com/en-us/kb/815065. 7.11.2016.
Mono Project. 2016. Mono Project. http://www.mono-project.com/. 7.11.2016.
Nintendo. 2016. Hardware And Software Sales Units. 30.6.2016. https://www.nintendo.co.jp/ir/en/sales/hard_soft/. 18.9.2016.
Nutt, C. 2016. Amazon Launches New, Free, High-Quality Game Engine: Lumberyard. 9.2.2016. http://www.gamasutra.com/view/news/265425/Amazon_launches_new_free_highquality_game_engine_Lumberyard.php. 29.9.2016.
Russell Jeff. 2015. Basic Theory Of Physically-Based Rendering. https://www.marmoset.co/toolbag/learn/pbr-theory. 10.10.2016.
Scaleform Help. 2016. http://help.autodesk.com/view/ScaleformStudio/ENU/. 20.10.2016.

Slick, J. 2016. 3D Model Components – Vertices, Edges, Polygons & More. 21.10.2016. https://www.lifewire.com/3d-model-components-1952. 7.11.2016.

Stingray Help. 2016a. Stingray Help. http://help.autodesk.com/view/Stingray/ENU/. 9.11.2016.

Stingray Help. 2016b. Interop With Maya, Maya LT, or 3ds Max. http://help.autodesk.com/view/Stingray/ENU/?guid=__stingray_help_importing_assets_link_to_dcc_html. 5.10.2016.

Stingray Help. 2016c. Import An FBX File. http://help.autodesk.com/view/Stingray/ENU/?guid=__stingray_help_importing_assets_import_an_fbx_html. 5.10.2016.

Stingray Help. 2016d. Create A Physics Actor. http://help.autodesk.com/view/Stingray/ENU/?guid=__stingray_help_creating_gameplay_physics_create_physics_actor_html. 5.10.2016.

Stingray Help. 2016e. Shading. http://help.autodesk.com/view/Stingray/ENU/?guid=__stingray_help_lighting_rendering_shading_and_materials_html. 16.10.2016.

Stingray Help. 2016f. Create A Parent Material. http://help.autodesk.com/view/Stingray/ENU/?guid=__stingray_help_lighting_rendering_shading_and_materials_create_parent_material_html. 16.10.2016.

Stingray Help. 2016g. Creating User Interfaces. http://help.autodesk.com/view/Stingray/ENU/?guid=__stingray_help_creating_uis_html. 20.10.2016.

Stingray Help. 2016h. Lua Or Flow: Which Do I Use? http://help.autodesk.com/view/Stingray/ENU/?guid=__stingray_help_creating_gameplay_lua_vs_flow_html. 28.10.2016.

Stingray Help. 2016i. Creating Gameplay. http://help.autodesk.com/view/Stingray/ENU/?guid=__stingray_help_creating_gameplay_html. 28.10.2016.

Stingray Help. 2016j. Video Tutorials. http://help.autodesk.com/view/Stingray/ENU/?guid=__stingray_help_video_tutorials_html. 28.10.2016.

Stingray Help. 2016k. Lighting. http://help.autodesk.com/view/Stingray/ENU/?guid=__stingray_help_lighting_rendering_lighting_html. 6.11.2016.

Stingray Help. 2016l. The Shading Environment And Post Effects. http://help.autodesk.com/view/Stingray/ENU/?guid=__stingray_help_lighting_rendering_shading_environment_html. 6.11.2016.

Stingray Help. 2016m. Creating FX. http://help.autodesk.com/view/Stingray/ENU/?guid=__stingray_help_creating_effects_html. 6.11.2016.

Stingray Help. 2016n. Ways To Get Runtime Feedback. http://help.autodesk.com/view/Stingray/ENU/?guid=__stingray_help_playtesting_feedback_html. 15.11.2016.

Stingray Help. 2016o. Console Commands. http://help.autodesk.com/view/Stingray/ENU/?guid=__stingray_help_reference_console_commands_html. 15.11.2016.

Stingray Help. 2016p. Supported Platforms. http://help.autodesk.com/view/Stingray/ENU/?guid=__stingray_help_deploying_platform_support_html. 20.11.2016.

Stingray Help. 2016q. Deploying And Building. http://help.autodesk.com/view/Stingray/ENU/?guid=__stingray_help_deploying_html. 20.11.2016.
Stingray Forum. 2016a. Extending the editor. 2.4.2016. https://forums.autodesk.com/t5/stingray-forum/extending-the-editor/td-p/6023618. 1.10.2016.
Stuart, R. 2009. The Hidden Story Of The 3D Engine – By The People Who Write Them. 14.12.2009. https://www.theguardian.com/technology/gamesblog/2009/dec/14/games-gameculture. 31.8.2016.
Tom' Hardware. 2003. What Does Co-Op Mode Mean. 24.12.2003. http://www.tomshardware.co.uk/forum/1324-13-what-mode-mean. 7.11.2016.
Unity Asset Store. 2015. RAIN AI For Unity. 15.4.2015. https://www.assetstore.unity3d.com/en/#!/content/23569. 24.9.2016.
Unity Documentation. 2016a. Extending The Editor. https://docs.unity3d.com/Manual/ExtendingTheEditor.html. 1.10.2016
Unity Documentation. 2016b. Importing Assets. https://docs.unity3d.com/Manual/ImportingAssets.html. 3.10.2016.
Unity Documentation. 2016c. Animation From External Sources. https://docs.unity3d.com/Manual/AnimationsImport.html. 3.10.2016
Unity Documentation. 2016d. Mesh Components/Meshes. https://docs.unity3d.com/Manual/class-Mesh.html. 3.10.2016.
Unity Documentation. 2016e. Materials, Shaders & Textures. https://docs.unity3d.com/Manual/Shaders.html. 12.10.2016.
Unity Documentation. 2016f. UI. https://docs.unity3d.com/Manual/UISystem.html. 18.10.2016.
Unity Documentation. 2016g. Scripting Overview. https://docs.unity3d.com/Manual/ScriptingConcepts.html. 24.10.2016.
Unity Documentation. 2016h. Lighting. https://docs.unity3d.com/Manual/LightingOverview.html. 1.11.2016.
Unity Documentation. 2016i. Image Effect Reference. https://docs.unity3d.com/Manual/comp-ImageEffects.html. 31.10.2016.
Unity Documentation. 2016j. Visual Effects Reference. https://docs.unity3d.com/Manual/comp-Effects.html. 31.10.2016.
Unity Documentation. 2016k. Particle Systems Reference. https://docs.unity3d.com/Manual/PartSysReference.html. 31.10.2016.
Unity Documentation. 2016l. The Profiler Window. https://docs.unity3d.com/Manual/Profiler.html. 12.11.2016.
Unity Documentation. 2016m. Scene View Control Bar. https://docs.unity3d.com/Manual/ViewModes.html. 12.11.2016.
Unity Documentation. 2016n. Build Settings. https://docs.unity3d.com/Manual/BuildSettings.html. 16.11.2016.
Unity Live Training. 2016. Live Online Training Courses. https://unity3d.com/learn/live-training. 24.10.2016.
Unity Store. 2016. Welcome To Unity. https://store.unity.com/. 8.11.2016.
Unity Technologies. 2016a. Unity Terms Of Service. https://unity3d.com/legal/terms-of-service. 28.9.2016.

Unity Technologies. 2016b. Unity Pro, Unity Plus and Unity Personal Software Additional Terms. https://unity3d.com/legal/terms-of-service/software. 8.11.2016.
Unity Technologies. 2016c. Learn. https://unity3d.com/learn. 9.11.2016.
Unity Technologies. 2016d. Community. https://unity3d.com/community. 9.11.2016.
Unity Technologies. 2016e. Multiplatform. https://unity3d.com/unity/multiplatform. 16.11.2016.
Unity Technologies. 2014. Card Life. https://unity3d.com/showcase/case-stories/hearthstone. 28.9.2016.
Unity Tutorials. 2016a. User Interface (UI). https://unity3d.com/learn/tutorials/topics/user-interface-ui. 18.10.2016.
Unity Tutorials. 2016b. Scripting. https://unity3d.com/learn/tutorials/topics/scripting. 18.10.2016.
Unreal Engine. 2016a. Behavior Trees. https://docs.unrealengine.com/latest/INT/Engine/AI/BehaviorTrees/index.html. 24.9.2016.
Unreal Engine. 2016b. Unreal Engine. https://www.unrealengine.com/. 9.11.2016.
Unreal Engine Documentation. 2015a. Extending The Editor. 22.1.2015. https://www.youtube.com/watch?v=zg_VstBxDi8&feature=youtu.be. 1.10.2016.
Unreal Engine Documentation. 2016a. Import Content – How To's. https://docs.unrealengine.com/latest/INT/Engine/Content/ImportingContent/index.html. 3.10.2016.
Unreal Engine Documentation. 2016b. Animation Sequences. https://docs.unrealengine.com/latest/INT/Engine/Animation/Sequences/index.html. 3.10.2016.
Unreal Engine Documentation. 2016c. Setting Up Collisions With Static Meshes. https://docs.unrealengine.com/latest/INT/Engine/Content/Types/StaticMeshes/HowTo/SettingCollision/index.html. 3.10.2016.
Unreal Engine Documentation. 2016d. Skeletal Mesh Actors. https://docs.unrealengine.com/latest/INT/Engine/Actors/SkeletalMeshActors/#collision. 3.10.2016.
Unreal Engine Documentation. 2016e. Materials. https://docs.unrealengine.com/latest/INT/Engine/Rendering/Materials/. 12.10.2016.
Unreal Engine Documentation. 2016f. UMG UI Designer. https://docs.unrealengine.com/latest/INT/Engine/UMG/. 18.10.2016.
Unreal Engine Documentation. 2016g. Programming Guide. https://docs.unrealengine.com/latest/INT/Programming/index.html. 25.10.2016.
Unreal Engine Documentation. 2016h. Blueprints Visual Scripting. https://docs.unrealengine.com/latest/INT/Engine/Blueprints/index.html. 25.10.2016.
Unreal Engine Documentation. 2016i. Lighting The Environment. https://docs.unrealengine.com/latest/INT/Engine/Rendering/LightingAndShadows/index.html. 2.11.2016.
Unreal Engine Documentation. 2016j. Use The Emissive Material Input. https://docs.unrealengine.com/latest/INT/Engine/Rendering/Materials/HowTo/EmissiveGlow/. 2.11.2016.

Unreal Engine Documentation. 2016k. Post Process Effects. https://docs.unrealengine.com/latest/INT/Engine/Rendering/PostProcessEffects/. 1.11.2016.
Unreal Engine Documentation. 2016l. Cascade Particle Systems. https://docs.unrealengine.com/latest/INT/Engine/Rendering/ParticleSystems/. 2.11.2016.
Unreal Engine Documentation. 2016m. Performance And Profiling. https://docs.unrealengine.com/latest/INT/Engine/Performance/index.html. 14.11.2016.
Unreal Engine Documentation. 2016n. View Modes. https://docs.unrealengine.com/latest/INT/Engine/UI/LevelEditor/Viewports/ViewModes/. 14.11.2016.
Unreal Engine Documentation. 2016o. Packaging Projects. https://docs.unrealengine.com/latest/INT/Engine/Basics/Projects/Packaging/. 17.11.2016.
Unreal Engine Wiki. 2015a. UMG, How To Extend A UUserWidget:: For UMG In C++. https://wiki.unrealengine.com/UMG,_How_to_extend_a_UUserWidget::_for_UMG_in_C%2B%2B.. 18.10.2016.
Unreal Engine Wiki. 2015b. UMG, Referencing UMG Widgets In Code. https://wiki.unrealengine.com/UMG,_Referencing_UMG_Widgets_in_Code. 18.10.2016.
WhatIs. 2007. Voxel. http://whatis.techtarget.com/definition/voxel. 7.11.2016.
Wikipedia. 2016a. Rasterisation. 10.8.2016. https://en.wikipedia.org/wiki/Rasterisation. 7.11.2016.
Wikipedia. 2016b. 3D Rendering. 29.9.2016 https://en.wikipedia.org/wiki/3D_rendering. 10.10.2016.
Wikipedia. 2016c. Shader. 2.10.2016. https://en.wikipedia.org/wiki/Shader. 11.10.2016.
Wikipedia. 2016d. Physics Engine. https://en.wikipedia.org/wiki/Physics_engine. 20.9.2016.
Wikipedia. 2016e. Path Finding. 24.4.2016. https://en.wikipedia.org/wiki/Pathfinding. 24.9.2016.
Wikipedia. 2016f. Unity (Game Engine). 26.9.2016. https://en.wikipedia.org/wiki/Unity_(game_engine). 27.9.2016.
Wikipedia. 2016g. Unreal Engine. 27.9.2016. https://en.wikipedia.org/wiki/Unreal_Engine. 28.9.2016.
Wikipedia. 2016h. CryEngine. 4.9.2016. https://en.wikipedia.org/wiki/CryEngine. 28.9.2016.