Mobiilipelit ja pelimoottorit

Tomi Tukiainen

Helsinki 06.11.2013

Pro gradu -tutkielma

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

HELSINGIN YLIOPISTO – HELSINGFORS UNIVERSITET – UNIVERSITY OF HELSINKI

Tiedekunta – Fakultet –Faculty	Laitos – Institution – Department
Matemaattis-luonnontieteellinen tiedekunta	Tietojenkäsittelytieteen laitos

Tekijä – Författare – Author

Tomi Tukiainen

Työn nimi – Arbetets titel – Title

Mobiilipelit ja pelimoottorit

Oppiaine – Läroämne – Subject

Tietojenkäsittelytiede

Työn laji – Arbetets art – Level	Aika – Datum – Month and year	Sivumäärä – Sidoantal – Number of pages
Pro gradu	6.11.2013	67 sivua + 15 liitesivua

Tiivistelmä – Referat – Abstract

Työssä esitellään lukijalle tietokonepelit ja mobiilipelimoottorit, jonka jälkeen edetään arvioimaan markkinoilta löytyviä pelimoottoreita pienen mobiilipelejä tuottavan peliyrityksen näkökulmasta. Arvioinnin tarkoituksena on löytää yritykselle parhaiten sopiva pelimoottori. Arvioinnissa hyödynnetään ISO/IEC 25000 sarjan ohjelmistojen laatustandardeja. Kahdestakymmenestä mobiilipelien tuottamiseen soveltuvasta pelimoottorista muodostetaan objektiivinen arvio, jonka perusteella yritykselle parhaalta vaikuttavaa pelimoottoria arvioidaan syvemmin prototyyppipeliprojektissa. Työn perusteella löydetään yritykselle hyvin soveltuva pelimoottori.

ACM Computing Classification System (CCS):

Software and its engineering → Interactive games
Software and its engineering → Middleware
General and reference → Evaluation

Avainsanat – Nyckelord – Keywords

pelit, arkkitehtuuri, pelimoottorit, grafiikka

Säilytyspaikka – Förvaringställe – Where deposited

Kumpulan tiedekirjasto, sarjanumero C-

Muita tietoja – Övriga uppgifter – Additional information

Sisältö

1 Johdanto

Pelimoottorit ovat pelialan väliohjelmistoja, jotka sisältävät peleissä usein tarvittavan perustoiminnallisuuden ja rakenteen valmiina. Pelimoottorit ovat vaikuttaneet koko peliteollisuuteen lisäämällä voimakkaasti peliohjelmien välistä uudelleenkäyttöä ja mahdollistamalla siten uusien pelien tuottamisen yhä tehokkaammin. Pelimoottoria käyttämällä voi vähentää pelituotannon työmäärää ja parantaa pelien laatua huomattavasti. Pelimoottori voi olla erityisen hyödyllinen työväline pienelle peliyritykselle, jolla ei ole laadukkaita pelituotannon työkaluja valmiina ja jonka ei kannata sitoa resursseja pelien perustoiminnallisuuden tai rakenteen toteuttamiseen.

Tässä tutkielmassa arvioidaan markkinoilta löytyviä pelimoottoreita pienen mobiilipelejä tuottavan yrityksen näkökulmasta. Arvioinnin tarkoituksena on löytää yritykselle parhaiten sopiva pelimoottori. Arvioinnissa tukeudutaan ISO/IEC 25000 -sarjan standardeihin [ISO11a]. ISO/IEC 25000 -sarjan standardit muodostavat ohjelmistojen laadusta loogisen ja yhtenäisen kokonaisuuden, joka auttaa arvioinnin suunnittelussa ja toteuttamisessa.

Aluksi työssä selvitetään yrityksen vaatimukset ja etsitään markkinoilta löytyvät mobiilipelimoottorivaihtoehdot. Seuraavaksi arvioidaan pelimoottoreiden laatua yrityksen näkökulmasta noudattaen ISO/IEC 25040 -standardissa esitettyä arviointiprosessia. Tällä tavoin saadaan muodostettua objektiivinen arvio kahdenkymmenen markkinoilta löytyvän pelimoottorin laadusta yrityksen näkökulmasta. Arvioinnin perusteella parhaalta vaikuttaneella pelimoottorilla toteutetaan prototyyppipeliprojekti, jossa testataan käytännössä pelimoottorin soveltuvuutta yrityksen käyttöön. Prototyyppipeliprojektin löydösten perusteella arvioidaan valitun pelimoottorin soveltuvuus yrityksen käyttöön.

Mobiilipelimoottorivertailussa tarvittavaa pohjatietoa kartutetaan luvussa 2 tutustumalla peleihin, tietokonepeleihin ja mobiilipeleihin. Tietokonepelien ja mobiilipelien kehitystä tarkastellaan teknisestä näkökulmasta sekä markkinoiden näkökulmasta. Luvussa 3 edetään tarkastelemaan peliohjelmistojen rakennetta. Tässä yhteydessä tutustutaan peliohjelmistojen tärkeimpiin rakenneosiin ja esitellään lukijalle käsite pelimoottori. Luvussa 5 esitellään neljä mobiilipelien tuottamiseen soveltuvaa pelimoottoria. Luvussa 5 esitellään ISO/IEC 25000 -sarjan standardit, joita hyödynnetään pelimoottorien laadun

arvioinnissa. Luvussa 6 kuvataan työn yhteydessä suoritettu pelimoottoreiden laadun arviointi.

2 Tietokonepelit ja peliteollisuus

Taustatieto tietokonepeleistä ja peliteollisuudesta on hyödyllistä peliohjelmistoissa tarvittavan toiminnallisuuden ymmärtämiselle. Tässä luvussa lähdetään liikkeelle lyhyestä katsauksesta leikkeihin ja peleihin, jonka jälkeen edetään tarkastelemaan tietokonepelejä. Myös peliteollisuuden kehittymistä käsitellään. Lopuksi esitellään mobiilipelit, jotka ovat tutkielmassa keskeisessä asemassa.

2.1 Pelit ja motivaatio pelaamiseen

Pelaamisen juuret ovat leikkimisessä. Termiä leikki käytetään kuvaamaan laajaa kirjoa vapaaehtoista, luonnostaan motivoivaa tekemistä, joka yleensä liittyy virkistystoimintaan ja vapaa-ajan viettoon. Leikkimistä esiintyy ihmisen lisäksi myös muiden kehittyneiden eläinlajien keskuudessa [Gar90, sivut 27-29].

Eläinten ja ihmisten luontainen halu leikkiä on saanut osakseen huomiota tiedeyhteisöltä. Evoluution näkökulmasta leikistä täytyy olla yksilölle merkittävä hyöty, koska toisaalta leikin välttämiseen on painavia syitä. Eläimet voivat esimerkiksi loukkaantua leikkiessään tai päätyä leikkiin keskittyessään pedon suuhun. Luontainen halu leikkiä on kuitenkin lajinkehityksessä säilynyt piirre, ja siksi sen uskotaan parantavan yksilön selviytymismahdollisuuksia auttamalla tätä kehittymään ja luomalla suhteita [Gar90, sivu 27]. Kuvassa 1 on kissanpentu leikkimässä saksanpähkinällä.

Kuva 1: Kissanpentu leikkimässä saksanpähkinällä.
Kuva: digidreamgrafix / FreeDigitalPhotos.net.

Roger Caillois asettaa leikit jatkumolle niiden määrämuotoisuuden mukaan [Cai61, sivu 13]. Cailloisin jatkumon toisena ääripäänä on paidia, joka tarkoittaa rakenteettomia, spontaaneja leikkejä. Toinen ääripää on ludus, johon kuuluvat määrämuotoiset leikit, joissa on selkeät säännöt. Ihmisten keskuudessa leikit tyypillisesti muovautuvat spontaaneista määrämuotoisemmiksi sitä mukaa kun leikkiessä sääntöjä kehitetään. Sääntöjen lisääminen, tarkentaminen ja muuttaminen on ihmiselle luonteenomaista. Pelit ovat määrämuotoisia leikkejä, joissa pätevät tietyt säännöt [Cai61, sivu 128].

Peli on myös itsenäisenä käsitteenä pyritty määrittelemään useilla erilaisilla tavoilla onnistumatta pääsemään yleisesti hyväksyttyyn määritelmään. Jesper Juul on ottanut lähtökohdakseen useita aiempia pelin määritelmiä ja pyrkinyt määrittelemään pelin kuuden ominaisuuden perusteella [Juu03].

1. Säännöt: pelit pohjautuvat sääntöihin.
2. Erilaiset, mitattavissa olevat lopputulokset: peleillä on useampia mahdollisia lopputuloksia.
3. Arvotetut lopputulokset: pelien mahdollisille lopputuloksille on annettu eri arvot, joidenkin ollessa huonoja ja toisten ollessa hyviä lopputuloksia.
4. Pelaajien vaivannäkö: pelaajien on nähtävä vaivaa vaikuttaakseen pelin lopputulokseen.
5. Pelaajat on sidottu lopputulokseen: pelaajat on sidottu pelin lopputuloksiin siten, että jotkut lopputulokset ovat pelaajan kannalta voittoisia ja toiset tappiollisia. Voittaessaan pelaaja on lähtökohtaisesti iloinen, ja hävitessään pelaaja on surullinen.
6. Neuvoteltavat seuraukset: samaa peliä voidaan pelata ilman tosielämän seurauksia tai neuvoteltujen tosielämän seurausten kanssa.

Annettua aktiviteettia voi verrata määritelmää vasten askel askeleelta, ja siten tulla lopputulokseen, onko aktiviteetti peli vai ei. Mikäli päädytään lopputulokseen, että annettu aktiviteetti ei ole peli, löytyy päätökselle perustelu ristiriidan aiheuttavista kohdista. Juul kuitenkin myöntää, että joissakin rajatapauksissa määritelmä ei toimi selkeästi, ja siten mielipide vaikuttaa päätökseen.

Motivaatio pelaamiseen

Pelaaminen on siis useimmiten perusluonteeltaan leikkimistä, ja motivaatio pelaamiseen vaikuttaa syntyvän evoluutiossa lajiimme kehittyneen leikkimistä suosivan mekanismin kautta. Pelaamiseen motivoiva mekanismi näyttää olevan erittäin voimakas, koska ihmiset käyttävät peliharrastuksiinsa erittäin paljon aikaa ja rahaa. Pelien suunnittelijan näkökulmasta on erittäin olennainen kysymys, millaiset ominaisuudet pelissä saavat ihmisen motivoitumisen peliä kohtaan käynnistymään ja jatkumaan.

Tom Chatfield ja Seth Priebatsch ovat toisistaan erillään pyrkineet tunnistamaan tiettyjä perusmalleja tai pelidynamiikkoja, jotka ovat omiaan motivoimaan pelaajia pelaamaan [Cha10, Pri10]. Esimerkki molempien tunnistamasta perusmallista on ”edistymisdynamiikka”, joka sisältää päämäärän ja edistymisen selkeän kommunikoinnin pelaajalle. Edistyminen on pelaajalle palkitsevaa, ja päämäärän tunteminen motivoi pelaajaa pyrkimään loppuun asti. Hyvä esimerkki tästä dynamiikasta on LinkedIniin lanseerattu profiilin valmiusastemittari, joka motivoi käyttäjiä täydentämään profiileihinsa puuttuvat tiedot. Vastaavanlainen dynamiikka on Tom Chatfieldin tunnistama periaate ”useat lyhyen ja pitkän tähtäimen tavoitteet”. Liian kaukana oleva tavoite voi tuntua saavuttamattomalta ja saattaa aiheuttaa kiinnostuksen lopahtamisen. Lyhyemmän tähtäimen tavoitteilla saadaan pelaajalle sopivasti välihaasteita ja -palkintoja, kun taas pitkän tähtäimen tavoite jaksaa ylläpitää motivaatiota pitkään.

Myös menestyneistä peleistä on jälkikäteen pyritty tunnistamaan ominaisuuksia, jotka siivittivät pelin menestykseen. Esimerkiksi Mike Rose on analysoinut suomalaisen suurmenestyksen Supercellin Clash of Clansin menestystekijöitä keskustelemalla yhdessä Clash of Clansin tuotejohtajan Lasse Louhennon kanssa [Ros13]. Analyysista tuli lopputuloksena viisi kohtaa, joista esimerkiksi pelin ominaisuuksien avaaminen pelaajan oppiessa kuulostaa hyvältä yleispätevältä neuvolta silloin, kun ominaisuuksia on pelissä liikaa uudelle pelaajalle. Liiat ominaisuudet voivat hämmentää uutta pelaajaa ja saada tämän luovuttamaan leikin sikseen.

Useat historialliset esimerkit osoittavat, että pelin ei tarvitse olla menestyäkseen erityisen monimutkainen tai erityisen näyttävä. Esimerkiksi Rovio Entertainmentin maailmankuulu Angry Birds on verrattain yksinkertainen peli, ja sitä on Rovion mukaan ladattu jo yli miljardi kertaa. Tällaiset esimerkit osoittavat, että keskittyminen olennaiseen on pelien kehityksessä erittäin kannattavaa. Nealen, Saltsman ja Boxerman ovat

kuvanneet minimalistisia pelejä, joiden tekeminen on helpompaa ja joiden tekemisessä päästään keskittymään enemmän ehkä siihen olennaisimpaan, pelin ydinmekaniikkaan [NSB11].

2.2 Tietokonepelit

Tässä tutkielmassa tietokonepelillä tarkoitetaan mitä tahansa peliä, jossa hyödynnetään tietokoneen suoritinta vuorovaikutteisen pelitilanteen luomiseksi. Tietokoneen suoritin voi tässä yhteydessä olla esimerkiksi perinteisessä PC:ssä, pelikonsolissa, tabletissa tai älypuhelimessa. Laite- ja suoritusympäristöjen yhdistelmiä kutsutaan pelialalla alustoiksi. Esimerkiksi Windows PC:t, Linux PC:t, pelikonsolit, iOS-mobiililaitteet ja Android-mobiililaitteet ovat kaikki pelien näkökulmasta eri alustoja. Tietokonepelit pyritään yhä useammin tuottamaan yhteensopivaksi useiden alustojen kanssa [Neo11].

Termiä tietokonepeli käytetään alan kirjallisuudessa melko väljästi. Joissain lähteissä termillä tarkoitetaan vain perinteisellä PC-tietokoneella pelattavia pelejä, kun toisissa lähteissä mukaan luetaan myös uudemmilla alustoilla, kuten tableteilla ja kännyköillä, ajettavat pelit. Tietokonepelin synonyyminä käytetään myös termiä videopeli. Videopeliltä kuitenkin useimmiten nimensä mukaisesti edellytetään sitä, että pelitapahtumat esitetään pelaajalle visuaalisesti jonkin näyttölaitteen kautta. Tarkasti ottaen tietokonepelin ei aina tarvitse olla videopeli.

Lähes aina tietokonepelit kuitenkin ovat videopelejä, ja pelitapahtumien visualisointi toteutetaan esittämällä pelitapahtumat kuvaa tuottavan näyttölaitteen kautta ja toistamalla pelitapahtumiin liittyvät äänet kaiuttimien kautta. PC:lla ja pelikonsoleilla näyttölaitteena toimii useimmiten vähintään 13 tuuman LCD-monitori tai televisio. Äänentoistojärjestelmä voi olla lähes millainen tahansa, mutta useimmiten käytettävä järjestelmä pystyy toistamaan laadukasta ääntä joko stereona tai 3-ulotteisesti. Pelaamiseen käytettävissä mobiililaitteissa on puolestaan tyypillisesti vähintään kolmen tuuman näyttö, ja ääni toistetaan joko laitteen kaiuttimesta heikkolaatuisena monoäänenä tai korvakuulokkeiden kautta stereona.

Pelaaja voi vaikuttaa pelitilanteeseen käyttämällä yhtä tai useampaa peliohjainta. PC-peleissä tyypillisesti käytettyjä peliohjaimia ovat ohjainsauva, hiiri ja näppäimistö. Konsolipeleissä käytetään useimmiten pelikonsolin mukana tulevia erillisiä peliohjaimia. Mobiilipeleissä ohjaimina käytetään tyypillisesti kosketusnäyttöä tai laitteen

näppäimistöä. Usein mobiilipelit hyödyntävät ohjaamisessa myös mobiililaitteeseen integroituja sensoreita, joiden avulla pelimaailmassa liikkumista voidaan helpottaa tai lisätä peliin muita, esimerkiksi paikkatietoon perustuvia, ominaisuuksia [Ben11].

Peliohjaimia on mainittujen perusesimerkkien lisäksi hyvin monenlaisia. Esimerkiksi ajopeleissä käytetään usein auton rattia ja polkimia jäljitteleviä ohjaimia, tanssipeleissä käytetään ohjaimina tanssimattoja, Xbox 360:n Kinect-peliohjain tunnistaa pelaajan kehon liikkeet kolmiulotteisesti kameroiden avulla, Singstarissa lauletaan mikrofoniin ja Rocksmithiä pelataan soittamalla oikeaa kitaraa, joka on liitetty tietokoneeseen sovittimella. Kuvassa 2 on esitetty Xbox 360:n Kinect-ohjain ja perinteisempi pelikonsolin mukana tuleva peliohjain.

Kuva 2: Vasemmalla vuonna 2010 julkaistu Xbox 360:n Kinect-peliohjain. Oikealla Nintendon pelikonsolin ohjain 1990-luvun alusta. Kuvat: Wikipedia.

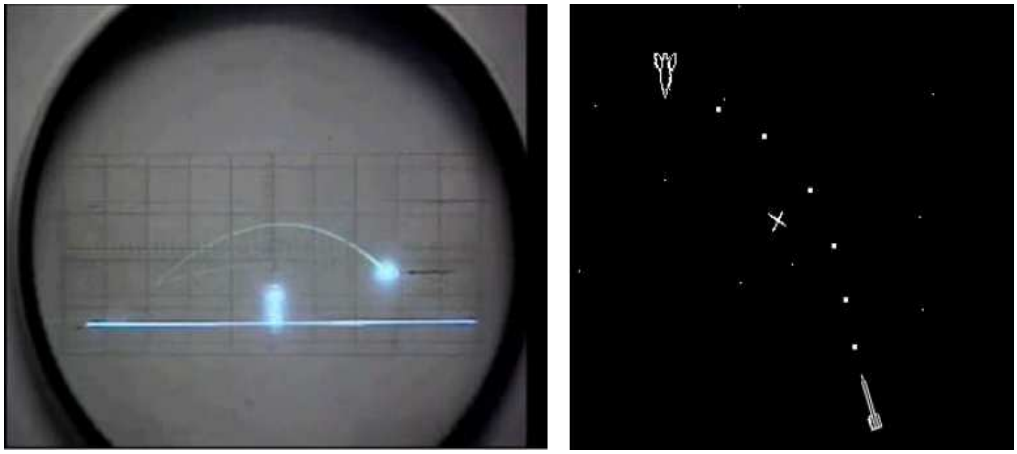
Joissain peliohjaimissa on vasteominaisuuksia, joiden avulla pelaajalle voidaan välittää palautetta pelitapahtumista. Esimerkiksi rattiohjaimet ja ohjainsauvat voivat vastustaa ohjausta kovassa nopeudessa tai pyrkiä pelitilanteen mukaan vääntämään siten kuin vastaavan laitteen oikea ohjain vääntäisi. Konsolien peliohjaimet voivat täristä tai sytyttää varoitusvaloja, kun pelaaja saa iskuja tai osumia luodeista. Tuntoaistiin perustuvalla vasteella peliin saadaan yhä mukaansatempaavampi ohjaustuntuma [SHK04].

Peliohjaimien saralla kehitys on ollut viime vuosina nopeaa. Kameroita ja kiihtyvyysantureita hyödyntävät peliohjaimet ovat vasta muutamia vuosia vanha ilmiö, ja ne mahdollistavat täysin uudenlaisen rajapinnan pelin ja pelaajan välillä. Aktiivisia tutkimuskohteita tällä alueella ovat muun muassa eleisiin perustuvat rajapinnat [NRL12] sekä niin sanotusti ajatuksen voimalla toimivat ohjauslaitteet [JKS10].

Tietokonepelien ja pelimarkkinoiden kehitys

Ensimmäiset tietokonepelit julkaistiin vuoden 1950 tienoilla. Klassisia esimerkkejä ensimmäisistä tietokonepeleistä ovat William Higinbothamin 1958 julkaisema Tennis for Two ja 1961 kolmen MIT:n opiskelijan julkaisema Spacewar!.

Ensimmäiset pelit olivat ulkoasultaan hyvin yksinkertaisia, eivätkä ne kyenneet simuloimaan aihettaan erityisen näyttävästi. Lisäksi pelaamiseen käytettävät laitteet olivat kalliita. Kuva 3 havainnollistaa ensimmäisten pelien visuaalista ulkoasua. Tietokonepelien pelaaminen oli marginaalista viihdettä 1970-luvun loppupuoliskolle asti [Vid12].

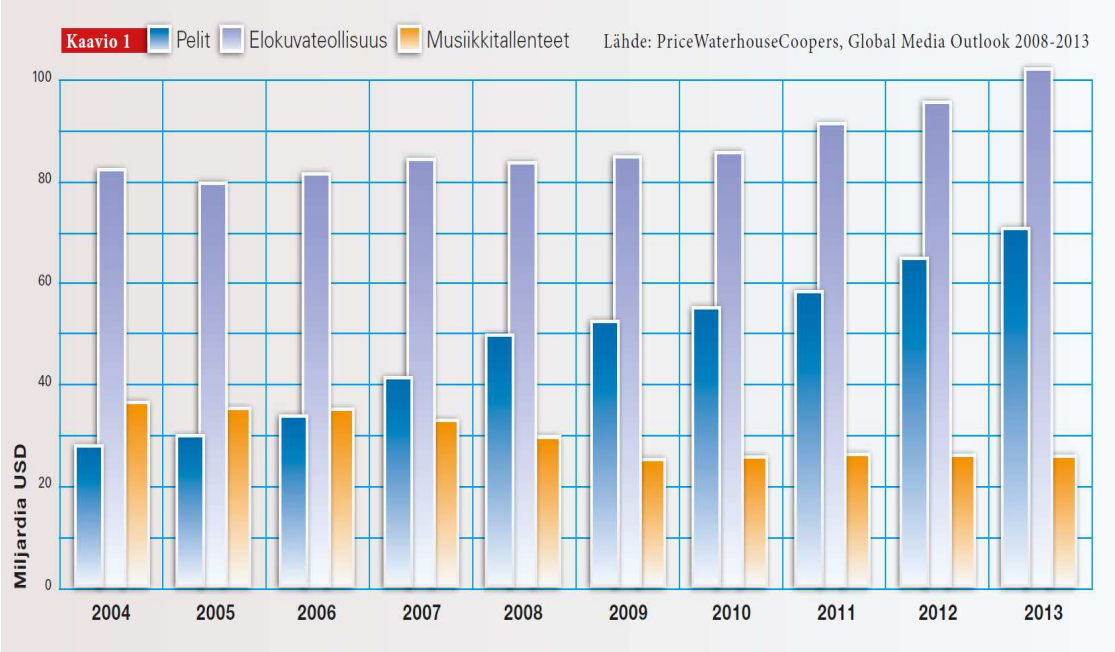


Kuva 3: Vasemmalla valokuva 1958 julkaistusta, oskilloskoopin näytöllä pelattavasta, Tennis for Twosta. Oikealla vuonna 1961 julkaistu Spacewar!. Kuvat: Wikipedia.

Pelimarkkinat lähtivät merkittävästi kasvamaan 1970-luvun lopussa, jolloin tietokoneet ja pelikonsolit alkoivat yleistyä kotikäytössä. Ensimmäisinä vuosikymmeninä tietokonepelit olivat vielä melko yksinkertaisia, ja siten yksittäisen pelin tuotantokustannukset olivat pienet. Tämä mahdollisti täysin uudenlaisten pelien tuottamisen kohtalaisen pienellä riskillä, ja 1980-luvulla julkaistiinkin useita innovatiivisia pelejä, jotka edustivat ennennäkemättömiä tyylilajeja. Seuraavina vuosikymmeninä pelien tuottaminen muuttui yhä vaativammaksi, ja yksittäisen peliprojektin tyypillinen koko lähti kasvamaan [Gre09, sivu 698].

Nykypäivänä tietokonepelit ovat miljardiluokan bisnes ja alan kokonaisliikevaihto on ohittanut muun muassa musiikkitallenteiden myynnin. PriceWaterHouseCoopers on

vuonna 2009 julkaisemassaan raportissa Global Media Outlook 2008-2013 arvioinut alan toteutuneita ja tulevia liikevaihtoja. Arviot on havainnollistettu kaaviomuodossa kuvassa 4, joka on poimittu Neogamesin julkaisemasta esityksestä Suomen pelitoimialan strategia 2010-2015 [Neo10].



Kuva 4: PriceWaterHouseCoopersin arvio ja ennuste tietokonepelialan vuosittaisesta liikevaihdosta. Lähde: [Neo10].

Ennuste näyttää olevassa melko hyvin linjassa DFC Intelligencen tuoreempien ennusteiden kanssa, joiden John Gaudiosi [Gau12] raportoi heinäkuussa 2012 ennustavan tietokonepelialan maailmanlaajuiseksi liikevaihdoksi vuodelle 2012 noin 67 miljardia dollaria. Samaisen lähteen mukaan DFC Intelligencen ennuste pelialan liikevaihdoksi vuodelle 2017 on 82 miljardia dollaria. Edellä kuvatut luvut pitävät sisällään kaikentyyppiset tietokonepelit perinteisistä arcade-peleistä mobiililaitteilla pelattaviin peleihin.

Tietokonepelien tekninen kehitys on ollut nopeaa. 1990-luvulla pelien visuaalinen ulkoasu alkoi voimakkaasti parantua, kun 3d-grafiikkakiihdyttimet integroitiin osaksi pelialustoja [Bau98]. Lisäksi tietoverkkojen kehittyessä peleihin alkoi 1990-luvulla tulla yhä enemmän verkkopeliominaisuuksia, jotka mahdollistavat usean pelaajan osallistumisen samaan peliin esimerkiksi internetin välityksellä.

Hienoimmat pelit ovat 2000-luvulla varsin näyttäviä simulaatioita aiheestaan, ja pelejä tuotetaan erittäin ammattimaisesti jopa satojen miljoonien eurojen budjeteilla. Pelien ulkoasun kehitystä pelien alkuajoista vuoteen 2006 havainnollistava video löytyy esimerkiksi David Perryn TED-puheesta [Per06]. Kuvassa 5 on kuvaruutukaappaus vuonna 2010 Playstation 3 pelikonsolille julkaistusta Gran Turismo 5:sta, joka toimii esimerkkinä modernin pelin visuaalisesta ulkoasusta.

Kuva 5: Kuvaruutukaappaus 2010 julkaistusta Gran Turismo 5:sta.

Useisiin pelien tyylilajeihin on viime vuosina ilmestynyt massiivisen moninpelaamisen mahdollistavia pelejä (Massively Multiplayer Online, MMO-pelit). MMO-peleissä pelimaailma on pysyvä, ja tapahtumat pelimaailmassa etenevät jatkuvasti. Pelaajia voi tietoverkon välityksellä liittyä peliin ja poistua pelistä kesken tapahtumien kulun. MMO-peliin voi osallistua satoja tai jopa tuhansia pelaajia samanaikaisesti, joten MMO-pelit mahdollistavat uudenlaisen sosiaalisen kanssakäymisen tietokonepelien pelaamisen yhteydessä [Sch11].

2.3 Mobiilipelit

Mobiilipeleillä tarkoitetaan tässä tutkielmassa matkapuhelimilla, tableteilla, mp3-soittimilla ja muilla päivittäin mukana kulkevilla kompakteilla alustoilla toissijaisena käyttötarkoituksena ajettavia tietokonepelejä. Mobiilipeleiksi ei lueta yksinomaan kannettaville pelilaitteille, kuten Nintendon Gameboy Advancelle, suunnattuja pelejä. Termiä käytetään alan kirjallisuudessa usein juuri tässä merkityksessä.

Mobiilipelien kehitys muistuttaa paljon tietokonepelien kehityskulkua, mutta se on tapahtunut huomattavasti nopeammin. Ensimmäiset mobiilipelit ilmestyivät 1990-luvun puolivälin tienoilla matkapuhelimille. Matkapuhelimet olivat 1990-luvulla laskentateholtaan vaatimattomia, ja vastaavasti ensimmäiset mobiilipelit olivat yksinkertaisia eivätkä erityisen näyttäviä. Matkapuhelimen käyttö pelialustana kuitenkin mahdollisti pelaamisen arjen tilanteissa, joissa perinteisten tietokonepelien pelaaminen oli mahdotonta.

Mobiililaitteet ovat sittemmin yleistyneet ja kehittyneet nopeasti. Matkapuhelimet ovat nykyään kotitalouksissa yleisempiä kuin perinteiset tietokoneet, ja matkapuhelinten määrästä noin kahtakymmentä prosenttia edustava parhaimmisto, älypuhelimet, sopii erityisen hyvin verkkopalveluiden ja pelien käyttämiseen [PKS09, Ans12]. Matkapuhelinten rinnalle on syntynyt kokonaan uusia mobiililaitealustoja kuten helposti mukana kulkevat tablet-tietokoneet ja mp3-soittimet. Mobiililaitteiden kärkeä laskentatehossa edustavat vuoden 2013 alussa 3d-grafiikkakiihdyttimillä ja moniydinprosessoreilla varustetut tabletit ja älypuhelimet, joissa on korkean pikselitiheyden kosketusnäytöt ja muistikapasiteettia kymmeniä gigabittejä [Low13].

Mobiililaitteiden laskentatehon kasvaessa mobiilipelit ja mobiilipelien markkinat ovat kehittyneet hätkähdyttävän nopeasti. Kärkikastin mobiilipelit ovat visuaalisesti erittäin näyttäviä huomioiden pienten laitteiden asettamat rajoitukset. Kuvassa 6 on älypuhelimessa suoritettava moderni mobiilipeli.

Mobiililaitteiden erityisiä vahvuuksia pelialustoina ovat muun muassa laitteiden aktiivinen käyttöprofiili, suurten alustatoimittajien digitaaliset jakelukanavat, massiiviset käyttäjämäärät, laitteiden käyttöön liittyvä sosiaalinen verkottuneisuus ja laitteisiin integroitu sensoriteknologia [Neo11, PKS09]. Erityinen haaste on pienen mobiililaitteen käyttö samanaikaisesti sekä audiovisuaalisen vasteen antamiseen että peliohjaimena,

mikä voi helposti johtaa ongelmiin käyttöliittymässä. Älypuhelimissa on usein pieni kosketusnäyttö, jota joudutaan lähes poikkeuksetta peleissä käyttämään sekä pelitapahtumien esittämiseen että peliohjaimena.

Kuva 6: Madfinger Gamesin 2012 julkaisema Dead Trigger suorituksessa älypuhelimessa. Huomaa kosketusnäytöllä olevat käyttöliittymäelementit.

Mobiilipelit ovat tuoneet tietokonepelien pelaamisen osaksi yhä suuremman kansan osan arkea, eikä tietokonepelin pelaaminen enää välttämättä edellytä istumista kotona tietokoneen tai pelikonsolin äärellä [Neo11]. Mobiilipelit nähdäänkin erittäin merkittävänä tulevaisuuden kasvualana koko tietokonepelialalle [Bar12].

2.4 Tietokonepelien tyylilajit

Tietokonepelejä on lukematon määrä, ja pelien luonne vaihtelee laidasta laitaan. Joissain peleissä pelaaja ohjaa tosiajassa etenevää hahmoa taistelukentällä toisten pelien ollessa vuoropohjaista strategiaa tai vaikkapa nettipokeria. Pelejä pyritään jakamaan tyylilajeihin pelitapahtuman vuorovaikutuksen luonteen mukaan [App06]. Pelien jaottelussa on siis lähtökohtana vuorovaikuttamisen tapa pelaajan ja pelin välillä, eivätkä erot pelin juonessa tai pelimaailman sisällössä. Tietyn tyylilajin sisällä pelejä voidaan jakaa edelleen tarkemmin alaluokkiin.

Tietokonepelien tyylilajeille ei ole tarkkoja yleisesti hyväksyttyjä määritelmiä, ja pelin sijoittaminen yhteen tai useampaan tyylilajiin on enemmän tai vähemmän mielipideky-

symys. Joitain pelejä on erityisen hankala sijoittaa yksittäiseen tyylilajiin pelintekijöiden sisällyttäessä peliin ominaisuuksia useammasta tyylilajista. Seuraavissa kappaleissa esitellään lyhyesti muutama hyvin tunnettu tyylilaji.

Toimintapelit muodostavat ehkä laajimman tietokonepelien tyylilajin. Toimintapelien alaluokkia ovat esimerkiksi tosiajassa etenevät taistelupelit, kamppailupelit, flipperit ja tasohyppelypelit. Toimintapeleissä peli etenee olennaisilta osin reaaliajassa, ja menestyminen pelissä edellyttää pelaajalta nopeita refleksejä, hyvää toiminnan ajoitusta ja tarkkuutta.

Kaupallisesti ehkä merkittävin toimintapelien alaluokka on ensimmäisen persoonan ammuntapelit (First Person Shooter, FPS), jonka Id:n julkaisema Doom popularisoi vuonna 1993. FPS-peleissä pelaaja näkee maailman ohjaamansa pelihahmon silmistä kuvattuna. Yleensä ruudun alareunassa näkyy hahmon kädessä oleva ase. Pelaaja ohjaa hahmon toimia pelimaailmassa ja pyrkii tuhoamaan vihollisia ampumalla. Uudempia FPS:ia edustaa esimerkiksi Call of Duty -sotapelisarja, josta kuvassa 7 on kuvaruutukaappaus.

Kuva 7: Kuvaruutukaappaus Call of Duty: Black Ops II.

Seikkailupelit muodostavat toisen suositun tyylilajin, ja tyylilaji oli edustettuna jo ensimmäisten tietokonepelien joukossa. Seikkailupelit voivat olla esimerkiksi tekstipohjaisia tai piirroselokuvan tyyliin vaihtelevista kuvakulmista esitettyjä. Seikkailupeleissä pelaajan haasteet eivät liity reflekseihin ja ohjaustarkkuuteen, vaan pelaajalle tulee pelin edetessä usein väkivallattomia ja kiireettömiä pulmia ratkaistavakseen. Pulmien ratkaisu edellyttää pelaajalta monimutkaisia toimia, jotka kohdistuvat pelimaailman esineisiin ja toisiin maailman pelihahmoihin. Kuvassa 8 on kuvaruutukaappaus Lucas Artsin 1990 julkaisemasta seikkailupelistä The Secret of Monkey Island.

Kuva 8: Kuvaruutukaappaus The Secret of Monkey Islandista.

Roolipelit muodostavat myös yhden vanhimmista ja laajimmista tyylilajeista. Roolipelit jakautuvat useisiin alaluokkiin, joita ovat esimerkiksi toimintaroolipelit, massiiviset online-roolipelit (Massive Multiplayer Online Roleplaying Game, MMORPG) ja taktiset roolipelit.

Roolipeleissä pelaaja ohjaa yhtä tai useampaa pelihahmoa. Roolipelien pelihahmot edustavat yleensä jotain lahkoa kuten soturit, varkaat tai velhot. Pelihahmot saavat pelin edetessä koetuista asioista kokemuspisteitä, jotka mahdollistavat uusien taitojen oppimisen ja vaikuttavat suoriutumiseen tulevissa taistelutilanteissa ja muissa tehtävissä.

Roolipeleissä on usein suurehko maailma, jossa pelaajan ohjaama hahmo tai ryhmä voi seikkailla, joutua mukaan taisteluihin ja kerätä mukaansa hyödyllisiä tavaroita. Roolipeleissä peli voi edetä reaaliajassa tai vuoropohjaisesti. Esimerkki modernista MMO-roolipelistä on Blizzard Entertainmentin jo vuonna 2004 julkaisema World of Warcraft, josta on kuvaruutukaappaus kuvassa 9.

Kuva 9: Kuvaruutukaappaus World of Warcraftista.

Mainittujen lisäksi pelien tyylilajeja ovat esimerkiksi simulaatiopelit, strategiapelit, musiikkipelit, rahapelit ja urheilupelit.

Mobiilipelejä kehittävän yrityksen näkökulmasta erityisen huomionarvoisen pelien osajoukon muodostavat niin sanotut rennot pelit (casual games). Rennot pelit voivat kuulua mihin tahansa tyylilajiin, mutta yhteistä niille on helppo pelattavuus ja nopea oppimiskäyrä. Rennot pelit eivät vaadi pelaajalta suurta panostusta peliin eivätkä mitään erityisiä pelitaitoja. Rennot pelit voivatkin sopia erityisen hyvin mobiilipeleiksi ja pelattavaksi hetkittäin keskellä jokapäiväisiä kiireitä [Kui07].

3 Peliohjelmistot

Peliohjelmistot ovat tietokoneohjelmia, joiden avulla tietokonepelit tuotetaan. Jaeltava peliohjelma on peliohjelmiston olennaisin osa, mutta sen lisäksi peliohjelmistoon kuuluvat olennaisena osana kehitystyökalut. Tässä luvussa tutustutaan peliohjelmistojen toiminnallisuuteen ja rakenteeseen.

3.1 Peliohjelmistojen perustoiminnallisuus

Tietokonepelit ovat hyvin vaihtelevan tyyppisiä. Pelien ja pelialustojen suuret keskinäiset erot tarkoittavat vääjäämättä sitä, että peliohjelmistojenkin välillä on suuria eroja. Keskinäisestä erilaisuudestaan huolimatta kaikki tietokonepelit simuloivat vuorovaikutteista pelitapahtumaa, joten niissä tarvitaan paljon samankaltaista toiminnallisuutta. Kaikissa tietokonepeleissä on oltava muun muassa käyttöliittymä, jonka avulla pelaaja saa tiedon, mitä pelissä tapahtuu, ja jonka kautta pelaaja voi vaikuttaa pelitapahtumiin. Peliohjelmistojen täytyy siksi aina käsitellä pelaajalta tulevia ohjauskomentoja, ja käyttöliittymän audiovisuaaliset komponentit täytyy käytännössä aina ladata tiedostoista, jotka on tuotettu erillisillä digitaalisen sisällön tuottamiseen tarkoitetuilla työvälineillä [Gre09, sivut 273-276].

Peliohjelmiston osat voidaan jakaa kahteen ryhmään: pelin suoritusaikaiset osat ja kehitystyökalut, joita käytetään pelin sisällön tuottamiseen. Kehitystyökalut eivät ole osa varsinaista loppukäyttäjälle toimitettavaa peliä. Esimerkkejä kehitystyökaluista ovat pelin maailman muokkaustyökalut ja työkalut, joilla peliin tulevaa sisältöä muutetaan erillisten digitaalisen sisällöntuotantotyökalujen tuottamista tiedostomuodoista pelin komponenttien kanssa yhteensopiviin tiedostomuotoihin.

Kun useimmissa tietokonepeleissä tarvittavia toiminnallisuuksia tarkastellaan lähemmin, saadaan hyvä käsitys siitä, mitä kaikkea peliohjelmistot tyypillisesti pitävät sisällään. Seuraavissa aliluvuissa käsitellään lyhyesti peliohjelmistojen kehittymistä ja kuvataan peliohjelmistoista tyypillisesti löytyviä osakokonaisuuksia. Lopuksi esitellään pelimoottorit.

3.2 Katsaus peliohjelmistojen rakenteen kehittymiseen

Ensimmäiset peliohjelmistot olivat tarkasti vain yksittäisen pelin tuottamiseen räätälöityjä yksinkertaisia ohjelmia. Ensimmäisille tietokonepeleille oli tyypillistä, että ne kehitettiin vain yhden tai muutaman henkilön voimin ilman erityisiä pelinkehitystyökaluja, ja ohjelmakoodin uudelleenkäyttö oli minimaalista [Gre09, sivu 698].

Pelialan kasvaessa ja tietokoneiden suorituskyvyn kehittyessä tietokonepelit alkoivat kehittyä yhä näyttävämmiksi ja monipuolisemmiksi. Tämä vaati, että peliohjelmiin toteutettiin yhä hienostuneempaa toiminnallisuutta. Muun muassa 3-ulotteinen grafiikka ja verkkopeliominaisuudet alkoivat yleistyä useissa pelien tyylilajeissa 1990-luvulla. Myös fysiikan simulointia alettiin 1990-luvulla yhä useammin soveltaa pelimaailman tapahtumien simuloinnissa. 2000-luvun puolella on kehitetty massiivia moninpelejä (MMO), joihin sadat tai jopa tuhannet pelaajat voivat osallistua samanaikaisesti. Peliprojektien tiimien keskimääräinen koko ja sisällön määrä ovat pelien kehittymisen myötä kasvaneet voimakkaasti, joten pelien kehittämisessä käytettäviin työkaluihin on kohdistunut kehitystarpeita.

Peliohjelmistot ovat näin muuttuneet entistä monimutkaisemmiksi, ja niiden koko on kasvanut. Peliohjelmistojen monimutkaistuminen on vaatinut alati kasvavia investointeja peliprojekteihin, ja peliohjelmistojen osien uudelleenkäyttöön on alettu kiinnittää peliteollisuudessa entistä enemmän huomiota [Gre09, sivu 11]. Uudelleenkäytöstä saatavien hyötyjen kasvaessa uudelleenkäyttö alkoi pelituotannossa lisääntyä, ja peliohjelmistojen toiminnalliset osakokonaisuudet alkoivat erottua toisistaan.

Tietyt toiminnalliset rakenneosat, komponentit, ovat sittemmin vakiinnuttaneet paikkansa osana peliohjelmistoja. Tänä päivänä tyypillisesti peliohjelmistoista löytyviä komponentteja ovat muun muassa piirtomoottori, törmäystunnistus- ja fysiikkamoottori, verkkokomponentti, skriptirajapinta ja -tulkki, muistinhallintakomponentti, äänimoottori sekä kehitystyökalut pelimaailman määrittelyä ja digitaalisen sisällön kanssa työskentelyä varten [Gre09, sivu 3]. Tärkeimpiä komponentteja käsitellään tarkemmin seuraavassa aliluvussa.

Isoilla pelitaloilla on komponenteista usein omia toteutuksiaan, joita käytetään peliprojektista toiseen ja kehitetään samalla eteenpäin. Komponentteja on tullut vuosien saatossa myös avoimille markkinoille erilaisin lisenssein. Valmiiden komponenttien

käyttö on omiaan parantamaan ohjelmiston laatua ja vähentämään pelin perustoiminnallisuuden toteuttamiseen kuluvaa työmäärää [Oin06]. Markkinoilla saatavilla olevia valmiita komponentteja ovat esimerkiksi äänikirjastot kuten Audiere, piirtomoottorit kuten Ogre, verkkopelikirjastot kuten RakNet ja fysiikkamoottorit kuten Open Dynamics Engine.

Kun peliprojektissa hyödynnetään valmiita komponentteja, täytyy projektissa integroida valitut komponentit yhteen siten, että saadaan aikaiseksi saumattomasti toimiva kokonaisuus, jonka avulla pelikohtainen toiminnallisuus on kätevä toteuttaa. Koska nykyaikaisen pelin tuottamisessa tarvitaan tyypillisesti paljon toiminnallisuutta, tarvitsee peliohjelmisto tyypillisesti koostaa useista komponenteista. Tällöin laadukkaasti toimivan ja ylläpidettävän kokonaisuuden muodostaminen vaatii huolellista kokonaisarkkitehtuurin suunnittelua ja toteutusta.

Kokonaisarkkitehtuuriin liittyviä haasteita ovat muun muassa komponenttien väliset integraatiot, pelikohtaisen toiminnallisuuden käyttämä rajapinta, valittujen komponenttien vaihteleva alustatuki, komponenttikohtaisesti vaihteleva ohjelmiston elinkaaren hallinta sekä pelin sisällön kehitysprosessi valituilla työvälineillä. Erityisesti sisällön kehittämisen sujuvuuteen voi olla tarpeen kiinnittää huomiota, koska pelien sisällön määrä on kasvanut voimakkaasti [Gre09, sivu 698].

3.3 Peliohjelmistojen tärkeimmät rakenneosat

Pelin pääsilmukka on jaeltavan peliohjelman keskeinen rakenne, joka yhdistää pelin ajonaikaiset komponentit toisiinsa. Pelin pääsilmukka on osa pelin ajonaikaista pääohjelmaa, ja sen tehtävänä on ohjata kaikkien pelin osajärjestelmien yhteistoimintaa. Yksi pääsilmukan iteraatiokierros vie pelitilanteen simulaatiota aina pienen aika-askeleen eteenpäin. Pääsilmukasta kutsutaan käytännössä kaikkien pelin osakomponenttien päivitysaskelfunktioita sopivassa järjestyksessä, jonka lisäksi pääsilmukka käynnistää pelikohtaisen toiminnallisuuden suorituksen tarvittavissa väleissä [Gre09, sivu 304].

Pelin pääsilmukan ja ylipäänsä peliohjelmien rakenteeseen on merkittävästi vaikuttanut tietokoneiden suoritinytimien määrän kasvu. Nykypäivänä pelin täytyy kyetä jakamaan pelin laskentakuormaa useille ytimille voidakseen hyödyntää käytettävissä olevaa laskentatehoa tehokkaasti. Kuormaa voidaan jakaa esimerkiksi ajamalla pelin osakomponentteja omissa säikeissään. Ongelmana tässä lähestymistavassa voi olla huo-

mattava epäsuhta eri osakomponenttien laskentatehon tarpeessa, jolloin yhteen säikeeseen voi helposti päätyä suuri osa koko pelin laskennasta. Toinen vaihtoehto on jonkinlaisen työnjakomekanismin toteuttaminen. Tällöin jokaisella iteraatiokierroksella tehtävästä laskennasta muodostetaan sopivan kokoisia tehtäviä, joita voidaan välittää esimerkiksi pysyvien työsäikeiden suoritettavaksi [Gre09, sivut 325-333].

Piirtomoottori

Komponenttia, joka huolehtii pelitapahtumien visualisoinnista näyttölaitteelle, kutsutaan piirtomoottoriksi. Poikkeustapauksia lukuun ottamatta kaikkien tietokonepelien täytyy visualisoida pelitapahtumat näyttölaitteelle, joten jonkinlainen piirtomoottori löytyy lähes kaikista peleistä.

Nykyisin peleissä hyödynnetään usein 3-ulotteista grafiikkaa, jota tuotetaan sujuvan vaikutelman aikaansaamiseksi vähintään noin 30 kuvaa sekunnissa. Reaaliaikainen 3-ulotteisen kuvan piirtäminen on erittäin laaja alue, joten tässä pääsemme raapaisemaan vain hieman aihepiirin pintaa.

Kolmiulotteisen kuvan piirtämisen ydinprosessi on seuraavanlainen [Gre09, sivu 400]:

- Määritellään virtuaalinen näkymä kolmiulotteisina pintoina
- Määritellään näkymässä olevien pintojen visuaaliset ominaisuudet. Ominaisuudet määritellään käytännössä määrittelemällä pinnoille materiaalit. Materiaalien ominaisuuksia ovat esimerkiksi pinnan piirrossa käytettävät tekstuurit, heijastusominaisuudet ja käytettävä sävytinalgoritmi [Mit07].
- Sijoitetaan ja suunnataan virtuaalinen kamera halutulla tavalla. Tyypillisesti kamera määritellään yksittäisenä ideaalipisteenä, jonka edessä kameran virtuaalinen valosensori leijailee. Kameran virtuaalisen valosensorin pisteet vastaavat visualisointiin käytettävän näyttölaiteen pikseleitä, ja kameran ideaalipiste vastaa katsojan paikkaa.
- Määritellään valonlähteet. Valonlähteistä tuleva valo reagoi näkymässä olevien esineiden kanssa ja heijastuu niistä kameraan.
- Jokaiselle pikselille virtuaalisessa valosensorissa lasketaan valonsäteiden väri ja voimakkuus, joka pikselin kautta kulkee ideaalipisteeseen. Tämä määrittää näytöllä vastaavalla kohdalla olevan pikselin värin.

Yllä kuvattu ydinprosessi kirjoittaa lasketut pikselien värit näyttöpuskuriin (frame buffer), jonka jälkeen valmiin näyttöpuskurin sisältö voidaan näyttää tietokoneen ruudulla. Joissain tapauksissa näyttöpuskurille tehdään jälkikäsittelyä, ennen sen näyttämistä ruudulla, tai käytettyä puskuria käytetäänkin jonkin pinnan tekstuurina esimerkiksi heijastusvaikutelman aikaansaamiseksi.

Piirtoprosessi on ollut tietokonepelien alkutaipaleella vakava pullonkaula, mutta se on tehostunut vuosien saatossa, kun piirtoon liittyvää laskentaa on siirretty yhä enemmän tietokoneen keskussuorittimelta näytönohjaimen grafiikkasuorittimen suoritettavaksi.

Myös piirtoon liittyvät algoritmit ovat kehittyneet. Näitä ovat esimerkiksi algoritmit kameran näkymän ulkopuolelle jäävien pintojen tehokkaaseen näkyvyyskarsintaan, varjojen laskemista reaaliajassa approksimoivat algoritmit ja varjokarttojen käyttö varjojen laskentaan etukäteen [Eis12, Mit07]. Piirtoprosessin tehostuminen on mahdollistanut yhä näyttävämpien tekniikoiden käytön reaaliaikaisessa piirrossa. Kaikkiin mainittuihin menetelmiin liittyvät omat hyvät ja huonot puolensa, ja menetelmiä pyritään kehittämään jatkuvasti.

Piirtomoottorin ja muun pelin välisen rajapinnan määrittelyssä olennaisia asioita ovat piirron tehokkuus sekä lisäominaisuudet, joita kuvan tuottamisen lisäksi tarvitaan. Nykyisin esimerkiksi tekstuuriin piirtoa käytetään peleissä usein heijastusten toteuttamiseen ja tuotetun kuvan jälkikäsittelyyn. Lisäksi peleissä käytetään paljon tapauskohtaisia sävytinohjelmia, joilla saadaan materiaalit näyttämään juuri halutunlaiselta. Peli saattaa myös tarvita tietoa, näkyykö jokin tietty peliolio tietyllä ajanhetkellä, ja tässä piirtomoottorin jo tekemää näkyvyyskarsintaa voidaan hyödyntää, mikäli pelin ja piirtomoottorin välinen rajapinta sen mahdollistaa.

Törmäystunnistus ja fysiikkamoottori

Peleissä, joissa pelimaailman oliot eivät ole paikallaan tai kulje vain ennalta määritettyjä ratoja pitkin, täytyy erikseen huolehtia siitä, että pelioliot eivät kulje toistensa läpi. Tähän tarvitaan toiminnallisuutta, joka huolehtii törmäysten tunnistuksesta. Törmäystunnistuskomponentti täytyy siksi aina tavalla tai toisella toteuttaa peleihin, joissa pelioliot liikkuvat ja vaikuttavat toisiinsa 2- tai 3-ulotteisessa avaruudessa.

Koska peliä simuloidaan pääsilmukan ohjauksella tietyn pituisina ajallisina harppauksina eteenpäin, on huomattava, että useilla menetelmillä törmäystä ei havaita välittömästi sen tapahtuessa, vaan kun toisiinsa törmäävät pelioliot ovat jo hieman toistensa kanssa limittäin. Törmäystunnistukseen liittyy epätriviaaleja ongelmia, kuten nopeasti liikkuvien pienten esineiden (mm. luodit) törmäykset. Tällaiset esineet läpäisevät helposti toisensa, ennen kuin törmäystä ylipäänsä tunnistetaan. Törmäystunnistuskomponentti voi tuottaa paljon tietoa törmäyksestä. Törmäyspisteiden lisäksi usein hyödyllistä tietoa ovat esimerkiksi törmäävien peliolioiden pintojen normaalit törmäyskohdassa, törmäyksen syvyys ja törmäyksen irrotusvektori.

Fysiikkamoottoriksi kutsutaan sitä pelin osaa, joka huolehtii peliolioiden liikkeen jatkuvuudesta ja peliolioiden välisten mekaanisten vuorovaikutusten simuloinnista. Peleissä käytettävät fysiikkamoottorit jäljittelevät tyypillisesti jäykkien kappaleiden mekaniikkaa, jonka lisäksi ne usein mahdollistavat erilaisten liitosten määrittelyn jäykkien kappaleiden välille. Monipuolisten fyysisten kokonaisuuksien mallintaminen on mahdollista yhdistelemällä jäykkiä kappaleita toisiinsa halutunlaisilla liitoksilla.

Fysiikkamoottori on läheisessä yhteydessä törmäystunnistukseen, koska kappaleiden sijainti, liike ja törmäykset ovat vahvasti toisiinsa kytköksissä. Lisäksi törmäystunnistuksesta saatavia tietoja tarvitaan fysiikkamoottorissa muun muassa törmäysten aiheuttamien impulssien laskennassa. Usein fysiikkamoottorit sisältävätkin myös törmäystunnistukseen liittyvän toiminnallisuuden.

Kaikki pelit eivät tarvitse varsinaista fysiikkamoottoria. Peleissä peliolioiden liikettä ja mahdollisia vuorovaikutuksia voidaan ohjata myös pelikohtaisten sääntöjen mukaan. Usein näin tehdäänkin, jos vuorovaikutuksia on pienehkö määrä ja ne ovat yksinkertaisia. Yleisluontoinen törmäystunnistus ja jäykkien kappaleiden mekaniikan simulointi ovat monimutkaisia ongelmia, joiden ratkaiseminen tyydyttävällä tavalla reaaliajassa

vaatii hienostunutta toiminnallisuutta. Ne pelit, joissa fysiikkamoottoria tarvitaan, hyödyntävät siksi useimmiten jotain pitkälle kehitettyä valmista moottoria kuten PhysX, ODE tai Havok. Valmiin komponentin hyödyntämisestä huolimatta fysiikkasimulaation tuonti mukaan pelimaailmaan on omiaan tuomaan uusia ennustamattomia tilanteita ja haasteita pelin kehitysprojektiin [Gre09, sivut 595-601].

Pelin, törmäystunnistuksen ja fysiikkamoottorin välisessä rajapinnassa on olennaista riittävä joustavuus ja läpinäkyvyys, joka mahdollistaa pelikohtaisen toiminnallisuuden vaivattoman toteuttamisen. On tyypillistä, että pelikohtainen toiminnallisuus tarvitsee monenlaisia tietoja tilanteista, joissa tietynlaiset pelioliot törmäävät keskenään. Tällöin pelikohtaisia sääntöjä voidaan soveltaa tarvittaessa törmäysten vaikutusten laskennassa. Lisäksi pelit usein tarvitsevat törmäystunnistuskomponentilta palveluita hypoteettisten törmäyskyselyiden suorittamiseksi. Hypoteettisella törmäyskyselyllä törmäystunnistuskomponentilta voidaan saada tietoa, mihin määrätystä pisteestä tiettyyn suuntaan lähetetty hypoteettinen kappale tai säde pelimaailmassa törmäisi. Esimerkiksi tekoäly voi tunnistaa seinän ohjaamansa hahmon ja lähellä olevan vihollisen välissä tekemällä törmäyskyselyn oman hahmonsa sijainnista kohti vihollista [Gre09, sivut 625-629].

Skriptirajapinta ja tekoäly

Useimpia pelejä voi pelata tietokonetta vastaan. Tällöin tietokoneen pelihahmojen toimia ohjaa pelin tekijöiden toteuttama tekoäly. Tekoälyn voi toteuttaa ajonaikaisen pelin pääohjelman yhtenä osana, mutta tällöin tekoälyn muuttaminen vaatii aina tekoälykomponentin uudelleenkääntämisen ja koko pääohjelman uudelleenlinkittämisen. Tällä tavoin tekoälyn kehittämisessä tapahtuva iterointi muodostuu tarpeettoman hitaaksi. Lisäksi pelin pääohjelman toteuttamisessa käytettävä ohjelmointikieli ei välttämättä sovellu kovin hyvin tekoälyn toteuttamiseen.

Edellä mainituista syistä tekoälyn toteuttamiseen on pyritty löytämään parempia keinoja. Nykyisin tekoäly toteutetaan usein ohjelmoimalla toiminnallisuus sopivalla tulkattavalla skriptikielellä. Tällöin tekoälyn muuttaminen onnistuu helposti muusta pelistä erillisiä skriptejä muuttamalla eikä iteroinnissa tarvitse kuin ladata muutetut skriptit uudelleen. Näin iterointi voi onnistua jopa kehitettävän pelin ollessa käynnissä. Peliohjelmistossa täytyy tällöin olla työkalut skriptien tulkkaamiseen, ja peliohjelmiston täytyy tarjota ajettaville skripteille skriptien tarvitsemat palvelut.

Myös muiden pelikohtaisten toiminnallisuuksien toteuttamisessa voidaan hyödyntää skriptejä. Esimerkiksi kaikki peliolioiden toiminnallisuudet ja pelin kulkua kontrolloiva logiikka voidaan toteuttaa skripteillä. Tällöin myös niitä on nopea muuttaa koskematta pelin pääohjelmaan. Esimerkiksi pelimoottori Unity toimiikin siten, että koko pelikohtainen ohjelmakoodi toteutetaan skripteillä.

Skripteille täytyy tarjota sopiva ohjelmointirajapinta, jonka kautta pelin olennaisten komponenttien palvelut ovat käytettävissä. Esimerkiksi tekoälyohjelmat tarvitsevat pääsyn ohjaamansa peliolion rajapinnan tarjoamiin palveluihin sekä usein muun muassa törmäystunnistuksen ja fyysisen simulaation palveluihin päättäessään toimistaan. Sopivan rajapinnan avulla tekoälyohjelma voi kätevästi selvittää esimerkiksi sen, mitä muita peliolioita tekoälyn ohjaaman pelihahmon lähellä on näkyvissä.

Verkkokomponentti

Yhä useammat pelit hyödyntävät verkkoa. Ilmeinen käyttökohde on moninpelimahdollisuuden toteuttaminen verkon välityksellä, mutta tämän lisäksi käyttötarkoituksia ovat esimerkiksi tulosten lähettäminen sosiaaliseen mediaan, ostosten tekeminen suoraan pelissä tai mainosten lataaminen verkosta pelaajan nähtäville. Peleissä on yleensä verkkoliikennettä varten erillinen komponentti, joka kätkee verkon kautta tapahtuvaan tietoliikenteeseen liittyvän monimutkaisuuden sisäänsä pelin muilta osilta.

Verkkopeli toteutetaan usein asiakas-palvelin-mallilla, jossa palvelin tarjoaa yhteisen pelipaikan usealle pelaajalle. Massiivisen moninpelin tapauksessa palvelimen skaalautuminen muodostuu kuitenkin helposti pullonkaulaksi. Massiivisen moninpelin mahdollistavissa roolipeleissä skaalautuvuusongelmat on yleensä ratkaistu jakamalla pelimaailma vyöhykkeisiin, joista jokaisesta vastaa yksi palvelin. Pelaajien välinen vuorovaikutus vyöhykkeiden välillä ei tällöin ole reaaliajassa mahdollista, vaan vuorovaikutusyhteys katkeaa toisen pelaajan siirtyessä eri vyöhykkeelle. Peleihin, joissa kaikkien pelaajien välillä on oltava jatkuva yhteys, tarvitaan toisenlaisia skaalautuvia arkkitehtuureita kuten Jens Müllerin ja Sergei Grolatchin esittelemä proxy-arkkitehtuuri [MG06].

Kehitystyökalut

Tietokonepeleissä sisältöä ovat esimerkiksi pelimaailmassa kuultavat äänet, kaikkien peliolioiden visuaaliset ulkoasut, peliolioihin liittyvät animaatiot ja itse pelimaailman tapahtumapaikan rakenne. Digitaalisen sisällön tuottamiseen on varsin kehittyneitä ja yleisesti käytettyjä sisällöntuotantotyökaluja. 3-ulotteisen sisällön, kuten mallien ja animaatioiden, tuottamiseen tunnettuja työkaluja ovat Maya, 3ds Max ja Blender. 2-ulotteista grafiikkaa voi tuottaa esimerkiksi Photoshopilla tai Gimpillä. Äänien käsittelyyn on vastaavasti omat erityistyökalunsa kuten Audacity.

Koska tällaisia työkaluja on valmiiksi saatavilla, peliohjelmistoon ei tarvitse toteuttaa digitaalisen sisällöntuotannon perustoiminnallisuutta. Jonkinlainen toiminnallisuus kuitenkin tarvitaan digitaalisen sisällön tuotantotyökaluilla tuotettujen ”irrallisen” sisällön osasten integroimiseksi peliin mielekkäiksi kokonaisuuksiksi. Kehitystyökaluja ovat ne peliohjelmiston osat, jotka eivät ole välttämättömiä itse suoritusaikaisessa pelissä mutta joita käytetään pelin teknisessä tuottamisprosessissa hyödyksi.

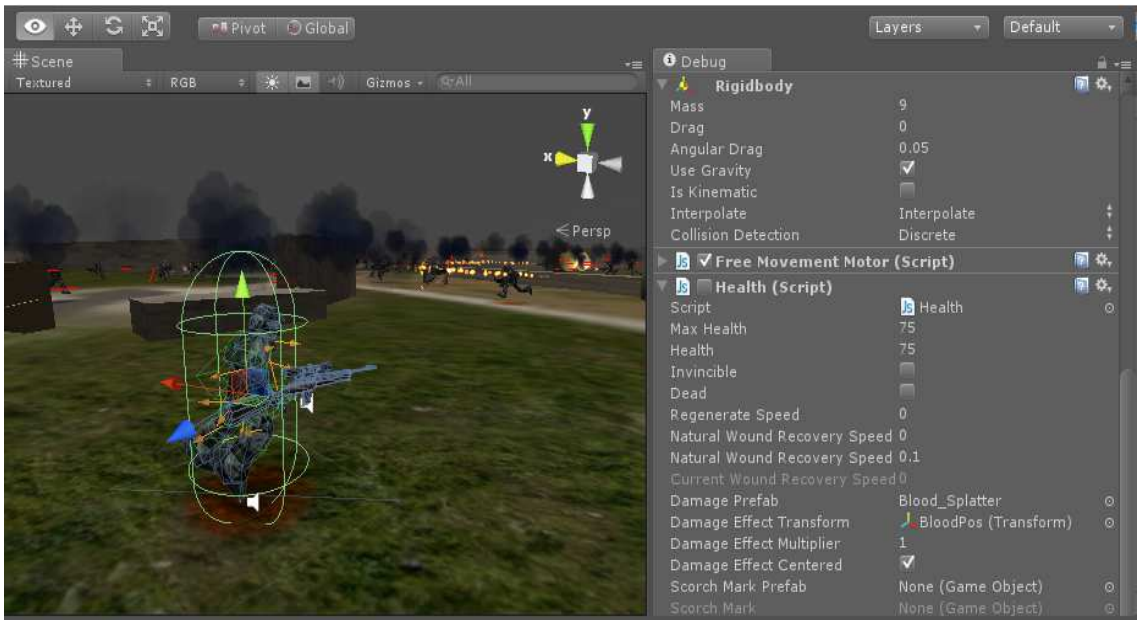
Peliprojektin jäsenten täytyy pystyä määrittelemään pelimaailma ja pelioliot rakenneosista, joita voivat olla muun muassa 3d-mallit, animaatiot, äänet ja skriptatut käyttäytymismallit. Peliohjelmiston täytyy sitten kyetä lukemaan tehdyt määritykset ja monimuotoinen digitaalinen sisältö ja koostaa niistä määritelty pelimaailma sisältöineen muodossa, jota ajonaikainen peliohjelmisto voi tehokkaasti hyödyntää.

Nykyisin peliprojekteissa työskentelee tyypillisesti kymmeniä henkilöitä ja kaikkien pitäisi pystyä näkemään oman työnsä tulos mahdollisimman nopeasti pelissä, jotta mahdolliset ongelmat saadaan korjattua, ennen kuin ne alkavat kertautua. Peliohjelmiston kehitystyökalut voivat tukea peliprojektia tarjoamalla kehittyneet työkalut sisällön koostamiseen sisällöntuotantotyökaluilla tuotetuista osasista sekä varsinaisen pelin testaamiseen. Kehitystyökalujen taso voi siten merkittävästi hidastaa tai nopeuttaa pelin tuottamista.

Esimerkkejä pitkälle viedyistä kehitystyökalusta ovat Unreal Editor [RP08] ja Unity Editor. Unity Editor, jota tämän työn yhteydessä on testattu, on integroitu kehitysympäristö pelikohtaisen toiminnallisuuden ja sisällön tuottamista varten. Pelimaailman olioita voi saman käyttöliittymän avulla koostaa eri tiedostomuodoissa saatavasta digitaalisesta sisällöstä, ja tuotetut kokonaisuudet näkyvät graafisessa käyttöliittymässä samalla taval-

la kuin valmiissa pelissä.

Kuvassa 10 on osittainen kuvaruutukaappaus Unity Editorista, jossa suoritetaan pelimoottoriarvioinnin yhteydessä toteutettua prototyyppipeliä. Kuvan vasemmassa reunassa on näkymäruutu (scene view), jossa näkyy pelimaailma ja valittu peliolio. Oikeassa reunassa on oliotarkistin (object inspector), jossa näkyy valitun peliolion komponenttirakenne ja komponenttien sisäisten muuttujien arvot.



Kuva 10: Osittainen kuvaruutukaappaus Unity Editorista, jossa työn osana toteutettu prototyyppipeli on suorituksessa.

Muut peliohjelmiston osat

Edellä käytiin läpi vain peliohjelmiston suurimpia komponentteja. Niiden lisäksi tarvitaan paljon toiminnallisuutta, jonka tarkkaan läpikäymiseen tässä yhteydessä ei ole mahdollisuutta. Peliohjelmistoissa usein nähtäviä osakokonaisuuksia jo mainittujen lisäksi ovat esimerkiksi animaatiot, graafiset käyttöliittymät, peliohjainten syötteiden käsittely, visuaaliset efektit, 2- tai 3-ulotteinen ääni, peliin sopiva muistinhallinta ja peliolioiden välinen viestinvälitysjärjestelmä. Esimerkiksi Jason Gregoryn kirja kuvaa tästä esityksestä pois jätettyjen osa-alueiden piirteitä tarkemmin [Gre09].

3.4 Pelimoottorit

Peliohjelmiston koostaminen komponenteista helpottaa pelin toteuttamisen työmäärää huomattavasti, mutta jättää peliprojektille paljon työtä, joka liittyy peliohjelmiston rakenteen suunnitteluun ja toteuttamiseen. Kaiken tarvittavan perustoiminnallisuuden rakentamiseen erillisiä komponentteja hyödyntäen liittyy edelleen suuri työmäärä ja epäonnistumisen riski [War08]. Pelimoottorit vievätkin peliohjelmistojen välistä koodin uudelleenkäyttöä vielä kertaluokkaa pidemmälle kuin komponenttien käyttö. Pelimoottoreiden suosio lähti pelikehityksessä suureen kasvuun vuonna 1993 Id Softwaren Doom-pelimoottorin vetämänä.

Tässä tutkielmassa seurataan Jason Gregoryn käyttämää määritelmää ja pelimoottorilla tarkoitetaan laajennettavissa olevaa ohjelmistoa, jota voidaan käyttää perustana useille erilaisille peleille ilman suuria muutoksia [Gre09, sivu 11]. Pelimoottorissa on peliohjelmistoissa tarvittava perustoiminnallisuus jo valmiiksi integroitu yhteen siten, että pelimoottori tarjoaa pelikohtaisen toiminnallisuuden toteuttamiseen selkeän mallin ja rajapinnan. Valmiin perustoiminnallisuuden ja rajapinnan lisäksi pelimoottorin mukana tulee usein myös hyviä kehitystyökaluja pelin sisällön tuotannon sujuvoittamiseen.

Pelimoottorin käyttö pelikehityksen pohjana tarkoittaa sitä, että suurimmat peliohjelmiston arkkitehtuuriin liittyvät päätökset on jo valmiiksi tehty, toteutettu ja todettu toimiviksi. Tähän pisteeseen on paljon matkaa, mikäli peliohjelmistoa lähdetään itse koostamaan markkinoilta löytyvistä komponenteista. Valmiin pelimoottorin käyttö voikin tehostaa pelituotannon prosessia radikaalisti, ja pelimoottoreiden kehitys on vaikuttanut koko peliteollisuuteen [Hui12]. Nykyisin sisällöntuotanto näyttelee tyypillisesti suurempaa roolia peliprojektissa kuin ohjelmointi.

Pelimoottoria pelikehityksen pohjana käyttävän peliprojektin tehtäväksi jää parhaassa tapauksessa pelikohtaisen toiminnallisuuden toteuttaminen pelimoottorin tarjoamaa kätevää rajapintaa hyödyntäen sekä pelin sisällön tuotanto pääosin valmiina saatavilla kehitystyökaluilla ja prosesseilla. Käytännössä pelin tekeminen valmiin pelimoottorin avulla alkaa usein tyhjästä pelimaailmasta ja ”tyhjän pelin” simulaation voi käynnistää jokseenkin välittömästi. Pelin toteuttaminen onnistuu määrittelemällä uudenlaisia peliolioita ja lisäämällä niitä pelimaailmaan, usein integroidun kehitysympäristön kautta.

Termiä pelimoottori käytetään alan kirjallisuudessa väljästi, ja joskus esimerkiksi pel-

kästä piirtomoottorista puhutaan pelimoottorina [And08]. Pelimoottoreita kutsutaan usein myös pelialan väliohjelmistoiksi (middleware), koska ne tuottavat väliohjelmistoille tyypilliseen tapaan joustavan ja uudelleenkäytettävän alustan. Pelimoottoreissa on pelien tehokkaaseen kehittämiseen tarvittava perustoiminnallisuus valmiina. Siitä huolimatta, mihin pelimoottorin raja vedetään, pelimoottoreita on paljon [Wik12]. Tunnettuja esimerkkejä nykypäivän pelimoottoreista ovat Epic Gamesin Unreal Engine 3, Valven Source Engine, Unity Technologiesin Unity ja id Softwaren id Tech 5.

Pelimoottorit ovat laajoja järjestelmiä, jotka tuovat pelin tuottamiseen mukaan omat erityispiirteensä ja reunaehtonsa. Näitä ovat esimerkiksi lisenssiehdot, tekninen tuki, pelikohtaisen toiminnallisuuden toteuttamisessa käytettävä ratkaisumalli, tuetut ohjelmointikielet, moottoriin sisältyvät (ja sisältymättömät) toiminnalliset osakokonaisuudet, ohjelmoijille tarjottu ohjelmointirajapinta, valitut tekniset lähestymistavat, tuetut alustat sekä pelin sisällön tuottamiseen liittyvät tekniset kyvykkyydet, prosessit ja työkalut.

Pelimoottori voi olla optimoitu useiden eri parametrien suhteen. Esimerkiksi:

- hienoimpien pelien tuottamiseen valituille alustoille tai mahdollisimman yhteensopivaksi useiden alustojen kanssa
- suurten pelistudioiden käyttöön tai pienemmille tuotannoille
- 2-ulotteisten pelien tekemiseen tai 3-ulotteisten pelien tekemiseen
- johonkin tiettyyn genreen hyvin sopivaksi tai varsin yleiskäyttöiseksi

Mikäli peliyritys harkitsee pelien kehittämistä valmiin pelimoottorin avulla, on omiin tarpeisiin sopivimman pelimoottorin valinnasta varmasti erittäin suuri hyöty. Pelien tekeminen ja julkaiseminen tietyllä teknologialla sitoo peliyrityksen kyseiseen teknologiaan pitkäksi aikaa. Valinnassa täytyy selvittää huolellisesti omat tarpeet, ja sen jälkeen edetä vertailemaan eri vaihtoehtoja keskenään omien tarpeiden pohjalta.

4 Pelimoottoreita mobiilipelituotantoon

Mobiilipelien tuotantoon liittyy erityispiirteitä, jotka asettavat pelimoottoreille erityisiä vaatimuksia. Tässä luvussa käydään mobiilipelituotannon erityispiirteitä läpi, jonka jälkeen edetään tarkastelemaan joitakin mobiilipelien tuottamiseen soveltuvia pelimoottoreita.

4.1 Mobiilipelituotannon erityispiirteitä

Mobiilipelimarkkinoilla pelin asennuspaketin koko vaikuttaa olennaisesti pelin levikkiin [Wil12]. Mobiilipelien jakelu pelaajien päätelaitteisiin tapahtuu tyypillisesti sovelluskauppojen kautta. Pelien julkaisijat lataavat pelien asennuspaketteja sovelluskauppoihin, ja potentiaaliset asiakkaat käyttävät mobiililaitteilleen esiasennettuja sovelluskauppasovelluksia, joilla he etsivät ja lataavat pelejä puhelinliittymän internetyhteyttä hyödyntäen. Osa sovelluskaupoista asettaa jaeltavien sovellusten koolle ylärajan, ja jotkin sovellukset eivät lataa yli 50 megatavun asennuspaketteja ilman lähiverkkoyhteyttä. Lisäksi potentiaaliset asiakkaat joutuvat usein maksamaan dataliikenteestä tai vähintäänkin odottelemaan suurten tiedostojen latausta epämiellyttävän kauan. Suuri asennuspaketin koko näin ollen vähentää pelin latausmääriä. Mobiilipelien tuotantoon tarkoitetun pelimoottorin pitäisikin helpottaa pienten asennuspakettien tuottamista ja tarvittaessa sen jakamista osiin.

Toinen mobiilipelimarkkinoiden erityispiirre on pelialustojen suuri määrä ja laitteiden suuret keskinäiset erot. Suosituimmat mobiilialustat ovat tällä hetkellä Android ja iOS, mutta huomattavat markkinaosuudet on myös Windows Phonella ja Blackberrylla [Coc12]. Uusia pyrkijöitä mobiilialustojen joukkoon ovat tällä hetkellä esimerkiksi Firefox OS, Ubuntu ja Sailfish [Box13]. Alustoista on käytössä useita vanhoja versioita, minkä lisäksi samaa käyttöjärjestelmää hyödyntävät laitteet voivat erota huomattavasti toisistaan laskentatehon, muistin ja fyysisten ominaisuuksien osalta.

Yhteensopivuus useiden eri alustojen ja alustaversioiden kanssa mahdollistaa pelin julkaisemisen suuremman asiakaskunnan saataville. Lisäksi alustojen markkinaosuuksien kehittymistä tulevaisuudessa on hankala ennustaa, joten yhteensopivuus eri alustojen kanssa on tärkeää myös markkinariskin pienentämisen näkökulmasta. Pelimoottorin oli-

si siis tärkeää olla yhteensopiva useiden alustojen ja alustaversioiden kanssa, ja sillä tuotettujen pelien pitäisi toimia erilaisissa ajoympäristöissä luotettavasti.

Lisäksi mobiililaitteissa on erityisiä ominaisuuksia, kuten rajallinen akkukapasiteetti, paikkatiedon vastaanottimet (GPS), kiihtyvyyssensorit, kamerat ja kosketusnäytöt. Näiden ominaisuuksien tehokas hyödyntäminen luo mahdollisuuksia uudenlaisten pelien toteuttamiseen, joten laajasta tuesta pelimoottorissa on hyötyä.

Seuraavissa aliluvuissa esitellään neljä mobiilipelien tuottamiseen soveltuvaa pelimoottoria. Esiteltävät pelimoottorit pärjäsivät parhaiten myöhemmin esiteltävän vertailun alkuvaiheessa, jossa oli mukana 20 pelimoottoria.

4.2 ShiVa3D Game Engine

ShiVa3D-pelimoottorin ensimmäinen versio on julkaistu 2005. Pelimoottorin toimittaja Stonetrip mainostaa Shivaa maailman yhteensopivimmaksi pelimoottoriksi. Shivan tukemien alustojen lista onkin vaikuttava – Shiva tukee dokumentaation mukaan pelikehitystä peräti yhdeksälletoista alustalle[1]. Tuettuja alustoja ovat muun muassa olennaisimmat mobiilialustat, web-selaimet, pelikonsolit ja tietokonekäyttöjärjestelmät.

Shivalla pelikehitys tapahtuu Shiva Editorissa. Shiva Editorissa on graafinen käyttöliittymä, jossa on näkymä pelimaailmaan. Näkymässä pelimaailma ja pelioliot näkyvät samanlaisina kuin miltä ne tulevat pelissä näyttämään. Pelimaailman sisältöä ja rakennetta voi muokata suoraan näkymän kautta, joten työkalua voi käyttää pelin tapahtumapaikkojen määrittelyyn. Digitaalisen sisällön tuottamista Shiva Editor ei tue, joten pelimaailman äänet, kuvat, pelihahmot ja muut mallit täytyy tuottaa erillisillä digitaalisen sisällöntuotannon työkaluilla. Digitaalinen sisältö tuodaan projektiin Shiva Editorin käyttöliittymän avulla, jonka jälkeen se on käytettävissä pelin sisällön määrittelyyn. Kuvassa 11 on kuvaruutukaappaus Shiva Editorista, jonka oikeassa yläkulmassa on näkymä pelimaailmaan.

1 http://www.stonetrip.com/developer/wiki/index.php?title=Supported_Platforms_and_the_UAT

Kuva 11: Kuvaruutukaappaus Shiva Editorista.

Shivaan pohjautuvissa peleissä pelioliot määritellään koostamalla ne ominaisuuksista (attribute) ja ohjaimista (controller). Ominaisuudet määrittävät peliolion ulkoisen olemuksen pelimaailmassa, ja niitä ovat esimerkiksi muoto (shape), törmääjä (collider) ja efektit. Ohjaimet puolestaan määrittävät peliolion käyttäytymisen pelimaailmassa, ja niitä ovat animaatio, tekoäly, ääniohjain ja dynamiikkaohjain. Ääniohjain ohjaa olion ääniä ja dynamiikkaohjain huolehtii olion fysiikan mallinnuksesta, eli esimerkiksi siitä että olio reagoi oikein törmäyksiin ja impulsseihin. Pelioliot saadaan näin toimimaan halutulla tavalla liittämällä niihin tarvittavat ominaisuudet ja ohjaimet. Ominaisuuksia ja ohjaimia voi edelleen parametroida ja määritellä skripteillä.

Shivassa pelikohtainen toiminnallisuus määritellään C++-kielellä tai Lua-skripteillä, joilla määritelty toiminnallisuus liitetään peliolioiden ohjaimiin. Shivassa on sisäänrakennettu skriptien muokkaustyökalu (Script Editor), johon muokattava koodi aukeaa suoraan Shiva Editorissa.

Shiva Editorin peruslisenssi maksaa $400 jokaista pelien kehitystyötä tekevää henkilöä kohden. Lisenssi oikeuttaa julkaisemaan kaupallisia pelejä Androidille, iOS:lle ja suu-

relle joukolle muita alustoja. Kaikki ominaisuudet saa käyttöönsä kehittyneellä lisenssillä, joka maksaa $2000 kehittäjää kohden.

4.3 Marmalade

Marmalade on yleisluonteinen sovelluskehitysympäristö, joka on tarkoitettu alustariippumattomien sovellusten tuottamiseen. Marmaladea on käytetty useiden Android- ja iOS-pelien tuottamiseen, joista esimerkkejä ovat Plants vs Zombies, Cut The Rope ja Need for Speed Shift.

Marmalade jakautuu Marmalade Systemiin ja Marmalade Studioon. Marmalade System on abstraktiokerros, joka mahdollistaa alustariippuman sovelluskehityksen tarjoamalla sovelluskehitystyökaluja ja alustariippumattoman ohjelmointirajapinnan. Marmalade Studio on joukko työkaluja ja komponentteja, joiden toiminnallisuus on keskittynyt 2- ja 3-ulotteiseen grafiikkaan, geometriaan, animaatioihin, käyttöliittymiin ja tekstin esittämiseen eri kirjasinlajeja käyttäen. Marmalade Studion toiminnallisuus tarjotaan käytettäväksi C++-kielisen rajapinnan kautta.

Marmalade tukee sovelluskehitystä seuraaville alustoille: iOS, Android, Blackberry PlayBook OS, bada (2.0), Windows, OS-X ja LG Smart TV. Yleisenä sovelluskehitysympäristönä Marmalade ei ole erityisesti pelimoottoriksi suunniteltu. Marmaladen käyttäjille on kuitenkin tehty helpoksi saada käyttöönsä kaikki peleissä usein tarvittavat komponentit. Esimerkiksi fysiikkamoottorina Marmaladen kanssa voi käyttää itse valitsemaansa valmista komponenttia, mutta Marmaladen toimittaja Ideaworks3D Limited on valmiiksi muokannut Open Dynamics Enginestä version Marmaladea varten ja asettanut sen helposti saataville. Vastaavasti skriptimoottoriksi on helposti saatavissa kolmannen osapuolen Lua-tulkki, jonka Ideaworks3D on valmiiksi paketoinut. Marmaladeen ei kuitenkaan ole saatavilla sellaisia pelikehityksen työkaluja, joilla pelimaailman sisältö ja pelioliot määritellään.

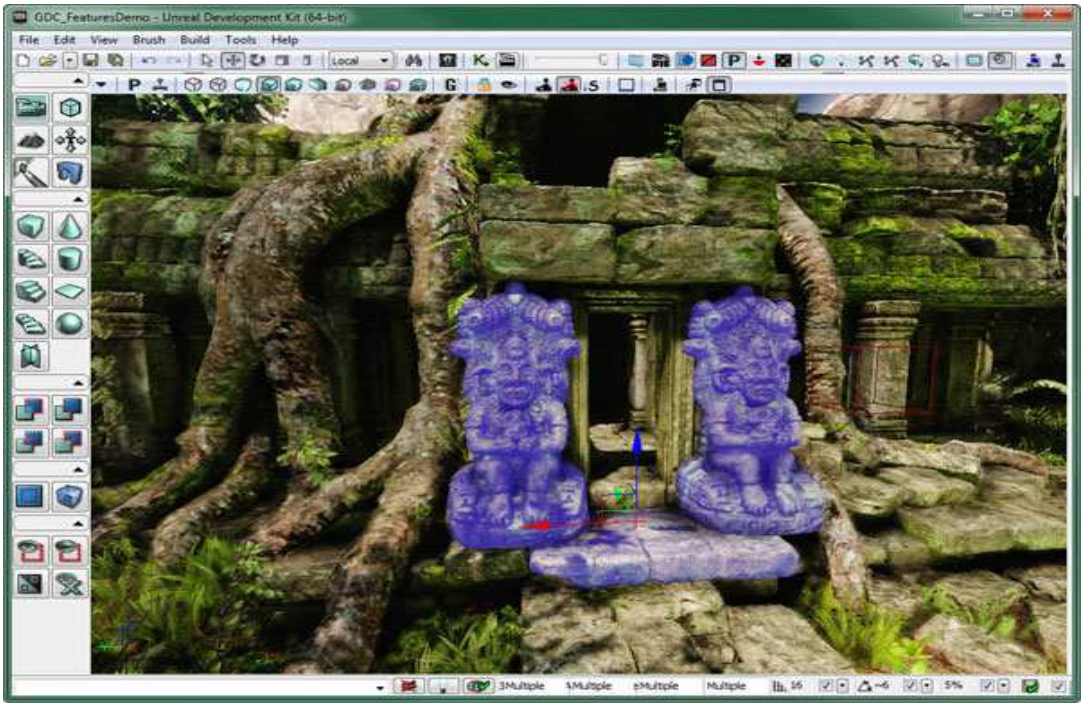
Marmaladen pienille pelikehittäjille suunnattu Indie-lisenssi maksaa $499 jokaista pelien kehitystyötä tekevää henkilöä kohden vuodessa.

4.4 Unreal Engine

Unreal Engine edustaa erittäin tunnettua ja menestynyttä pelimoottoriteknologiaa. Juuri Unreal Enginen toimittaja Epic Games syrjäytti Id Softwaren pelimoottoreiden markkinajohtajan paikalta. Pelimoottorin nykyinen versio on 3, ja Epic Games on ehtinyt kehittää neljättä versiota nykyisen version rinnalla jo vuodesta 2003 lähtien. Uudesta versiosta on julkaistu vuonna 2013 Infiltrator-nimeä kantava teknologiademo.

Unreal Engine -teknologia jakautuu kahteen pelimoottorituotteeseen, joista molemmilla on tehty useita huippupelejä. Täyden lisenssin Unreal Engine sisältää pelimoottorin ja useiden esimerkkiprojektien lähdekoodit, ja se on Epic Gamesin mukaan suunnattu kokeneille pelikehityksen ammattilaisille. Unreal Development Kit, myöhemmin UDK, on Epic Gamesin tarjoama vaihtoehto pienille tai aloitteleville pelikehittäjille. UDK on valmiiksi käännettynä ohjelmana toimitettava rajoitettu versio täydestä pelimoottorista. Epic Games suosittelee UDK:n käyttämistä ennen Unreal Enginen lisensointia. UDK tukee iOS- ja Windows-pelien tuottamista. Unreal Engine tukee näiden lisäksi Flash, Android, Xbox 360, OS-X, PS3, PS Vita ja Wii U -pelien tuottamista.

Pelimaailman määrittelyyn Unreal Enginen kanssa käytetään Unreal Editoria, jossa on peliä vastaava näkymä muokattavaan pelimaailmaan. Näkymän kautta pelimaailman sisältöä on helppo muokata ja tulokset näkee välittömästi. Digitaalisen sisällön tuottamista Unreal Editor ei tue, joten pelimaailmassa nähtävät äänet, pelihahmot ja muut mallit täytyy tuottaa erillisillä digitaalisen sisällöntuotannon työkaluilla. Digitaalinen sisältö tuodaan projektiin Unreal Editorin käyttöliittymän avulla, jonka jälkeen se on käytettävissä pelin sisällön määrittelyyn. Kuvaruutukaappaus Unreal Editorista on kuvassa 12.



Kuva 12: Kuvaruutukaappaus Unreal Editorista.

Pelikohtainen toiminnallisuus toteutetaan Unreal Engineen perustuvissa peleissä UnrealScriptillä, mutta täyden pelimoottorin käyttäjien täytyy Epic Gamesin mukaan kyetä myös ymmärtämään ja muokkaamaan pelimoottorin C++-pohjaista toiminnallisuutta. Unreal Engine määrittää valmiin rakenteen pelin tärkeimmille toiminnallisuuksille kuten pelin tyyppi (gametype), useat erilaiset ohjaimet (controllers) ja hahmot (pawn). Esimerkiksi hahmoille tulee pelimoottorin mukana valmis perustoiminnallisuus, joka sisältää muun muassa toiminnallisuutta tavaroiden keräämiseen ja säilyttämiseen. Peliä erikoistetaan luomalla valmiille toteutuksille aliluokkia. Epic Games ei tarjoa C++-koodin ja UnrealScriptin ylläpitoon työvälineitä, vaan pelikehittäjät valitsevat itse mitä työvälineitä käyttävät. Kolmannet osapuolet ovat toteuttaneet UnrealScriptin ylläpitoon työvälineitä, kuten maksullinen WOTgreal tai ilmaiset Unreal-X Editor ja UDK IDE. Työvälineistä löytyy tietoa Epic Gamesin keskustelualueelta[1] ja Unreal Enginen dokumentaatiosta.

Unreal Development Kit -lisenssi on yrityskohtainen ja maksaa $99. Lisäksi Epic Gamesille tulee maksaa rojalteja 25 prosenttia kaikista yli $50000 tuotoista, jotka on ansaittu UDK:ta hyödyntäen. Unreal Enginen hinta on tapauskohtainen, eikä siitä ole saatavilla tietoa muuten kuin aloittamalla neuvottelut Epic Gamesin myynnin kanssa.

1 http://forums.epicgames.com/

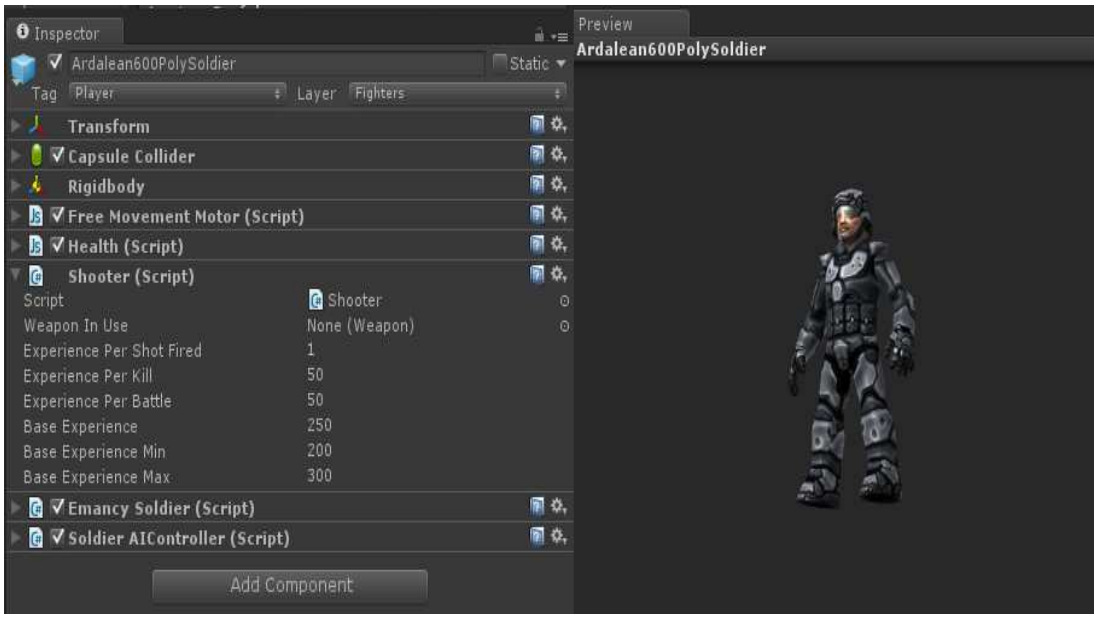
4.5 Unity

Unity on pelimoottori, joka on pyritty optimoimaan helppokäyttöiseksi ja tehokkaaksi työvälineeksi alustariippumatonta pelituotantoa varten. Unity vaikuttaa olevan selkeä markkinajohtaja mobiilipelimoottoreiden joukossa. Sillä on yli miljoona rekisteröitynyttä kehittäjää [Uni12], ja Game Developer Magazinen mobiilipelikehitystä koskevassa tutkimuksessa 53.1 prosenttia pelikehittäjistä raportoi käyttävänsä Unitya [Del12].

Unity tukee iOS- ja Android-alustojen lisäksi pelien tuottamista seuraaville alustoille: Windows, Windows Store Apps, Windows 8 Phone, OS-X, Linux, verkkoselaimet, PS3, Xbox 360 ja Wii U. Lisäksi Unity Technologies on maaliskuussa 2013 julkaissut tiedotteen tekevänsä Sonyn kanssa yhteistyötä laajentaakseen alustatukea Sonyn nykyisille ja tuleville alustoille.

Unitylla pelikehitys tapahtuu Unity Editorissa. Unity Editorissa on graafinen käyttöliittymä, jossa pelimaailman näkymä (scene) ja pelioliot näkyvät samanlaisina kuin miltä ne tulevat pelissä näyttämään. Pelimaailman sisältöä ja rakennetta voi muokata suoraan Unity Editorissa, joten työkalua voi käyttää pelin tapahtumapaikkojen määrittelyyn. Digitaalisen sisällön tuottamista Unity Editor ei tue, joten pelimaailmassa nähtävät äänet, pelihahmot ja muut mallit täytyy tuottaa erillisillä digitaalisen sisällöntuotannon työkaluilla. Unityn mallissa digitaalinen sisältö tallennetaan projektin hakemistohierarkiaan, jonka jälkeen se on suoraan käytettävissä pelin sisällön määrittelyyn. Unity kääntää ja lataa uudet ja muutetut skriptit sekä digitaalisen sisällön välittömästi muutosten tallentamisen jälkeen, ja muutokset on heti nähtävissä Unity Editorissa.

Unityn valitsemassa arkkitehtuurimallissa pelioliot määritellään koostamalla ne komponenteista. Komponentit tarkoittavat tässä yhteydessä pieniä toiminnallisia kokonaisuuksia, joita lisäämällä saadaan peliolioille kaikki toivotut ominaisuudet. Esimerkkejä peliolion komponenteista ovat 3D-transformaatio peliolion sijaintia ja skaalausta varten, piirtäjäkomponentti peliolion visualisointia varten, äänilähde peliolion äänien toistamista varten sekä erilaiset skriptikomponentit, jotka saavat peliolion käyttäytymään halutulla tavalla. Kuvassa 13 on osittainen kuvakaappaus Unity Editorista. Kuvan vasemmalla puolella on oliotarkistin, jossa näkyy oikealla puolella olevan sotilashahmon komponenttirakenne.



Kuva 13: Osittainen kuvakaappaus Unity Editorista.
Vasemmalla näkyy sotilashahmon komponenttirakenne.

Unityyn pohjautuvissa peleissä pelikohtainen toiminnallisuus määritellään UnityScript-, C#- tai Boo-skripteinä, jotka liitetään komponentteina peliolioihin. Tarjotuista vaihtoehdoista Boo-kieli ei tosin vaikuta olevan yleisesti käytössä. Unity Editoriin on integroitu MonoDevelop-ohjelmakoodin muokkaustyökalu, johon skriptit aukeavat kaksoisklikkaamalla skriptin nimeä tai skriptiin liittyvää virheilmoitusta Unity Editorissa.

Unityn mukana toimitetaan paljon sisältöä, jota voi käyttää opetteluun ja omissa projekteissa rajoituksetta. Lisäksi pelimoottorin käyttäjille on yhteinen kauppapaikka, jossa esimerkiksi yleiskäyttöistä skriptattua toiminnallisuutta, 3-ulotteisia malleja ja animaatioita on helppo ostaa ja myydä. Kauppapaikkaa pääsee käyttämään suoraan Unity Editorista.

Unityn rajoitetumman perusversion käyttö on maksutonta yrityksille, joiden vuotuinen liikevaihto ei ylitä sataa tuhatta dollaria. Unityn ammattilaislisenssin hinta on $1500 jokaista pelien kehitystä tekevää henkilöä kohden. Hinta sisältää OS-X, Windows, Windows Store, Windows Phone 8, verkkoselain ja Linux -julkaisuoikeudet. Tämän lisäksi jokaisen mobiilialustan julkaisuoikeuden hinta on $1500 jokaista kehittäjää kohti. Ammattilaislisenssi iOS- ja Android-julkaisuoikeuksin maksaa siis $4500 jokaista pelien kehitystyötä tekevää henkilöä kohti.

Kaikilla tässä luvussa esitellyillä pelimoottoreilla on se yhteinen piirre, että niillä on tuotettu merkittäviä pelejä mobiilialustoille useiden eri pelitalojen toimesta (katso liite 5, evidenssi asiakaskäytöstä). Tämä on osoitus siitä, että vakavasti otettavan peliyrityksen teknologia-alustana voi toimia mikä tahansa näistä pelimoottoreista. Esitellyt pelimoottorit kuitenkin eroavat toisistaan huomattavasti esimerkiksi arkkitehtuurin, kehitystyökalujen, alustatuen ja lisenssiehtojen suhteen. Absoluuttisessa mielessä parasta pelimoottoria ei voida nimetä, mutta pelimoottoreiden laatua voidaan arvioida, ja siten verrata, erilaisten tarpeiden näkökulmasta.

5 Ohjelmistojen laadun arviointi

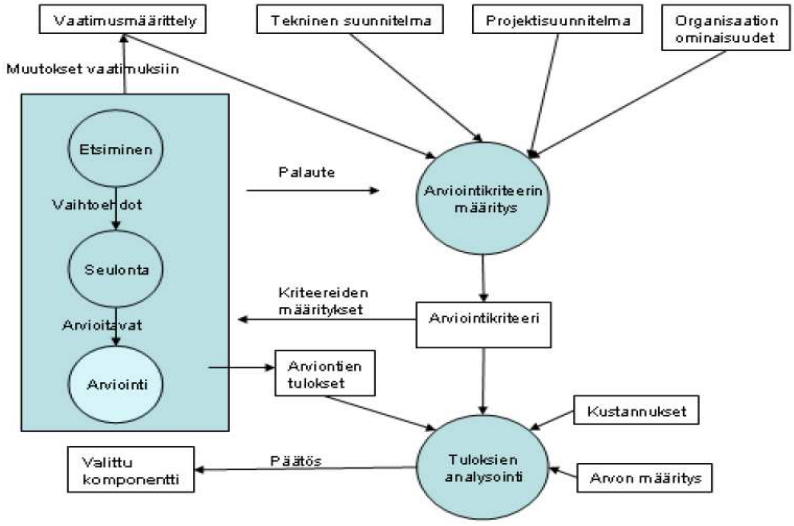
Järjestelmän laatu on suure, joka kuvaa missä määrin järjestelmä täyttää siihen eri sidosryhmiltä kohdistuvat tiedostetut ja tiedostamattomat tarpeet ja siten tuottaa lisäarvoa [ISO11b]. Laadun arvioinnilla selvitetään systemaattisesti, missä määrin järjestelmä saavuttaa sille määritellyt kriteerit [ISO11a]. Tässä työssä tehtävässä pelimoottoreiden vertailussa on siis kyse laadun arvioinnista, joten aihepiiriin tutustutaan tässä luvussa.

5.1 Johdatus ohjelmistojen laadun arviointiin

ISO/IEC 25040 -standardin mukaan laadun arvioinnin tavoitteena on, että käyttöön otettava ohjelmistotuote täyttää siihen kohdistuvat tiedostetut ja tiedostamattomat tarpeet [ISO11a]. Ohjelmistojen laadun arviointia suoritetaan tavalla tai toisella aina, kun ohjelmistotuotteita kehitetään tai hankitaan. Sovellusten kehittämiseen liittyvät muutokset ja sovellusten hankinnat perustuvat aina arvioon muutoksen hyödyllisyydestä, joka on jonkinlaisen laadun arviointitoiminnan tulos.

Ohjelmistolla on useita laatupiirteitä, joiden suhteellinen tärkeys riippuu ohjelmistolle suunnitellusta käyttötarkoituksesta. Ohjelmiston laatua täytyy arvioida, jotta voidaan päätellä, vastaavatko ohjelmiston laadulliset piirteet käyttötarkoituksen kannalta olennaisia vaatimuksia kuten toiminnallinen kattavuus, käytön taloudellisuus ja luotettavuus [ISO11a].

Laadun arviointi ei toisaalta kata koko ohjelmistotuotteen valintaprosessia, vaan ohjelmistojen valinnassa tarvitaan myös muuta toimintaa. Esimerkki kokonaisesta ohjelmistokomponenttien valintaprosessista on OTSO-menetelmä, joka kattaa tarvittavan toiminnan ohjelmistokomponenttien etsimisestä valitsemiseen asti [Kon95]. Kuvassa 14 on OTSO-menetelmän mukainen valintaprosessi, jonka osana oleva arviointiprosessi on korostettu.



Kuva 14: Laadun arviointi osana ohjelmistokomponenttien valintaan soveltuvaa OTSO-prosessia. Kuvan lähde [Oin06].

Laadun arviointiin on olemassa suuri joukko valmiita menetelmiä. ISO ja IEC ovat yhdessä luoneet joukon kansainvälisiä standardeja ohjelmistotuotteiden laatuvaatimusten määrittelyyn, laadun mittaamiseen ja laadun arviointiin liittyen. Näitä ovat muun muassa laatumalli ISO/IEC 9126-1, ulkoiset mittarit ISO/IEC 9126-2, evaluointiprosessi ISO/IEC 14598-5 ja uudemmat ISO/IEC 25000 -sarjan standardit, jotka korvaavat aiempia standardeja ohjelmistojen laatuun liittyen.

ISO/IEC 25000 -sarjan, niin sanotun SQuaRE-sarjan (Software Quality, Requirements and Evaluation) standardien luonnissa tavoitteena on ollut muodostaa loogisesti organisoitu, rikastettu ja yhdenmukainen sarja standardeja kattamaan ohjelmistojen laatuun liittyvät kaksi ydinprosessia: laatuvaatimusten määrittely ja laadun arviointi. Ydinprosesseja sarjassa tuetaan laadun mittausprosessilla. Lisäksi sarja määrittelee kattavan laatumallin, joka sisältää useimmissa tapauksissa olennaiset näkökulmat ohjelmiston laatuun. Sarja on tarkoitettu hyödynnettäväksi ohjelmistojen kehityksessä, hankinnassa tai riippumattomassa arvioinnissa. Sarja on jaettu kuuteen kokonaisuuteen, joita ovat laatuvaatimukset, laatumalli, laadun hallinta, laadun mittaus, laadun arviointi ja lisäosat. Laadun arvioinnin kannalta erittäin keskeistä sisältöä ovat näissä standardeissa esitetyt laadun arviointiprosessit ja laatumallit, joihin tutustutaan seuraavassa aliluvussa.

5.2 Laatumallit ja laadun arviointiprosessit

Ohjelmiston laadun arvioinnin tavoitteen saavuttamisen kannalta on tärkeää tunnistaa kaikki arvioitavaan ohjelmistotuotteeseen kohdistuvat olennaiset laatuvaatimukset. ISO/IEC-standardien laatumallit ovat valmiita malleja, joissa ohjelmistotuotteiden kaikki olennaiset laadulliset piirteet on jo pyritty valmiiksi kuvaamaan. Laatumalleja voi käyttää esimerkiksi muistilistana tapauskohtaisia laatuvaatimuksia määritettäessä, tai laatumallia vasten voi tarkastaa määriteltyjen laatuvaatimusten kattavuuden. Lisäksi laatumalli luo yhteisen terminologian puhuttaessa ohjelmistojen laatupiirteistä.

Standardissa ISO/IEC 25010 esitetään laatumallit sekä ohjelmiston käytönaikaisen laadun että ohjelmistotuotteen laadun arviointia varten, ja standardi ISO/IEC 25012 esittää laatumallin datan laadun arviointia varten. Datan laatu kuvaa missä määrin data täyttää siihen eri sidosryhmiltä kohdistuvat tiedostetut ja tiedostamattomat tarpeet [ISO08]. ISO/IEC 25012 -standardin datan laatumallissa on 15 laatupiirrettä, joita voidaan käyttää hyödyksi datan laatuvaatimusten määrittelyssä ja datan laatuarviointia tehdessä. Datan laatupiirteitä ovat esimerkiksi oikeellisuus, tarkkuus, kattavuus, ajantasaisuus, saatavuus ja palautettavuus. Ohjelmiston käytönaikaisen laadun ja ohjelmistotuotteen laadun arviointia koskevat osuudet ovat tämän työn kannalta olennaisimpia, joten niitä esitellään seuraavaksi.

Standardin ISO/IEC 25010 laatumallissa käytönaikaisen laadun määrittelee viisi ylätason laatupiirrettä, jotka liittyvät tuotteen ja sen sidosryhmien muodostaman kokonaisuuden toimintaan. Nämä ovat vaikuttavuus, tehokkuus, tyytyväisyys, riskittömyys ja tarpeen kattavuus. Mallissa ylätason laatupiirteille on tunnistettu alipiirteitä, esimerkiksi tyytyväisyyden alipiirteet ovat hyödyllisyys, luottamus, mielihyvä ja mukavuus. Kaikki standardissa käytetyt termit on tarkasti määritelty. Käytönaikainen laatu kuvaa vaikutusta, joka tuotteella on sen sidosryhmiin. Tämä määrittyy ohjelmiston laadun, laitteiston, toimintaympäristön, käyttäjien, tehtävien ja sosiaalisen ympäristön perusteella. Käytönaikainen laatu kuvaa ohjelmistotuotetta laajemman systeemin toimintaa, ja siihen voivat vaikuttaa esimerkiksi yksilöiden väliset suhteet ja järjestelmän ulkopuoliset tapahtumat, jotka vaikuttavat järjestelmän käyttäjiin.

Standardin laatumallissa ohjelmistotuotteen laadun määrittelee kahdeksan ylätason laatupiirrettä: toiminnallinen sopivuus, suorituskykytehokkuus, yhteensopivuus, käytettävyys, luotettavuus, tietoturvallisuus, ylläpidettävyys ja siirrettävyys. Laatupiir-

teillä on jälleen alipiirteitä, esimerkiksi suorituskykytehokkuuden alipiirteitä ovat aikakäyttäytyminen, resurssien käyttö ja kapasiteetti [ISO11b, sivut 10-16]. Ohjelmistotuotteen laatumallia voi soveltaa ohjelmistotuotteeseen tai laajempaan tietojärjestelmään, jossa ohjelmisto on osana.

5.3 ISO/IEC 25040 -standardi laadun arviointiin

SQuaRE-sarjan standardi ISO/IEC 25040 kuvaa viitemallin ohjelmistotuotteen laadun arviointiin. Viitemallissa keskeisenä sisältönä on arviointiprosessi ja vaatimukset prosessin soveltamiselle. Vaatimuksiin kuuluvat muun muassa arviointiin liittyvien vaatimusten standardin mukainen määrittely, arvioinnin määrämuotoinen dokumentointi ja standardin mukaisen laatumallin käyttäminen laatuvaatimusten määrittelyssä. Arviointiprosessia, soveltamisvaatimuksia ja ohjeita tarkennetaan standardin liitteissä standardin eri käyttäjäryhmiä varten. Standardin tunnistamia käyttäjäryhmiä ovat ohjelmistotuotteen kehittäjät, hankkijat ja riippumattomat arvioijat.

Viitemalli sisältää arviointiprosessin aktiviteettien yksityiskohtaiset kuvaukset tavoitteineen, syötteineen ja tuotteineen. Lisäksi viitemallissa on täydentävää tietoa esimerkiksi arvioinnin rajoituksista, resursseista ja arviointiin liittyvistä rooleista. Viitemallin kuvaamaa prosessia tullaan soveltamaan pelimoottoreiden vertailussa, joten se käydään tässä tarpeellisilta osin läpi.

Viitemallin arviointiprosessissa on viisi vaihetta ja edeltävien vaiheiden tuotokset toimivat prosessissa aina seuraavan vaiheen syötteinä. Standardissa hyväksytään, että prosessi ei etene täysin vesiputouksen lailla alusta loppuun, vaan iterointia voi tapahtua prosessin vaiheiden välillä. Seuraavissa kappaleissa kuvataan korkealla tasolla prosessin vaiheet.

Ensimmäinen vaihe laadun arviointiprosessissa on arvioinnin vaatimusten laatiminen. Tässä vaiheessa selvitetään ja kirjataan arvioinnin tarkoitus. Ohjelmistotuotteen laatuvaatimukset selvitetään, ja tunnistetaan mitkä tuotteen osat tullaan arvioimaan. Lisäksi tässä vaiheessa määritellään vielä arvioinnin tiukkuus (tarkkuus, ankaruus) arvioinnin tarkoitusta vastaavalle tasolle. Vaiheen lopputuotosta kutsutaan laadun arvioinnin vaatimusdokumentaatioksi.

Toinen arviointiprosessin vaihe on arvioinnin määrittely. Tässä vaiheessa valitaan suori-

tettavat mittaukset ja määritellään päätöskriteerit mittauksille. Mittausten päätös-kriteerejä noudattaen mittauksia joko jatketaan tai ne lopetetaan suoritusvaiheessa saatujen tulosten perusteella. Lisäksi määritellään kriteerit, joita noudattaen tullaan arvioimaan laatua kokonaisuutena saatujen mittaustulosten perusteella. Vaiheen lopputuotosta kutsutaan laadun arvioinnin määrittelyksi.

Kolmas arviointiprosessin vaihe on arvioinnin suunnittelu. Tässä vaiheessa sidotaan resurssit arviointiin, tunnistetaan arviointiin kuuluvat tehtävät ja aikataulutetaan tehtävät. Kyse on käytännössä projektisuunnitelman laatimisesta arvioinnin suorittamiselle. Vaiheen lopputuotosta kutsutaan arviointisuunnitelmaksi.

Neljäs vaihe on arvioinnin suorittaminen. Tässä vaiheessa suoritetaan mittaukset ja kirjataan mittaustulokset. Mittauksia tehdessä sovelletaan päätöskriteereitä, joiden perusteella mittauksia jatketaan tai ne lopetetaan. Lopulta sovelletaan laadun arviointikriteereitä saatuihin mittaustuloksiin laadun arvioimiseksi. Vaiheen lopputuotosta kutsutaan arvioinnin tulokseksi.

Viides vaihe on arvioinnin loppuunsaattaminen. Tässä vaiheessa katselmoidaan arvioinnin tulos ja muodostetaan arviointiraportti. Arviointiraportti koostuu kaikesta arviointiin liittyvästä dokumentaatiosta. Arviointiraportti luovutetaan sovituille osapuolille.

6 Suoritettu pelimoottoreiden laadun arviointi

Tässä luvussa kuvataan tutkielman yhteydessä suoritettu pelimoottoreiden arviointi Sadetta Oy:n tarpeisiin. Arviointi kohdistuu mobiilipelien tekemiseen soveltuviin pelimoottoreihin, ja arvioinnin avulla on tarkoitus löytää yritykselle parhaiten sopiva pelimoottori. Tarkemmin arviointiprosessin vaiheiden syötteet, aktiviteetit ja tuotokset on kuvattu liitteissä, joihin tekstistä viitataan. Arvioinnista ei ole tuotettu erillistä arviointiraporttia, vaan seuraavat aliluvut ja aliluvuissa mainitut liitteet yhdessä muodostavat arviointiraportin.

6.1 Arvioinnin vaatimukset

Sadetta Oy on vuonna 2009 perustettu mobiilipelejä tuottava pieni yritys. Yrityksen ydinosaamista on pelien suunnittelu ja pelien tekninen tuottaminen. Audiovisuaalisen sisällön tuotannossa käytetään alihankintaa. Yritys pyrkii tuottamaan pelejä käyttäjämääriltään suurimmille mobiilialustoille, joita ovat tänä päivänä Android ja iOS [Per12].

Yritys on tuottanut mobiilipelejä Qt-teknologialla [Dig13]. Qt on käytössä todettu yrityksen tarpeisiin melko heikosti soveltuvaksi, koska yleiskäyttöisenä teknologiana se ei erityisesti tue pelien tuotantoa. Qt-pohjaisia komponentteja pelissä tarvittavien toiminnallisuuksien toteuttamiseen on toki saatavilla, mutta pelien tuottaminen Qt:lla on osoittautunut työlääksi pelinkehitykseen erityisesti suunniteltujen kehitystyökalujen puuttuessa. Lisäksi Qt ei ole ainakaan toistaiseksi yhteensopiva iOS- tai Android-alustojen kanssa, joten sillä tehtyjä pelejä ei voi julkaista yrityksen tärkeimmille kohdemarkkinoille.

Arvioinnin tarve ja tarkoitus

Yritys on päättänyt siirtyä uuteen teknologia-alustaan, jolla pelejä pystytään tuottamaan tehokkaasti suosituimmille mobiilialustoille nyt ja tulevaisuudessa. Uusi teknologia-alusta on tarkoitus valita valmiina saatavilla olevien pelimoottoreiden joukosta. Järjestelmältä odotetut tärkeimmät hyödyt ovat yrityksen pelituotannon tehostuminen, ja suosituimpien mobiilialustojen saaminen tuettujen alustojen piiriin nyt ja tulevaisuudessa. Järjestelmän käyttöiän toivotaan olevan yli 3 vuotta.

Yrityksellä on tarve tunnistaa ja hankkia markkinoilla olevista pelimoottoreista omiin tarpeisiin paras vaihtoehto. Yritykselle sopivimman pelimoottorin tunnistaminen on tutkielman osana tehtävän pelimoottorivertailun tarkoitus.

Vaatimukset pelimoottorille

Toiminnallisesta näkökulmasta yritys tarvitsee kokonaisratkaisun, jolla pystyy tuottamaan pelejä vähintään Android- ja iOS-alustoille. Pelimoottorilla tulisi voida tuottaa pelejä, joissa on 3-ulotteista grafiikkaa, 3-ulotteista fysiikkasimulaatiota, äänet ja taustamusiikit, animaatioita, graafiset käyttöliittymät ja verkkoliikennettä. Pelien pelaajilla voi olla käytössään erilaisia mobiililaitteita, joita käytetään peliohjaimina. Androidia ja iOS:aa laajemmasta, myös mobiilialustojen ulkopuolelle ulottuvasta, alustatuesta on yritykselle hyötyä. Tarkemmin toiminnalliset vaatimukset on kuvattu liitteessä 1.

Laadun näkökulmasta yritykselle avainasemassa on pelituotannon tehokkuus (ISO/IEC 25010 efficiency). Tehokkuus tarkoittaa tässä sitä, että pelien tuottaminen pelimoottorilla vaatii yritykseltä suhteellisesti vähän resursseja, varsinkin pelin kehitykseen käytettävää työaikaa. Toinen yritykselle tärkeä laadun osa-alue on sopeutumiskyky (ISO/IEC 25010 adaptability). Jotta pelimoottorilla voidaan tuottaa pelejä tärkeimmille alustoille nyt ja tulevaisuudessa, täytyy pelimoottorin olla yhteensopiva useiden mobiilialustojen kanssa sekä sopeutumiskykyinen mobiilimaailmassa tapahtuviin alustamuutoksiin. Kolmas yritykselle erityisen tärkeä laadun osa-alue on yleinen yhteensopivuus (ISO/IEC 25010 interoperability). Pelimoottorin yhteensopivuus eri sisällöntuotannon työkalujen ja tiedostomuotojen kanssa on tärkeää, jotta ulkoisilta tahoilta hankittava digitaalinen sisältö on vaivattomasti integroitavissa peliin. Yrityksen tarpeiden ja ISO/IEC 25010 -laatumallin perusteella laadittu laatuvaatimusmäärittely on kuvattu liitteessä 2. Vaatimusmäärittelyssä on huomioitu yrityksen ja pelien pelaajien laatuvaatimukset.

Vaatimukset ja rajoitukset arvioinnille

Parhaan pelimoottorin löytäminen ensi yrittämällä olisi erittäin toivottavaa, mutta ei yrityksen liiketoiminnan kannalta välttämätöntä. Yrityksen kannalta on tärkeää, että arviointi viedään nopeasti läpi ja käytännössä testataan pelimoottorin sopivuutta yrityksen käyttöön. Vertailun suunnitteluun ja suorittamiseen on käytettävissä pääosin vain allekirjoittaneen työaikaa. Vertailun kriittisyys on alhainen.

Mahdollisesti tarvittavat hankinnat sovitaan yrityksen ja allekirjoittaneen välillä tapauskohtaisesti. Yrityksen edustaja, DI Vesa Oinonen, osallistuu arvioinnin loppuvaiheessa suoritettavaan prototyyppiprojektiin, jossa valittua pelimoottoria kokeillaan käytännössä.

Arvioitavat pelimoottorit

Koska kyse on parhaan vaihtoehdon etsimisestä, arviointiin on alkuvaiheessa pyritty sisällyttämään kaikki tunnetut pelimoottorit, jotka tukevat 3-ulotteisten pelien kehitystä iOS- ja Android-alustoille. Mainitut valintaehdot ovat yrityksen asettamia välttämättömiä vaatimuksia, joiden perusteella on helppo rajata internetistä hakemalla löytyviä pelimoottoreita.

Lista arvioitavista pelimoottoreista on kerätty internetistä mobiilipelimoottoreita etsimällä maaliskuussa 2013. Arvioitavista pelimoottoreista Unreal Engineä lukuun ottamatta kaikki löytyivät myös www.mobilegameengines.com:sta. Arvioitavia pelimoottoreita on 20, ja ne on listattu liitteessä 3.

Arvioitavat pelimoottorit koostuvat ohjelmistotuotteina verkkodokumentaatiosta, pelinkehitystyökaluista sekä pelimoottorin mukana tulevista sovelluskehyksistä ja ohjelmointirajapinnoista toteutuksineen. Pelimoottori ohjelmistotuotteena ulottuu pelimoottorilla tuotettaviin peleihin asti, koska suuri osa pelimoottorilla tuotettavan pelin toteutuksesta tulee valmiina pelimoottorista. Pelimoottorin käytönaikaisessa arvioinnissa tarkastellaan järjestelmää, jossa yritys kehittää pelimoottorilla prototyyppipeliä. Lisäksi käytönaikaisessa arvioinnissa tarkastellaan tuotetun prototyyppipelin toimivuutta tärkeimmillä kohdealustoilla, joita ovat iOS- ja Android-mobiililaitteet.

6.2 Arvioinnin määrittely ja arviointisuunnitelma

Vertailussa mitattavat laatuvaatimukset on pyritty valitsemaan siten, että vertailussa mukana olevien pelimoottoreiden laadusta saadaan aluksi nopeasti aikaan objektiivinen arvio. Objektiivisen arvion perusteella parhaalta vaikuttavaan pelimoottoriin syvennytään prototyyppipeliprojektissa, jonka perusteella muodostetaan lopullinen arvio pelimoottorin laadusta. Lähestymistapa vastaa yrityksen vaatimusta siitä, että parhaalta vaikuttavat pelimoottorit saadaan rajattua nopeasti ja parhaalta vaikuttavaa pelimoottoria päästään testaamaan käytännössä.

Arviointi tullaan suorittamaan kolmessa vaiheessa. Ensimmäisessä vaiheessa selvitetään kaikkien kahdenkymmenen pelimoottorin toiminnallinen kattavuus yrityksen käyttötarkoituksessa. Selvityksessä kaikkien yrityksen vaatimien toiminnallisuuksien löytyminen pelimoottorista tarkastetaan pelimoottorin virallisesta dokumentaatiosta. Tarkemmin mittaus on kuvattu liitteessä 4. Yrityksen näkökulmasta toiminnoiltaan epärelevantit pelimoottorit suljetaan pois arvioinnin myöhemmistä vaiheista arvioinnin tehostamiseksi.

Arvioinnin toisessa vaiheessa vertaillaan dokumentaation perusteella toiminnallisesti kattavia pelimoottoreita keskenään. Toisen vaiheen ensimmäisessä mittauksessa kerätään evidenssiä pelimoottoreiden käytöstä tarkastelemalla pelimoottoreilla vuoden 2012 alun jälkeen tehtyjä julkaisuja. Pelimoottorin aktiivinen käyttö kertoo hyvästä markkina-asemasta, joka on tuotteen tulevaisuuden kannalta tärkeää. Aktiivinen ja monipuolinen käyttö on omiaan parantamaan myös pelimoottorin laatua, toiminnallista kattavuutta ja joustavuutta. Tarkemmin mittaus on kuvattu liitteessä 5.

Toisen vaiheen toisena mittauksena tarkastetaan pelimoottoreiden alustatuen laajuus virallisesta dokumentaatiosta. Laaja alustatuki on jo itsessään hyödyllinen ominaisuus, mutta se osoittaa myös aiempaa mukautuvuutta uusille alustoille pelimoottorilta ja sen toimittajalta. Aiempi mukautuvuus useille alustoille viittaa mukautuvuuteen myös tulevaisuudessa. Mittaus on kuvattu liitteessä 6.

Toisen vaiheen mittausten jälkeen lasketaan arvioinnin toisessa vaiheessa mukana olleille pelimoottoreille laatupisteet laskemalla suoraan yhteen toisen vaiheen mittausten tulokset. Laatupisteet muodostavat objektiivisen kokonaisarvion pelimoottoreiden laadusta yrityksen käytössä. Laadun objektiivisen arvioinnin jälkeen kokonaisuutena, siis myös muut ominaisuudet kuin mitattu laatu huomioiden, yritykselle parhaalta vaikuttava pelimoottori valitaan arvioinnin kolmanteen vaiheeseen yhdessä yrityksen edustajan kanssa. Edellä kuvatun objektiivisen laatuarvioinnin toivotaan huomattavasti rajaavan vaihtoehtoja ja siten helpottavan valintaa.

Arvioinnin kolmannessa vaiheessa kokeillaan valittua pelimoottoria käytännössä tuottamalla pelimoottorin avulla prototyyppipeli. Prototyyppipeliprojektissa testataan erityisesti seuraavia pelimoottorin piirteitä: pelituotannon tehokkuus annetuilla työvälineillä, yhteensopivuus Android ja iOS-mobiililaitteiden kanssa, resurssitehokkuus, 3-ulotteisen piirron ominaisuudet ja suorituskyky, äänet ja 3-ulotteinen fysiikkasimulaatio. Mikäli pelimoottori ei prototyyppiprojektin perusteella vaikutakaan lupaavalta,

voidaan siirtyä arvioimaan kokonaisuutena seuraavaksi parhaalta vaikuttanutta pelimoottoria.

Arvioinnin suorittamisessa käytetään hyödyksi ISO/IEC 25040 -standardissa kuvattua laadun arviointiprosessia ohjelmistotuotteen hankkijan näkökulmasta, mutta arviointi ei ole täysin standardin mukainen. Standardin tuomaa valmista rakennetta ja laatumallin kattavuutta hyödynnetään kattavan laatuvaatimusmäärittelyn ja arvioinnin aikaansaamiseksi.

Arvioinnin ensimmäisessä ja toisessa vaiheessa ISO/IEC 25040 -standardia pyritään noudattamaan täysin. Arvioinnin kolmatta vaihetta, prototyyppipeliprojektia, ei kuitenkaan suoriteta standardin edellyttämällä tavalla. Prototyyppipeliprojekti ja siihen liittyvä laadun arviointi päätettiin toteuttaa epämuodollisesti, koska standardin mukainen laatuvaatimusten ja mittausten määrittely, dokumentointi ja suorittaminen vaatisivat huomattavasti työaikaa ja samanaikaisesti yksittäistä projektia tarkastelemalla ei kuitenkaan saataisi aikaan objektiivista arviota projektin käyttämän pelimoottorin laadusta. Tämä johtuu siitä, että pelikehityksessä kehittäjän suunnittelupäätökset vaikuttavat hyvin merkittävästi mittaustuloksiin, kuten pelikehityksen tehokkuuteen ja tuotetun pelin laatuun. Tällöin pelimoottorin laadun objektiivista arviointia varten otannan pitäisi olla suurempi.

Yksittäistä projektia tarkastellen tehdyt arviot pelimoottorin laadusta ovat siis joka tapauksessa subjektiivisia. Tässä tapauksessa arviot perustuvat allekirjoittaneen aiempaan kokemukseen pelikehityksestä erilaisin työkaluin. Arviointisuunnitelma on kuvattu liitteessä 13.

6.3 Arvioinnin eteneminen ja tulokset

Toiminnallisen kattavuuden tarkastaminen pelimoottoreiden verkkodokumentaatiosta suoritettiin 1.4. - 5.4.2013. Tarkastuksen perusteella 12 pelimoottoria 20:sta todettiin vain osittain yrityksen tarpeet kattaviksi. Nämä 12 pelimoottoria jätettiin jatkoselvitysten ulkopuolelle, koska yritys etsii valmista kokonaisratkaisua pelien tuottamiseen. Mukaan arviointiin jäivät Antiryad GX, Esenthel Engine, Marmalade, ShiVa3D Game Engine, SIO2 Engine, UNIGINE Engine, Unity ja Unreal Engine. Mittauksen tulokset on kuvattu liitteessä 7.

Seuraavana mittauksena hankittiin evidenssiä pelimoottoreiden käytöstä tutkimalla verkkodokumentaatiossa mainittuja viimeaikaisia referenssipelejä. Tässä mittauksessa jatkotarkasteluun mukaan otetut 8 pelimoottoria jakautuivat kahteen ryhmään. Ensimmäiseen ryhmään kuuluvilla neljällä pelimoottorilla oli huomattavasti julkaistuja pelireferenssejä vuoden 2012 alusta lukien, kun toiseen ryhmään kuuluvilla niitä oli vain muutama tai ei lainkaan. Hyvin mittauksessa suoriutuivat Unity (10 pistettä), Unreal Engine (10), ShiVa3D (10) ja Marmalade (9). Tarkemmin tulokset on kuvattu liitteessä 8.

Kolmantena mittauksena selvitettiin pelimoottoreiden verkkodokumentaatiosta alustatuen laajuus. Kaikki 8 pelimoottoria tukivat iOS- ja Android-alustojen lisäksi vähintään Windows- ja OS-X-alustoja. Ylivoimaisesti laajin alustatuki oli ShiVa3D:lla (17 pistettä). Seuraaviksi sijoittuivat Unity (8), Unreal Engine (7) ja Marmalade (5). Mittaustulokset ovat nähtävissä liitteessä 9. Huomattavaa on, että molemmissa vertailevissa laatumittauksissa edellä mainitut 4 pelimoottoria menestyivät parhaiten.

Johtopäätökset arvioinnin toisen vaiheen jälkeen

Dokumentaatioon perustuvassa laatuarvioinnissa parhaat kokonaispisteet saivat ShiVa3D Game Engine (27), Unity (18), Unreal Engine (17) ja Marmalade (14). Loput toiminnallisesti kattavat pelimoottorit jäivät selkeästi jälkeen sekä referenssipelien määrässä että tuettujen alustojen määrässä. Tarkemmin pisteytys on kuvattu liitteessä 10. Viimeaikaisten referenssipelien puuttuminen herättää huolen pelimoottorin tulevaisuudennäkymistä ja yhteensopivuudesta uusien alustojen kanssa. Jo tästä syystä vaikuttaa selvästi kannattavalta valita pelimoottori neljän arvioinnissa parhaiten menestyneen joukosta.

Objektiivisella arvioinnilla saatiin rajattua yritykselle mahdollisesti sopivien pelimoottoreiden joukko neljään, mikä on saavutus sinänsä. Toisaalta arvioinnin pistejärjestys neljälle parhaalle määräytyi käytännössä suoraan alustatuen laajuuden perusteella. Koska suuresta alustatuesta ei ole yritykselle merkittävästi suoraa hyötyä, kokonaisuutena parhaan vaihtoehdon valinnassa laatupisteiden painoarvo jää pieneksi.

Pelimoottorin valinta prototyyppipeliprojektiin

Pelimoottorin valinta päätettiin yhdessä yrityksen edustajan kanssa suorittaa vertailussa

neljän parhaiten menestyneen pelimoottorin joukosta. Valinnan eteneminen käydään perusteluineen läpi seuraavissa kappaleissa.

ShiVa3D sai vertailussa korkeimmat pisteet poikkeuksellisen laajan alustatuen perusteella. Lisäksi pelimoottori sai täydet 10 pistettä referenssipeleistä. Vertailussa riittävästä referenssipelipistemäärästä huolimatta ShiVa3D:n toimittaja Stonetrip näyttää juuri äskettäin ajautuneen selvitystilaan [Cha13]. Tämä muodostaa ShiVa3D:n jatkokehitykselle ja tuelle suuren uhan.

Unity tuli vertailussa toiseksi. Unity on erittäin suosittu pelimoottori, ja se vaikuttaa kaikin puolin lupaavalta yrityksen käyttötarkoitusta ajatellen.

Epic Gamesin Unreal Engine tuli vertailussa kolmanneksi. Unreal Engine vaikuttaa teknisesti hyvältä pelimoottorilta, mutta Sadetta Oy ei vastaa Epic Gamesin kuvaamaa asiakasyrityksen profiilia ”kokenut ammattimainen pelikehittäjä”. Unreal Development Kit, joka on tarkoitettu yrityksen kaltaisille riippumattomille toimijoille, ei tue Android-pelikehitystä. UDK:n alustatuki on Windowsiin ja iOS:iin rajoittuvana muutenkin suppeampi kuin yritys toivoo.

Marmalade ylsi neljän parhaan joukkoon. Yritys näkee samankaltaisuutta Qt:n ja Marmaladen yleisluontoisessa sovelluskehitysarkkitehtuurissa. Haasteen yritykselle muodostaa tässä arkkitehtuurissa se, että osa olennaisimmista komponenteista täytyy itse valita ja integroida peliin. Tällöin pelin eri osa-alueiden väliset integraatiot ja rajapinnat täytyy itse määritellä ja toteuttaa. Esimerkiksi grafiikkakirjaston lataamista 3-ulotteisista malleista ei saa ilman integrointityötä muodostettua fysiikkamoottoriin samanmuotoisia kappaleita, koska fysiikkamoottori on täysin erillinen kolmannen osapuolen komponentti. Pelin sisällön määrittelyyn ei myöskään ole valmiita työkaluja, kuten muilla neljän parhaan joukossa olevilla pelimoottoreilla.

Edellisten pohdintojen perusteella Unity vaikuttaa yritykselle selkeästi sopivimmalta pelimoottorilta, ja prototyyppiprojekti tullaan toteuttamaan Unitylla. Tiivistäen vielä perustelut, joilla muut vaihtoehdot suljettiin pois:

- ShiVa3D jätettiin valitsematta siihen liittyvien tulevaisuuden uhkakuvien vuoksi
- Unreal Engine jätettiin valitsematta, koska se on suunnattu isoille pelitaloille. Pienille pelikehittäjille tarkoitettu UDK ei tue Androidia.

- Marmalade jätettiin valitsematta, koska sen kanssa jouduttaisiin itse integroimaan komponentteja yhteen. Lisäksi pelikehityksen valmiit työkalut jäisivät osin puuttumaan. Tämä heikentäisi pelikehityksen tehokkuutta.

6.4 Prototyyppipeliprojektin eteneminen ja tulokset

Prototyyppiprojektissa Unity asennettiin aluksi yhdelle Windows 8 -koneelle, jolla allekirjoittanut kehitti prototyyppipeliä. Aluksi prototyyppiprojektissa toteutettiin yläviistosta kuvatun taistelupelin alku ilman valikkoja. Prototyyppipeliin toteutettiin tässä vaiheessa liikkuminen näppäimistön ja hiiren avulla, animoitu sotilashahmo, kolme erilaista asetta, räjähdys, rättinukkekuolema, yksi taistelukenttä, hiukkasefektit (particle effects), graafiset käyttöliittymäelementit sekä tietokoneen toimia ohjaava tekoäly.

Ensimmäinen työviikko projektissa kului huolelliseen tutustumiseen Unityyn. Tutustuminen tapahtui lukemalla Unityn dokumentaatiota ja tutustumalla Unityn esimerkkiprojekteihin, varsinkin Unityn asennuspaketin mukana toimitettavaan Angry Botsiin. Viikon tutustumisen jälkeen luotiin oma projekti, jonka pohjaksi otettiin kaikki Angry Botsin sisältö kenttiä (scene) lukuun ottamatta. Näin saatiin prototyypin rakentamista varten valmiina useita 3-ulotteisia malleja ja taistelupeleissä usein tarvittavaa toiminnallisuutta. Angry Botsin toiminnallisuutta hyödyntäen omaa peliä lähdettiin kehittämään, ja seuraavien kolmen työviikon aikana allekirjoittanut toteutti kaikki yllä mainitut ominaisuudet.

Jo prototyyppiprojektin alkuvaiheessa kävi selväksi, että pelin kehitys-testi-sykli Unitylla oli erittäin nopea. Unity kääntää ja lataa muutettua sisältöä (skriptejä, kuvia, ääniä ja niin edelleen) jatkuvasti kehityksen aikana ja sisällön näkee Unity Editorissa jatkuvasti sellaisena, kuin se pelissä tulee olemaan. Pelin pystyy Unity Editorissa käynnistämään milloin tahansa, kunhan kaikki muutetut skriptit ovat menneet kääntäjästä läpi. Kun peliä suoritetaan Unity Editorissa, muutokset sisällössä näkyvät jopa pelin suorituksen aikana.

Suoritusaikaiset poikkeukset skripteissä eivät kaada peliä, vaan poikkeuksista tulee virheilmoitukset Unity Editorin konsoliin ja peli jatkuu. Virheilmoitusta kaksoisklikkaamalla pääsee MonoDevelopilla muokkaamaan sitä koodiriviä, jossa poikkeus tapahtui. Lisäksi skriptien sisäisten muuttujien arvot ovat suorituksen aikana Unity Editorissa nähtävissä ja pelin saa väliaikaisesti pysähtymään virheen sattuessa (Error

Pause). Kaikki tämä tekee pelin testaamisesta sekä virheiden havaitsemisesta, selvittämisestä ja korjaamisesta erittäin nopeaa. Skriptien sisäisten muuttujien arvot ovat suorituksen aikana nähtävissä ja jopa muutettavissa, mikä nopeuttaa myös pelin hienosäätöä todella paljon. Sopivat arvot vakioille voi näin ollen määritellä kokeilemalla eri arvoja pelin suorituksen aikana.

Unitylle ominainen peliolioiden koostaminen pienistä toiminnallisista komponenteista edisti prototyyppiprojektin alusta asti toiminnallisuuden uudelleenkäyttöä, kun prototyyppipelissä voitiin suoraan, tai pienellä muokkauksella, hyödyntää useita toiminnallisuuksia Angry Botsista. Esimerkiksi sotilaiden animointia, terveyttä (health), liikkumista, asetta, luoteja ja räjähdyksiä simuloiva sisältö otettiin suoraan Angry Botsista ja niitä muokattiin vain tarvittavilta osin.

Animaatioiden ja hiukkasefektien testaus

Projektin alkuvaiheessa kaikki sotilaat näyttivät Angry Botsin päähenkilöiltä ja hahmojen animaatiot olivat suoraan Angry Botsista. Oman hahmon integrointi omaan peliin on välttämätöntä, joten sitä testattiin heti alkuvaiheessa. Testaus suoritettiin ottamalla käyttöön Unityn uusi Mecanim-animaatiojärjestelmä, jolla humanoidihahmon animaatiot voi uudelleenkohdistaa toiseen humanoidihahmoon (animation retargeting). Unityn kauppapaikalta haettiin ilmainen sotilashahmo ja siihen uudelleenkohdistettiin Angry Botsin päähenkilön animaatiot. Mecanimin käyttöönotto aiheutti jostain syystä animoitavan pelihahmon siirtymisen noin metrin verran alaspäin ja näkymän ulkopuolella olevien hahmojen siirtymisen koordinaatiston origoon. Ongelmista löytyi tietoa Unityn käyttäjäfoorumilta, ja ne saatiin kierrettyä noin yhden päivän työllä. Angry Botsin animaatiot saatiin uudelleenkohdistuksen avulla toimimaan uuden sotilashahmon kanssa melko helposti.

Alkuvaiheessa peli näytti ankealta. Tunnelmaa haettiin toteuttamalla peliin raketti ja sille räjähdys, joka muun muassa säteilee valoa. Jälleen Angry Botsin valmiita komponentteja käytettiin pohjana, jolloin työaikaa säästyi huomattavasti. Myös luodin osuminen seinään laitettiin tuottamaan kipinöitä. Efekti toteutettiin käyttäen Unityyn kuuluvaa Shuriken-hiukkasefektijärjestelmää. Kipinöitä tuottavan hiukkassuihkun pystyi näkemään Unity Editorissa samalla, kun sen parametreja muokattiin. Kipinäefektin toteuttaminen ja integrointi peliin kesti näin vain alle kaksi tuntia.

Rättinukkemallin käyttöönotto

Sotilaat jähmettyivät alkuvaiheessa kuollessaan paikoilleen. Tämä oli kelvottoman näköistä, joten seuraavana vuorossa oli kuoleman näyttävämpi toteuttaminen sotilaille. Kuolema päätettiin toteuttaa rättinukkemallin avulla. Tällöin sotilaan kuollessa sotilasta esittävä malli korvataan rättinukkemallilla, jonka liikkeitä ohjaa fysiikkasimulaatio animaation sijaan. Rättinukkemallissa on jäykkiä kappaleita, jotka ovat toisissaan kiinni sopivin liitoksin siten, että yhdistelmän fyysinen käyttäytyminen muistuttaa löysää ihmisruumista. Sotilaan 3-ulotteinen malli (mesh) on sidottu fyysisiin kappaleisiin siten, että 3-ulotteinen malli liikkuu ja muuttaa muotoansa fyysisten kappaleiden liikkeiden mukana.

Rättinukkemallia hyödyntäen jokainen kaatuminen näyttää erilaiselta, eikä erillisiä animaatioita tarvitse kuolemaa varten toteuttaa. Unitystä löytyi automaattitoiminnallisuus rättinukkemallin tuottamiseksi humanoidihahmolle. Toiminnallisuus vaati tiedon hahmon luurankomallin tietyistä nivelistä ja halutusta rättinuken kokonaispainosta. Nämä saatiin määriteltyä syöttämällä haluttu paino sekä raahaamalla ja tiputtamalla (drag & drop) sotilaan luurankomallin nivelet syötekenttiin Unity Editorissa. Tämän jälkeen toiminnallisuus loi rättinuken automaattisesti. Automaattisesti luotua rättinukkea pystyi helposti jälkeenpäin hienosäätämään Unity Editorissa muokkaamalla siihen liittyvien fyysisten kappaleiden kokoja ja painoja. Rättinukkekuolemien toteuttaminen ja integrointi prototyyppipeliin kesti näin alle yhden työpäivän. Kuvassa 15 vasemmalla on sotilashahmolle tuotettu rättinukkemalli ja oikealla on prototyyppipelistä otettu osittainen kuvakaappaus, jossa sotilaita makaa eri asentoihin kaatuneina maassa.

Kuva 15: Vasemmalla sotilaalle tuotettu rättinukkemalli ja oikealla eri asentoihin kaatuneita sotilaita.

Vaikutelma Unityn kehitystyökaluista ja hajautetun kehityksen testi

Myöskään PC-ohjausrajapinnan, tekoälyn tai muun pelikohtaisen toiminnallisuuden toteuttaminen peliin ei tuottanut ikäviä yllätyksiä. Päinvastoin, pelikehityksen edetessä Unityn rajapinta osoittautui helppokäyttöiseksi, laadukkaaksi ja tarpeet hyvin kattavaksi. Ainoastaan näyttävien käyttöliittymien toteuttaminen Unitylla osoittautui hankalaksi. Unity on tiedostanut ongelman jo aikaisemman käyttäjäpalautteen perusteella, ja ilmoittanut kehittävänsä täysin uutta ratkaisua käyttöliittymien toteuttamiseen. Käyttöliittymiä lukuun ottamatta kaikkiin eteen tulleisiin ongelmiin löytyi hyviä ratkaisuvinkkejä Unityn foorumeilta. Noin 3 viikon kuluttua pelissä oli kaksi joukkuetta, jotka ottivat mittaa toisistaan taisteluissa, joiden välillä näytettiin tilastoja hyvin yksinkertaisella valikkoruudulla. Ihmispelaaja pystyi hiirellä ja näppäimistöllä ohjaamaan toisen joukkueen yhtä sotilasta, loppujen sotilaiden liikkuessa tekoälyn ohjaamina.

Eniten pelikehitystä haittaava ongelma oli MonoDevelopin kaatuminen suunnilleen päivittäin. Myös Unity Editor jumittui muutamia kertoja, mutta huomattavasti harvemmin. Tässä vaiheessa PC-pelien tuottamisen tehokkuus Unitylla joka tapauksessa vaikutti ilmeiseltä, ja prototyyppiprojektissa oli aika siirtyä kokeilemaan hajautettua kehitystä yhdessä yrityksen edustajan kanssa.

Hajautetun kehityksen testissä Unity asennettiin myös yrityksen edustajan Windows 8 PC:lle, ja keskitettyyn projektitiedostojen versionhallintaan otettiin käyttöön Git, joka on Unitysta riippumaton versionhallintajärjestelmä[1]. Git-versionhallinnan käyttöönotto oli projektissa helppoa, koska siihen löytyi Unityn verkkodokumentaatiosta valmiit ohjeet. Ainoana hajautettuun kehitykseen liittyvänä ongelmana havaittiin, että Unity Editor ei aina luotettavasti lataa uudelleen kaikkea muuttunutta sisältöä, kun muutoksia on haettu versionhallinnasta. Tämän vuoksi on tarpeen kokeilla Unity Editorin uudelleenkäynnistystä, mikäli versionhallinnasta haetun päivityksen jälkeen esiintyy ongelmia.

Alustayhteensopivuuden ja suorituskyvyn testi

Seuraavaksi oli vuorossa Android-pelituotannon käytännön testaus. Pelin asennuspaketin tuottaminen ja asentaminen Android-alustalle oli odottamattoman helppoa. Aluksi valmisteltiin Google Nexus 7 ja PC Android-kehitykseen Googlen ohjeiden mukaisesti. Tämän jälkeen, kun laite oli PC-koneessa kiinni, valittiin Unity Editorin asetuksista kohdealustaksi Android ja valittiin ”rakenna ja suorita” -vaihtoehto. Muutamien minuuttien prosessoinnin jälkeen peli käynnistyi Android-laitteessa.

Peliä ei vielä tässä vaiheessa päässyt pelaamaan, koska kosketusnäytölle ei ollut toteutettu käyttöliittymää. Yksinkertainen kosketuskäyttöliittymä toteutettiin valikkoja varten, ja itse peliin integroitiin Angry Botsin kahteen virtuaaliseen ohjainsauvaan perustuva käyttöliittymä sotilaan ohjauksen toteuttamiseksi. Kun peliä muutama tunti myöhemmin päästiin Androidilla pelaamaan, grafiikka näytti olevan PC-versiota vastaavalla tasolla ja suorituskyky oli yllättävän hyvä. Peli oli lähes pelikelpoinen kuvan virkistystaajuuden ollessa 15 kuvaa per sekunti tai yli. Tämä on hyvä arvo, kun huomioidaan, että pelissä oli 40 animoitua sotilasta. Käytetyn sotilaan hahmossa oli noin 1200 kolmiopintaa, ja aseissakin oli kolmiopintoja noin 1000 per ase. Yrityksen edustaja testasi pelin tuottamista ja suorittamista toisella PC:lla Samsung Galaxy S2:n kanssa, jolloin kaikki toimi yhtä hyvin kuin Google Nexus 7:n kanssa.

Android-testauksen jatkuessa huomattiin, että Android-asennuspaketin rakentaminen Unitylta epäonnistuu satunnaisesti, jolloin rakennusprosessin joutuu käynnistämään uudelleen. Myös Mecanim-animaatiot hajosivat satunnaisesti Android-versiossa, jonka

1 http://git-scm.com/

jälkeen ne saatiin takaisin toimimaan vain tekemällä mikä tahansa muutos sotilaan animaatioita ohjaavaan Mecanim-tilakoneeseen. Havaitut satunnaiset ongelmat hidastavat kehitystä ja vähentävät työmukavuutta, mutta eivät erityisen merkittävästi.

Lopuksi testattiin prototyyppipeliä iPhone 5:n kanssa. Applen politiikan mukainen rekisteröityminen kehittäjäksi sekä iPhone 5:n rekisteröinti kehityskäyttöön oli suoritettava ennen varsinaista testiä. Pelien tuottaminen iOS:lle on Unitylla mahdollista vain OS-X-koneella, joten Unity Editor asennettiin Macbook Pro -kannettavaan. Asennuksen jälkeen peliprojektin sisältö haettiin Git:sta, jonka jälkeen päästiin rakentamaan ja asentamaan asennuspaketti iPhoneen. Pääpiirteissään rakennus- ja asennusprosessi vastaa Android-alustan prosessia. Suurimpana erona on se, että iOS-prosessissa Unityn ja laitteen välisen rajapinnan hoitaa Xcode, joka on Applen integroitu kehitysympäristö. Unity tuottaa ja lataa Xcodeen projektin valmiiksi ensimmäisellä rakennuskerralla.

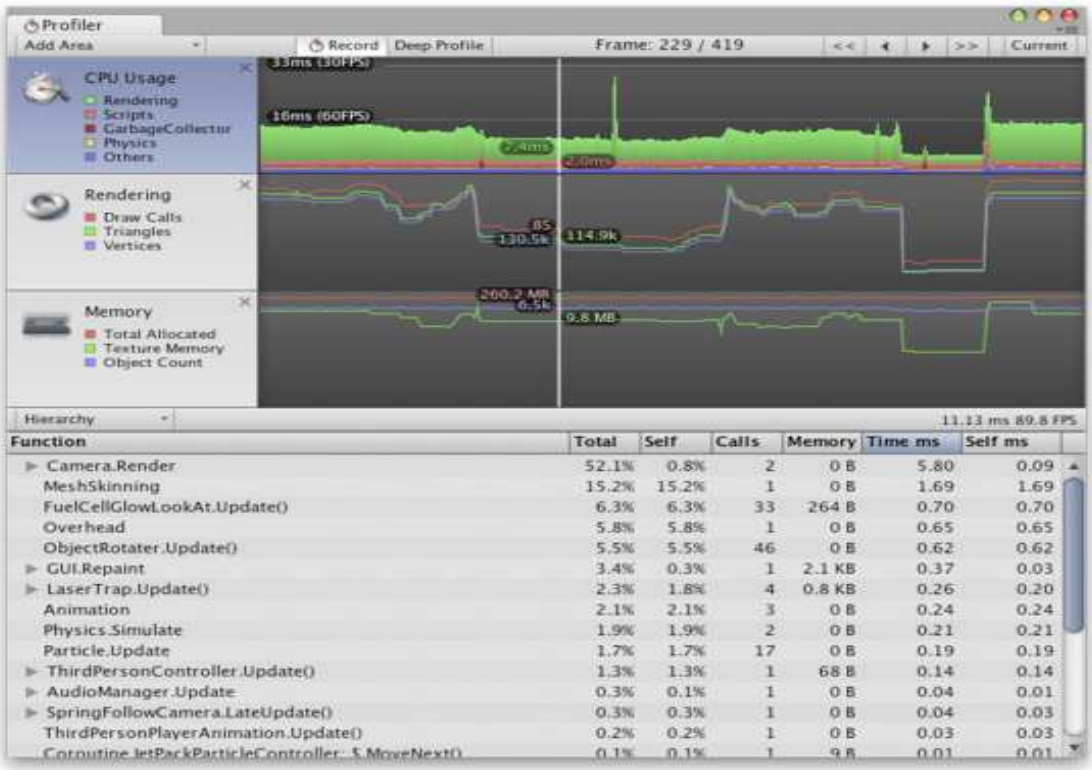
Prototyyppipeli toimi iPhone 5:ssa hyvin. Pelin asennuspaketin koko oli kuitenkin iOS-puolella noin 7 megatavua suurempi kuin 26 megatavun Android-käyttöjärjestelmän asennuspaketti. Suuremmalle tilantarpeelle ei ehditty projektissa selvittää syytä, mutta se saattaa vaikeuttaa isokokoisten pelien jakelua Applen sovelluskaupan kautta, koska sovelluskauppa rajoittaa datayhteyden kautta ladattavan asennuspaketin koon 50 megatavuun. Havaitsimme myös, että ainoa Unityn iOS-alustalla tukema tekstuurien pakkausformaatti on PVRTC, joka tukee vain neliön muotoisia tekstuureita [Fen03]. Tällöin Unity tarvittaessa venyttää tekstuurin neliöksi ennen pakkausta ja litistää sen alkuperäiseen kokoon purkamisen jälkeen, mikä huonontaa tekstuurien laatua. Ongelman voi kuitenkin kiertää ohittamalla Unityn automaattisen tekstuurien käsittelyn kokonaan. Positiivinen asia puolestaan oli se, että satunnaisia asennuspaketin asennuksen epäonnistumisia ei iOS-prosessissa havaittu.

Resurssitehokkuuden ja suorituskyvyn ongelmakohtien korjaaminen

Pelin virrankulutus oli aluksi erittäin suuri. Syyksi paljastui se, että peli pyrki päivittämään ruudun 60 kertaa sekunnissa, mikä vaati useimmissa pelitilanteissa lähes 100% mobiililaitteen suorituskyvystä. Virrankulutus saatiin laskemaan sopivalle tasolle tiputtamalla Unityn ruudunpäivitysnopeuden tavoitearvo 30 ruutuun sekunnissa. Toimenpide vaati vain yhden rivin lisäämisen peliä ohjaavaan skriptiin.

Unityn mukana toimitetaan profilointityökalu, joka kerää tiedot pelin eri komponenttien

suoritusajoista jokaista ruudulle muodostettua kuvaa kohden. Profilointityökalua testattiin Android-laitteiden kanssa suorituskyvyn pullonkaulojen havaitsemiseksi ja korjaamiseksi. Profiloinnin avulla saatiin selville, että aluksi Nexus 7:ssa pullonkaulana oli suorittimen aika, jota erityisesti fysiikkasimulaatio kulutti huomattavasti. Pullonkaula saatiin pian poistettua, ja profilointityökalun avulla löydettiin ja korjattiin seuraavatkin pullonkaulat nopeasti. Työkalu osoittautui erittäin hyödylliseksi analysoitaessa pelin suorituskyvyn pullonkauloja. Kuvassa 16 on kuvaruutukaappaus Unity Editorin profilointi-ikkunasta. Kuvan profilointityökalussa on valittu tarkasteluun 229. ruudulle muodostetun kuvan pelitapahtumien laskentaan liittyvä suoritinajan käyttö. Kuvan alareunassa olevan erittelyn perusteella selviää, että piirto (Camera.Render) kulutti tuolloin 52.1% pelin kuluttamasta suoritinajasta. Kuvan yläosasta näkee trendejä pidemmältä ajanjaksolta ja siitä selviää muun muassa, että piirto (Rendering) kuluttaa jatkuvasti eniten suorittimen aikaa.



Kuva 16: Kuvaruutukaappaus Unity Editorin profilointi-ikkunasta

Johtopäätökset prototyyppipeliprojektin jälkeen

Prototyyppipeliprojektin aikana totesimme käytännössä, että Unity on erittäin tehokas pelimoottori mobiilipelien tuottamista varten. Skripteille tarjottu ohjelmointirajapinta oli looginen, kattava ja toimi odotusten mukaisesti. Lisäksi Unityn esimerkkiprojektit, foorumi ja sovelluskauppa tarjosivat lähes valmiita ratkaisuita useimpiin projektin aikana vastaan tulleisiin ongelmiin. Molemmat helpottavat uuden toiminnallisuuden toteuttamista huomattavasti ja osoittavat Unityn saavuttaneen kypsyyttä aktiivisessa asiakaskäytössä.

Unityn valitsema arkkitehtuurimalli, jossa pelioliot koostetaan pienistä toiminnallisista komponenteista, osoittautui helposti omaksuttavaksi, käytännössä toimivaksi ja uudelleenkäyttöä tehokkaasti edistäväksi. Unityn esimerkkiprojekteista, foorumilta ja sovelluskaupasta löytyvät ratkaisut olivat usein valmiita komponentteja, jotka saatiin helposti otettua käyttöön omassa projektissa.

Pelin kehittämiselle ominainen iterointi ja virheiden selvitys havaittiin niin ikään projektin aikana Unityn tarjoamilla työkaluilla erittäin nopeaksi. Peliä voi kehityksen aikana suorittaa Unity Editorissa, ja koska Unity Editor kääntää muutetut skriptit heti muutosten tallentamisen jälkeen, kestää muutetun pelin käynnistäminen Unity Editorissa tyypillisesti vain sekunteja. Unity Editorissa ajettavan pelin saa pysähtymään virhetilanteessa automaattisesti ja peliolioiden sisäisten muuttujien arvot ovat heti nähtävissä. Lisäksi virheen aiheuttanut skripti avautuu virheilmoitusta kaksoisklikkaamalla integroituun ohjelmakoodin muokkaustyökaluun. Ominaisuudet nopeuttavat virheenselvitys- ja kehitystyötä huomattavasti.

Havaitut laadulliset ongelmat liittyvät satunnaisiin jumittumisiin Mono Developissa ja Unity Editorissa. Näistä ei ollut merkittävää haittaa projektille. Toiminnallisena puutteena havaittiin, että monipuolisen käyttöliittymän toteuttaminen on Unityn tarjoamilla valmiilla työkaluilla[1] hankalaa. Unityn työkaluilla oli helppoa tehdä prototyyppikäyttöliittymät, mutta näyttävien käyttöliittymien tuottamiseen hankaluuksia aiheuttaa esimerkiksi se, että Unityn työkaluilla luodut käyttöliittymät ovat aina kaiken muun, esimerkiksi mahdollisten efektien, edessä. Oletettavasti tästä syystä Unityn kauppapaikalta löytyykin käyttöliittymien toteuttamiseen useampia työkalukokonaisuuksia, kuten

1 http://docs.unity3d.com/Documentation/Components/GUIScriptingGuide.html

NGUI[1] ja DF-GUI[2].

Kokonaisuutena Unityn luotettavuus oli odotusten mukainen, ja pelin kehitys Unitylla oli erittäin tehokasta ja jopa mukavaa. Tuotettu prototyyppipeli oli allekirjoittaneen ja yrityksen edustajan arvioiden mukaan käytettyyn aikaan suhteessa erittäin pitkällä. Pelin luotettavuus, suorituskyky ja resurssitehokkuus olivat jopa yllättävän hyvällä tasolla. Tarkemmin prototyyppiprojektin aikana muodostuneet arviot Unitysta on kuvattu liitteissä 11 (yrityksen käyttötapaus) ja 12 (pelaajan käyttötapaus). Tulosten perusteella Unity näyttää olevan erittäin hyvin yritykselle sopiva teknologia-alusta.

6.5 Arvioinnin katselmus

Arvioinnissa käytetyt laatuvaatimukset pohjautuivat ISO/IEC 25010 -standardin laatumalliin, joka määrittelee selkeästi ja kattavasti ohjelmistojen laadun piirteet. Valmiin laatumallin käyttö auttoi huomioimaan pelimoottoriin kohdistuvat laatuvaatimukset kokonaisvaltaisesti, ja käytännössä laatumalli toimi ikään kuin muistilistana laatuvaatimuksia määriteltäessä. Arvioinnissa seurattiin ISO/IEC 25040 -standardissa esitettyä laadun arviointiprosessia, joka määritteli selkeästi arvioinnin etenemisen vaihe vaiheelta sekä eri työvaiheiden syötteet ja tuotokset. Prosessin noudattaminen toi arvioinnin suorittamiselle valmiina selkeän rakenteen, joka helpotti työn tehokasta läpivientiä. Kokonaisuutena ISO/IEC 25000 -sarjan standardien hyödyntäminen arvioinnin suorittamisessa tehosti työtä ja paransi arvioinnin laatua.

ISO/IEC 25040 -standardi edellyttää standardia noudattavalta arvioinnilta, että muun muassa laatuvaatimukset ja niihin liittyvät mittaukset dokumentoidaan ja suoritetaan standardin asettamat vaatimukset täyttävästi. Laatumittausten tulee vaatimusten mukaan perustellusti osoittaa arvioitavan ohjelmistotuotteen laatua. Tässä työssä tehdyssä laadun arvioinnissa viimeisen työvaiheen tavoitteena oli testata pelimoottoria käytännössä, jotta laadusta saatiin muodostettua syvällinen ja luotettava arvio. Koska kyse oli pelimoottorista, luonteva testi oli prototyyppipelin tuottaminen pelimoottorin avulla. Testin suorittamiseen oli käytettävissä pääasiassa vain allekirjoittaneen työaikaa.

Prototyyppipeliprojektin toteuttaminen standardin mukaisesti osoittautui kannattamattomaksi. Syvällistä pelin suunnittelu- ja toteutustyötä on yhtäältä työlästä ja hankalaa

1 http://www.tasharen.com/?page_id=140
2 http://www.daikonforge.com/dfgui/

kuvata standardin mukaisesti erillisinä laatuvaatimuksina ja mittauksina. Toisaalta pelimoottori tarjoaa pelikehittäjälle lähes loputtomasti vaihtoehtoja ratkaista vastaantulevia suunnitteluongelmia, eikä pelituotannon tehokkuus tai luodun pelin laatu siksi luotettavasti korreloi pelimoottorin laadun kanssa. Sitä vastoin pelin kehityksen edetessä tehdyt suunnittelupäätökset vaikuttavat hyvin merkittävästi pelin laatuun, pelituotannon tehokkuuteen ja myöhemmin eteen tuleviin suunnitteluongelmiin. Yksittäiseen prototyyppipeliprojektiin liittyvien laatumittausten tuloksista ei siksi voi tehdä objektiivisesti luotettavia päätelmiä pelimoottorin laadusta. Standardin noudattaminen arvioitiin edellisin perusteluin tässä yhteydessä kannattamattomaksi, jonka vuoksi prototyyppipeliprojekti päätettiin toteuttaa ja kuvata epämuodollisemmin.

ISO/IEC 25000 -sarjan standardien hyödyntämisestä on todennäköisesti paljon apua missä tahansa ohjelmistojen laadun arvioinnissa. Standardin noudattaminen voi kuitenkin tässä työssä koetun perusteella olla kannattamatonta, jos kaikki alla olevat ehdot täyttyvät samanaikaisesti.

1. Sovellukselle pyritään tekemään syvällinen käytönaikaisen laadun arviointi, jolloin suunniteltavia ja dokumentoitavia mittauksia on paljon.
2. Sovelluksen käyttötarkoitus on niin monimutkainen, että tarkasteltavan kokonaisjärjestelmän laatu riippuu suurelta osin sovelluksen käyttäjien päätöksistä.
3. Laadun arviointiin ei ole käytettävissä niin suurta otantaa käyttäjiä, että käyttäjistä johtuvat erot laatumittausten tuloksissa voidaan sulkea pois kun arvioidaan sovelluksen laatua.

Yhdessä edelliset kolme kohtaa aiheuttavat väistämättä sen, että standardin vaatimusten noudattamisen työmäärä on suuri, yksittäisen käyttäjän mittaustulokset eivät luotettavasti korreloi tarkasteltavan sovelluksen laadun kanssa ja käytönaikaisessa mittauksessa ei saada mittaustuloksista riittävän suurta otantaa, jotta voitaisiin rajata pois käyttäjistä johtuvat erot mittaustuloksissa ja muodostaa objektiivinen arvio tarkasteltavan sovelluksen laadusta. Tässä tilanteessa standardin noudattamisesta aiheutuva työmäärä jää siis suureksi, mutta standardin seuraamisesta normaalisti saatava hyöty jää saavuttamatta.

Pelimoottoreiden laadun arvioinnin tulokset katselmoitiin allekirjoittaneen ja yrityksen edustajan kesken. Arvioinnin katsottiin onnistuneen hyvin, koska yritykselle löydettiin sopiva pelimoottori nopeasti. Unitya päästiin testaamaan prototyyppipeliprojektissa kattavasti, ja löydösten perusteella yritys aikoo aloittaa pelien kehittämisen Unitylla.

Arvioinnille tuo lisäarvoa se, että kevyellä objektiivisella arvioinnilla saatiin aikaan kattava yleisnäkymä mobiilipelimoottoreihin yrityksen näkökulmasta. Objektiivisella arvioinnilla saatiin valintaa varten rajattua suuri osa markkinoilla olevista pelimoottoreista yritykselle selvästi soveltumattomina pois vaihtoehtojen listalta. Tällä tavoin neljään jäljelle jääneeseen pelimoottoriin voitiin ennen valintaa tutustua paremmin, ja pelimoottorivalinnalle niiden joukosta olikin lopulta selvät perustelut.

7 Yhteenveto

Tutkielmassa tutustuttiin aluksi tietokonepeleihin, mobiilipeleihin ja pelimoottoreihin. Sen jälkeen suoritettiin markkinoilla olevien pelimoottoreiden laadun arviointi pienen peliyrityksen näkökulmasta. Arvioinnin tarkoituksena oli löytää yritykselle parhaiten sopiva pelimoottori, joka olisi toiminnallisesti kattava ratkaisu yrityksen mobiilipelien tuotantoon. Yritykselle laadullisesti tärkeimpiä vaatimuksia pelimoottorille olivat pelituotannon tehokkuus, alustayhteensopivuus, sopeutumiskyky sekä yleinen yhteensopivuus eri sisällöntuotannon työkalujen ja tiedostomuotojen kanssa.

Arvioinnissa oli aluksi mukana kaksikymmentä markkinoilta löytyvää pelimoottoria. ISO/IEC 25000 -sarjan standardeihin perustuvan objektiivisen arvioinnin perusteella niiden joukosta löydettiin neljä yritykselle selkeästi parhaalta vaikuttavaa pelimoottoria. Näihin neljään pelimoottoriin tutustuttiin lähemmin, jonka jälkeen joukosta valittiin jatkotarkasteluun yritykselle kokonaisuutena parhaalta vaikuttanut pelimoottori, Unity. Unityn soveltuvuutta yrityksen tarpeisiin testattiin käytännössä prototyyppipeliprojektissa.

Pelimoottoreiden arviointi pyrittiin suorittamaan ISO/IEC 25000 -sarjan standardien mukaisesti. Standardit määrittelevät selkeästi ja kattavasti ohjelmistojen laadun piirteet sekä prosessin ohjelmistojen laadun arviointiin. Standardien käyttö auttoi huomioimaan pelimoottoriin kohdistuvat laatuvaatimukset kokonaisvaltaisesti, ja valmis arviointiprosessi helpotti työn tehokasta läpivientiä. Standardien hyödyntäminen arvioinnin suorittamisessa paransi arvioinnin laatua.

Prototyyppipeliprojektin toteuttaminen ISO/IEC 25040 -standardin mukaisesti osoittautui kuitenkin kannattamattomaksi. Havaitsimme, että syvällistä suunnittelu- ja toteutustyötä on yhtäältä työlästä ja hankalaa kuvata standardin mukaisesti erillisinä laatuvaatimuksina ja mittauksina. Toisaalta pelimoottori tarjoaa pelikehittäjälle lähes loputtomasti vaihtoehtoja ratkaista vastaantulevia suunnitteluongelmia, jolloin pelituotannon tehokkuus tai luodun pelin laatu ei yksittäisessä projektissa korreloi luotettavasti pelimoottorin laadun kanssa. Objektiivisen laatuarvion tekemiseen vaadittu mittaustarkkuus olisi siksi jäänyt väistämättä projektissa saavuttamatta ja standardin noudattaminen olisi merkinnyt merkittävää lisätyötä ilman merkittävää lisähyötyä.

Yleistäen ISO/IEC 25040 -standardin vaatimusten noudattaminen voi tämän työn yhteydessä koetun perusteella muodostua kannattamattomaksi, jos arvioitavalle sovellukselle pyritään tekemään pienellä käyttäjämäärällä syvällinen käytönaikaisen laadun arviointi ja sovelluksen käyttötarkoitus on niin monimutkainen, että tarkasteltavan kokonaisjärjestelmän laatu riippuu suuresti käyttäjien päätöksistä. Tällöin standardin seuraamiseen käytetystä suuresta työpanoksesta huolimatta ei todennäköisesti saada aikaan objektiivisia mittaustuloksia sovelluksen laadusta.

Prototyyppipeliprojekti päädyttiin edellä mainituista syistä toteuttamaan epämuodollisemmin. Projektin perusteella Unity varmistui yritykselle hyvin sopivaksi pelimoottoriksi. Prototyyppipelin kehittäminen oli Unitylla tehokasta muun muassa nopean kehitys-testi -syklin, laadukkaan ohjelmointirajapinnan, tehokkaan pelin sisällön hallinnan ja tehokkaiden virheiden selvittelymahdollisuuksien vuoksi. Unityn kehitystyökaluissa havaittiin satunnaisia ongelmia prototyyppipelin kehityksen aikana, mutta ne eivät merkittävästi haitanneet työntekoa. Unitylla tuotetun prototyyppipelin suorituskyky, resurssitehokkuus ja luotettavuus olivat hyvällä tasolla sekä Android- että iOS-alustoilla. Unity vaikuttaa olevan laajasti yhteensopiva monenlaisen digitaalisen sisällön kanssa, sen alustatuki on laaja, ja toimittaja vaikuttaa olevan kyvykäs mukautumaan myös uusille mobiilialustoille.

Tehdyn selvityksen perusteella Unity vaikuttaa yritykselle parhaalta pelimoottorivaihtoehdolta ja yritys on päättänyt aloittaa pelien tuotannon Unitylla. Siirtyminen Qt:n käytöstä Unityn käyttöön tulee tehostamaan yrityksen pelituotantoa ja mahdollistaa pelituotannon tärkeimmille mobiilialustoille.

Lähteet

And08 Anderson Eike Falk et al., The Case for Research in Game Engine Architecture. Proceedings of the 2008 Conference on Future Play: Research, Play, Share, ACM, New York, USA, 2008, sivut 228-231.

Ans12 Anson Alexander, Smartphone Usage Statistics 2012. AnsonAlex.com, 24.1.2012. http://ansonalex.com/infographics/smartphone-usage-statistics-2012-infographic/, 31.1.2013.

App06 Apperley Thomas, Genre and game studies: Toward a critical approach to video game genres. Simulation & Gaming, vol. 37 no. 1, maaliskuu 2006.

Bar12 Barnett Michael, Mobile Plays Lead Role in Growth of Gaming, MarketingWeek 19.7.2012. http://www.marketingweek.co.uk/trends/mobile-plays-lead-role-in-growth-of-gaming/4002633.article, 29.1.2013.

Bau98 Baum Dan, 3D Graphics Hardware: Where Have We Been, Where Are We Now, and Where Are We Going. Computer Graphics, vol. 32 no. 1, helmikuu 1998.

Ben11 Benzina Amal et al., Phone-based motion control in VR: analysis of degrees of freedom. CHI EA '11 Extended Abstracts on Human Factors in Computing Systems, ACM, New York, USA, 2011, sivut 1519–1524.

Box13 Boxall Andy, War For Your Pocket: These 5 New Operating Systems Plan to Battle Android and iOS in 2013. Digitaltrends.com, 6.1.2013. http://www.digitaltrends.com/mobile/five-new-mobile-operating-systems-ready-for-launch-in-2013/, 22.9.2013.

Cai61 Caillois Roger, Man, Play and Games. Free Press of Glencoe, Inc., New York, USA, 1961.

Cha10 Chatfield Tom, 7 Ways Games Reward The Brain, TEDGlobal 2010. http://www.ted.com/talks/tom_chatfield_7_ways_games_reward_the_brain.html, 2.1.2013.

Cha13 Chapple Craig, ShiVa game engine maker Stonetrip in liquidation. http://www.develop-online.net/news/43756/ShiVa-game-engine-maker-Stonetrip-in-liquidation, 20.4.2013.

Coc12 Cocotas Alex, CHART: Android Dominates Mobile Platform Market Share, Business Insider. http://www.businessinsider.com/mobile-platform-market-share-2012-8, 29.4.2013.

Del12 Deloura Mark, 2012 Social/Mobile technology survey. Game Developer Magazine, toukokuu 2012.

Dig13 Digia Oyj, Qt 5.1. http://qt.digia.com/Product/Whats-New/Qt-51/, 8.7.2013.

Eis12 Eisemann Elmar et al., Efficient real-time shadows. ACM SIGGRAPH 2012 Courses, ACM, New York, USA, 2012.

Fen03 Fenney Simon, Texture Compression using Low-Frequency Signal Modulation. Graphics Hardware 2003, San Diego, USA, 2003.

Gar90 Garvey Catherine, Play, Harward University Press, Massachusetts, USA, 1990.

Gau12 Gaudiosi John, New Reports Forecast Global Video Game Industry Will Reach $82 Billion By 2017. Forbes.com, 2012. http://www.forbes.com/sites/johngaudiosi/2012/07/18/new-reports-forecasts-global-video-game-industry-will-reach-82-billion-by-2017/, 22.3.2013.

Gre09 Gregory Jason, Game Engine Architecture, A K Peters, Ltd., Massachusetts, USA, 2009.

Hui12 Huiqiang Lu, Design and implementation of three-dimensional game engine. World Automation Congress, IEEE, Puerto Vallarta, Mexico, kesäkuu 2012.

ISO08 ISO/IEC 25012:2008. Software engineering. Software product Quality Requirements and Evaluation (SQuaRE). Data quality model 2008.

ISO11a ISO/IEC 25040:2011. Systems and software engineering. System and software Quality Requirements and Evaluation (SQuaRE). Evaluation process 2011.

ISO11b ISO/IEC 25010:2011. Systems and software engineering. System and software Quality Requirements and Evaluation (SQuaRE). System and software quality models 2011.

Juu03 Juul Jesper, The Game, the Player, the World: Looking for a Heart of Gameness. Level Up: Digital Games Research Conference Proceedings 2003, Utrecht, Alankomaat, 2003, sivut 30-45. Myös http://www.jesperjuul.net/text/gameplayerworld/.

JKS10 Jaeyoung Park, Kee-Eung Kim, Sungho Jo, A POMDP approach to P300-based brain-computer interfaces. Proceedings of the 15th international conference on intelligent user interfaces, ACM, New York, USA, 2010, sivut 1-10.

Kon95 Kontio Jyrki, OTSO: A Systematic Process for Reusable Software Component Selection. University of Maryland Technical Reports, 1995.

Kui07 Kuittinen Jussi et al, Casual games discussion. Proceedings of the 2007 conference on Future Play, ACM, New York, USA, sivut 105-112.

Low13 Lowe Scott, The Best Tablets and Smartphones of CES 2013. IGN 9.1-.2013, http://www.ign.com/articles/2013/01/10/the-best-tablets-and-smartphones-of-ces-2013, 30.1.2013.

MG06 Müller Jens, Grolatch Sergei, Rokkatan: scaling an RTS game design to the massively multiplayer realm. Computers in Entertainment (CIE) – Theoretical and Practical Computer Applications in Entertainment, vol. 4, no. 3, heinäkuu 2006, artikkeli 11.

Mit07 Mittring Martin, Finding next gen: CryEngine 2. ACM SIGGRAPH 2007 courses, ACM, New York, 2007, sivut 97-121.

Neo10 Neogames, Suomen pelitoimialan strategia 2010-2015. http://hermia-fi-bin.directo.fi/@Bin/acf7177790496cbe74ee7299e291da51/1359469641/application/pdf/777120/Pelistrategia%202010-2015.pdf, 1.2.2013.

Neo11 Neogames, The Finnish Games Industry. http://hermia-fi-bin.directo.fi/@Bin/75082041b79c03100d0354d181dc52fc/1359469442/application/pdf/928763/Finnish%20Games%20Industry%202010-2011.pdf, 29.1.2013.

NRL12 Negulescu Matei, Ruiz Jaime, Li Yang, Tap, wipe, or move: attentional demands for distracted smartphone input. Proceedings of the international Working Conference on Advanced Visual Interfaces, ACM, New York, USA, 2012, sivut 173-180.

NSB11 Nealen Andy, Saltsman Adam, Boxerman Eddy, Towards Minimalist Game Design. Proceedings of International Conference on the Foundations of Digital Games, ACM, New York, Usa, 2011, sivut 38-45.

Oin06 Oinonen Vesa, Pelin kehitys hyödyntäen avoimen lähdekoodin komponentteja. Diplomityö, Teknillinen korkeakoulu, Espoo, Suomi, 2006.

Per06 Perry David, Are Games Better Than Life?, TED 2006. http://www.ted.com/talks/david_perry_on_videogames.html, 29.1.2013.

PKS09 Paavilainen Janne, Korhonen Hannu, Saarenpää Hannamari, Pelaaminen matkapuhelimella nyt ja tulevaisuudessa. Pelitutkimuksen vuosikirja 2009, sivut 67-81.

Per12 Perez Sarah, IDC: Android Market Share Reached 75% Worldwide in Q3 2012. http://techcrunch.com/2012/11/02/idc-android-market-share-reached-75-worldwide-in-q3-2012/, 24.3.2013.

Pri10 Priebatsch Seth, The Game Layer on Top of the World, TEDxBoston 2010. http://www.ted.com/talks/seth_priebatsch_the_game_layer_on_top_of_the_world.html, 2.1.2013.

Ros13 Rose Mike, Clash of Clans' 5 keys to success, Gamasutra. http://www.gamasutra.com/view/news/185406/Clash_of_Clans_5_keys_to_success.php, 7.2.2013.

RP08 Rubino Christian, Power John, Level design optimization guidelines for game artists using the epic games: Unreal editor and unreal engine 2. Computers in entertainment, vol 6 issue 4, joulukuu 2008.

Sch11 Schiano Diano et al., A new look at World of Warcraft's social landscape. Proceedings of the 6th International Conference on Foundations of Digital Games, ACM, New York, USA, 2011.

SHK04 Sam-ha Choi, Hee-Dong Chang, Kyung-Sik Kim, Development of Force-Feedback Device for PC-Game using vibration. Proceedings of the 2004 ACM SIGCHI International Conference on Advances in computer entertainment technology, ACM, New York, USA, 2004.

Uni12 Unity Technologies, Unity Reaches One Million Registered Developers. http://www.marketwire.com/press-release/unity-reaches-one-million-registered-developers-1641486.htm, 20.4.2013.

Vid12 Video Game Sales Wiki, Video game industry. http://vgsales.wikia.com/wiki/Video_game_industry, 30.1.2013.

War08 Ward Jeff, What is a Game Engine? GameCareerGuide.com, http://www.gamecareerguide.com/features/529/what_is_a_game_.php, 9.2-.2013.

Wik12 Wikipedia.org, List of game engines. http://en.wikipedia.org/wiki/List_of_game_engines, 27.1.2013.

Wil12 WillyG Productions, Reducing Your Mobile Game Size. http://willyg302.wordpress.com/2012/06/20/reducing-your-mobile-game-size/, 29.4.2013.

Liite 1. Yrityksen toiminnalliset vaatimukset

Alla on dokumentoitu yrityksen korkean tason toiminnalliset vaatimukset. Sadetta etsii pelikehitykseen kokonaisratkaisua, joten kaikki ovat pakollisia vaatimuksia. Lista on laadittu Tomi Tukiaisen ja Vesa Oinosen yhteistyönä.

TVS-01 3-ulotteinen grafiikka. Vaikka useissa peleissä hyödynnetään 2-ulotteista grafiikkaa, yritys tarvitsee tekniset valmiudet tuottaa myös 3-ulotteista piirtoa hyödyntäviä pelejä.

TVS-02 3-ulotteinen fysiikkasimulaatio. Yrityksen aiemmissa ja tulevissa peleissä fysiikan simulointia hyödynnetään voimakkaasti pelitapahtumien mallintamisessa.

TVS-03 Ääni. Äänet ja musiikki ovat olennainen osa lähes mitä tahansa modernia peliä ja uuden teknologia-alustan tulee tukea äänen tuottamista.

TVS-04 3-ulotteinen animaatio. Peliolioiden animointi on useimmissa pelityypeissä välttämätöntä uskottavan visuaalisen ulkoasun aikaansaamiseksi.

TVS-05 Käyttöliittymä. Ikkunoidut käyttöliittymät perusrakenneosineen (painikkeet, teksti ja vaihtuvat kirjasintyylit, tekstinsyöttökentät ja niin edelleen) kuuluvat peleihin kuten mihin tahansa muihinkin sovelluksiin.

TVS-06 Verkkoliikenne. Yrityksen pelit lähettävät tulostietoja palvelimille ja peleissä näytetään pelaajalle mainoksia. Myös moninpelissä vaaditaan verkkoliikenteeseen liittyvää toiminnallisuutta.

TVS-07 Peliohjaimet. Mobiililaitteissa kosketusnäytöt ja kallistussensorit toimivat ohjaimina. Perinteisillä tietokoneilla tyypillisiä ohjaimia ovat näppäimistö, hiiri ja ohjainsauva. Teknisen alustan täytyy tukea olennaisimpien peliohjaimien käyttöä kohdealustoilla.

Liite 2. Laatuvaatimukset pelimoottorille

Seuraavassa liitteessä on dokumentoitu yrityksen laatuvaatimukset uudelle teknologia-alustalle. Laatuvaatimukset on dokumentoitu järjestelmien laadun arviointia varten, ja arvioinnin tarkoitus on parhaan pelimoottorin löytäminen yritykselle. Vaatimukset pohjautuvat ISO/IEC 25010 -standardin laatumalliin sekä ISO/IEC 25030 -standardiin laatuvaatimuksista. Lista on laadittu Tomi Tukiaisen toimesta. Vesa Oinonen on tarkastanut ja hyväksynyt vaatimukset yrityksen puolesta.

Olennaiset sidosryhmät ja intressit:

- **Sadetta:** tuottaa mobiilipelejä järjestelmän avulla. Julkaisee ja myy mobiilipelejä kauppapaikoissa.
- **Sovelluskauppa**: tarjoaa, toimittaa ja myy mobiilipelejä ja pelien sisäisissä kaupoissa (in-app purchase) olevia tuotteita pelaajille. Ei käytä järjestelmää suoraan. Esimerkiksi Google Play ja Apple Store. Intressit: sovelluskaupan sääntöjen noudattamisesta huolehtiminen (mm. asiakkaiden yksityisyyttä ja mainostamista koskevat säännöt, pelien maksimikoko), pelien myynti. Sovelluskaupat tekevät peleille hyväksyntätestausta eri tavoin. Sidosryhmää ei huomioida laatuvaatimusten laatimisessa, koska sidosryhmän vaatimukset on pitkälti huomioitu jo pelaajan vaatimuksissa.
- **Pelaaja**: yrityksen pelien pelaaja. Hankkii pelejä sovelluskaupoista, ostaa pelejä ja pelien sisäisiä tuotteita. Ei käytä järjestelmää suoraan, vaan pelaa järjestelmän avulla tuotettuja pelejä. Intressit: peliviihde, pelin käytön turvallisuus (yksityisyys, tietoturva), mobiililaitteen resurssien käyttö (akun kesto), yksityisyys.
- **Alihankkija**: tuottaa digitaalista sisältöä yrityksen peleihin. Ei käytä järjestelmää suoraan, mutta saattaa joutua esimerkiksi neuvomaan yritystä tuottamansa sisällön integroimisessa peliin. Sidosryhmää ei huomioida laatuvaatimusten laatimisessa, koska sidosryhmän kannalta järjestelmän laatu ei ole olennainen.
- **Mainospalvelu**: hyödyntää pelien sisäistä mainostilaa näyttämällä pelaajalle pelin aikana mainoksia. Sidosryhmää ei huomioida laatuvaatimusten laatimisessa, koska sidosryhmän kannalta järjestelmän laatu ei ole olennainen.

Yrityksen laatuvaatimukset ja vaatimuksiin liitetyt mittaukset:

Käytönaikainen laatu / (pelimoottori, tietokone, Sadetta)	Arvo Sadetta Oy:n pelikehityksessä	Huomiointi arvioinnissa
Vaikuttavuus	**Korkea (järjestelmällä tuotetut pelit)**	**Prototyyppiprojektissa**
Tehokkuus	**Korkea**	**Prototyyppiprojektissa**
Tyydyttävyys	**Keskimääräinen**	
Hyödyllisyys	Keskimääräinen	Prototyyppiprojektissa
Luottamus	Keskimääräinen	Prototyyppiprojektissa
Nautinnollisuus	Matala	Prototyyppiprojektissa
Mukavuus	Matala	Prototyyppiprojektissa
Riskittömyys	**Matala**	
Taloudellisen riskin lieventäminen	Matala	Ei huomioida
Terveydellisen ja turvallisuusriskin lieventäminen	Ei relevantti	Ei huomioida
Ympäristöriskin lieventäminen	Ei relevantti	Ei huomioida
Tarpeen kattavuus	**Korkea**	
Toiminnallinen kattavuus	Korkea	Prototyyppiprojektissa
Joustavuus	Korkea	Prototyyppiprojektissa

Ohjelmistotuotteen laatu (pelimoottori, pelimoottorilla tuotettu peli)	Arvo Sadetta Oy:n pelikehityksessä	Huomiointi arvioinnissa
Toiminnallinen sopivuus	**Korkea**	
Toiminnallinen kattavuus	Korkea	M1 Dokumentoitu tarpeen kattavuus, M2 Evidenssi käytöstä, Prototyyppiprojektissa
Toiminnallinen oikeellisuus	Korkea	Prototyyppiprojektissa
Toiminnallinen hyödyllisyys	Korkea	Prototyyppiprojektissa
Suorituskykytehokkuus	**Korkea (koskee enemmän tuotettua peliä)**	
Aikakäyttäytyminen	Korkea	Prototyyppiprojektissa
Resurssien käyttö	Korkea (huom. myös akun käyttö)	Prototyyppiprojektissa
Kapasiteetti	Keskimääräinen	Prototyyppiprojektissa
Yhteensopivuus	**Keskimääräinen**	
Yhteiselo	Keskimääräinen	Prototyyppiprojektissa
Yhteistoiminta	Keskimääräinen	Prototyyppiprojektissa
Käytettävyys	**Keskimääräinen**	
Hyödyllisyyden tunnistettavuus	Keskimääräinen	Prototyyppiprojektissa
Opittavuus	Keskimääräinen	Prototyyppiprojektissa
Hallittavuus	Keskimääräinen	Prototyyppiprojektissa
Käyttäjän virheiltä suojautuminen	Keskimääräinen	Prototyyppiprojektissa
Käyttöliittymän esteettisyys	Keskimääräinen	Prototyyppiprojektissa
Saavutettavuus	Ei relevantti	Ei huomioida
Luotettavuus	**Korkea (koskee enemmän tuotettua peliä)**	
Maturiteetti	Korkea	M2 Evidenssi käytöstä, Prototyyppiprojektissa
Saatavuus	Ei relevantti	Ei huomioida
Vikasietoisuus	Keskimääräinen	Prototyyppiprojektissa
Palautettavuus	Ei relevantti	Ei huomioida
Turvallisuus	**Ei relevantti**	
Luottamuksellisuus	Ei relevantti	Ei huomioida
Eheys	Ei relevantti	Ei huomioida
Kiistämättömyys	Ei relevantti	Ei huomioida
Vastuullisen tunnistettavuus	Ei relevantti	Ei huomioida
Aitouden tunnistettavuus	Ei relevantti	Ei huomioida
Ylläpidettävyys	**Korkea (koskee tuotettavaa peliä)**	
Modulaarisuus	Korkea	Prototyyppiprojektissa
Uudelleenkäytettävyys	Korkea	Prototyyppiprojektissa
Analysoitavuus	Korkea	Prototyyppiprojektissa
Muokattavuus	Korkea	Prototyyppiprojektissa
Testattavuus	Korkea	Prototyyppiprojektissa
Siirrettävyys	**Korkea**	
Mukatuvuus	Korkea (mukautuvuus uusille mobiilialustoille)	M3 Dokumentoitu alustatuki
Asennettavuus	Keskimääräinen (pelin helppo paketointi)	Prototyyppiprojektissa
Vaihdettavuus	Ei relevantti	Ei huomioida

Pelaajan laatuvaatimukset ja vaatimuksiin liitetyt mittaukset

Käytönaikainen laatu (peli, pelilaite ja pelaaja)	Arvo Pelaajalle	Huomiointi arvioinnissa
Vaikuttavuus	**Korkea (järjestelmällä tuotetut pelit)**	**Prototyyppiprojektissa**
Tehokkuus	**Ei relevantti**	
Tyydyttävyys	**Korkea**	
Hyödyllisyys	Ei relevantti	
Luottamus	Korkea	Prototyyppiprojektissa
Nautinnollisuus	Korkea	
Mukavuus	Korkea	
Riskittömyys	**Ei relevantti**	
Taloudellisen riskin lieventäminen	Ei relevantti	
Terveydellisen ja turvallisuusriskin lieventäminen	Ei relevantti	
Ympäristöriskin lieventäminen	Ei relevantti	
Tarpeen kattavuus	**Ei relevantti**	
Toiminnallinen kattavuus	Ei relevantti	
Joustavuus	Ei relevantti	

Ohjelmistotuotteen laatu (peli)	Arvo Pelaajalle	Huomiointi arvioinnissa
Toiminnallinen sopivuus	**Ei relevantti**	
Toiminnallinen kattavuus	Ei relevantti	
Toiminnallinen oikeellisuus	Ei relevantti	
Toiminnallinen hyödyllisyys	Ei relevantti	
Suorituskykytehokkuus	**Korkea**	**Prototyyppiprojektissa**
Aikakäyttäytyminen	Korkea	
Resurssien käyttö	Korkea	
Kapasiteetti	Korkea	
Yhteensopivuus	**Keskimääräinen**	**Prototyyppiprojektissa**
Yhteiselo	Keskimääräinen	
Yhteistoiminta	Keskimääräinen	
Käytettävyys	**Korkea**	
Hyödyllisyyden tunnistettavuus	Korkea	
Opittavuus	Korkea	
Hallittavuus	Korkea	
Käyttäjän virheiltä suojautuminen	Korkea	
Käyttöliittymän esteettisyys	Korkea	Prototyyppiprojektissa
Saavutettavuus	Ei relevantti (sadettan kohdeyleisö)	
Luotettavuus	**Matala**	
Maturiteetti	Ei relevantti	
Saatavuus	Ei relevantti	
Vikasietoisuus	Ei relevantti	
Palautettavuus	Keskimääräinen (maksetut pelin sisäiset ostokset)	
Turvallisuus	**Matala**	
Luottamuksellisuus	Matala	
Eheys	Ei relevantti	
Kiistämättömyys	Ei relevantti	
Vastuullisen tunnistettavuus	Ei relevantti	
Aitouden tunnistettavuus	Ei relevantti	
Ylläpidettävyys	**Ei relevantti**	
Modulaarisuus	Ei relevantti	
Uudelleenkäytettävyys	Ei relevantti	
Analysoitavuus	Ei relevantti	
Muokattavuus	Ei relevantti	
Testattavuus	Ei relevantti	
Siirrettävyys	**Keskimääräinen**	
Mukatuvuus	Ei relevantti	
Asennettavuus	Korkea	Prototyyppiprojektissa
Vaihdettavuus	Ei relevantti	

Ohjelmistotuotteen sisäisen laadun mittaus (ISO/IEC 25010 Internal Quality) on jätetty arvioinnin ulkopuolelle. Valmiin räätälöimättömän ohjelmistotuotteen hankinnassa sisäisen laadun mittaus ei ole tässä käyttötapauksessa olennainen. Hyvin korkeaa luotettavuutta edellyttävässä käyttötapauksessa sisäisen laadun arviointi voisi olla perusteltua myös valmiille tuotteelle.

Liite 3. Lista pelimoottoreista mobiilikäyttöön

Lista perustuu pitkälti web-sivulta http://mobilegameengines.com/iphone/game_engines 2013 maaliskuun alussa löytyneeseen mobiilipelituotantoon soveltuvien pelimoottorien listaukseen. Listan rajausehtoina on käytetty mainitulla web-sivulla raportoitua 3-ulotteisen piirron tukea sekä iOS- ja Android-tukea. Unreal Engine on lisätty listalle yleistiedon perusteella.

1. Antiryad GX
2. BatteryTech Engine
3. Corona SDK
4. EDGELIB
5. Esenthel Engine
6. GameKit
7. GamePlay (beta)
8. IwGame Engine
9. Marmalade
10. Matali Physics Engine
11. MoSync Mobile
12. NME
13. openFrameworks
14. Orx
15. Proton SDK
16. ShiVa3D Game Engine
17. SIO2 Engine
18. UNIGINE Engine
19. Unity
20. Unreal Engine

Liite 4. Mittaus: dokumentoitu tarpeen kattavuus

Laatuvaatimuksen nimi	**Dokumentoitu tarpeen kattavuus**
Laatupiirre	Toiminnallinen sopivuus (ISO/IEC 25010 Functional suitability)
Alipiirre	Toiminnallinen tarpeen kattavuus (Functional completeness)
Tavoitearvo	”Kyllä”
Tuotteen laadun elinkaaren vaihe	Ohjelmistotuotteen laatu
Mittauksen tarkoitus	Arvioida tuotteen soveltuvuus toiminnallisen vaatimuksen täyttämiseen selvittämällä toiminnallisen kokonaisuuden löytyminen tuotteesta verkosta löytyvän virallisen dokumentaation perusteella. Tarkoitus on sulkea tarkemman arvioinnin ulkopuolelle ne mobiilipelimoottorit, joista jokin olennainen toiminnallinen kokonaisuus puuttuu.
Päätöskriteerit	Lopetetaan mittaus, mikäli toiminnallinen kokonaisuus virallisen verkkodokumentaation perusteella löytyy. Mikäli tukea ei virallisen verkkodokumentaation perusteella löydy (tai siitä ei ylipäänsä ole dokumentaatiossa mainintaa), kirjataan puute ja lopetetaan mittaus.
Indikaattori / visuaalinen esitys	Kyllä / Ei tuotetta koskevan taulukkorivin vastaavassa sarakkeessa.
Mittausfunktio	”Kyllä”, mikäli toiminnallisuus dokumentoidusti tuotteesta löytyy. Muussa tapauksessa ”Ei”.
Käytetyt laatumittauselementit	Toiminnallisuus dokumentoitu tuotteen verkkodokumentaatioon
Mittausmetodi	Toiminnallisuuteen liittyvän dokumentaation etsintä tuotteen verkkodokumentaatiosta
Tietolähteet	Tuotteen virallinen verkkodokumentaatio
Valintakriteeri: evidenssi mittauksen oikeellisuudesta	Toiminnallisen kokonaisuuden löytyminen tuotteesta voidaan korkealla todennäköisyydellä päätellä tarkastamalla tuotteen virallinen verkkodokumentaatio. Mikäli dokumentaatiota toiminnallisesta kokonaisuudesta ei etsittäessä löydy tai sen puuttuminen tuotteesta mainitaan, indikoi se loogisesti päätellen että toiminnallista kokonaisuutta ei tuotteessa ole. Toisaalta toiminnallisen tuen mainitseminen dokumentaatiossa indikoi, että jonkinlainen tuki toiminnallisuudelle melko varmasti löytyy.
Valintakriteeri: evidenssi mittauksen luotettavuudesta	Mikäli toiminnallisuus on dokumentoitu, se korkealla todennäköisyydellä dokumentaation tarkastuksessa löytyy. Dokumentoimaton toiminnallisuus jää tarkastuksessa huomioimatta, joten on mahdollista, että hylkäämme pelimoottorin jossa on dokumentoimatonta toiminnallisuutta. Tämä on tiedostettu ja hyväksytty.
Valintakriteeri: mittauksen kustannus	Alhainen. Dokumentaatiosta etsitään mainintaa toiminnallisen kokonaisuuden täyttymisestä tai täyttymättä jättämisestä. Toiminnallisuutta kuvaavaa dokumentaatiota ei aleta pitkällisesti etsimään tai tarkemmin tutkimaan, koska tarkoitus on vain karsia ominaisuuksiltaan täysin puutteelliset ”pelimoottorit”.
Käyttöskenaariot rooleittain	Laatuarvioinnin suorittaja käyttää mittaustuloksia jatkotutkimuksiin osallistuvien pelimoottoreiden karsintaan vertailun ensimmäisessä vaiheessa. Sadetta Oy:n työntekijät ja muut arvioinnin lukijat voivat käyttää mittaustuloksia pelimoottorituntemuksensa kartuttamiseen.

Pohja: ISO/IEC 25020

Mittauksia suoritettaessa dokumentoitu tarpeen kattavuus tarkastetaan pelimoottorin dokumentaatiosta jokaista liitteessä 1 kuvattua yrityksen toiminnallista vaatimusta vasten.

Liite 5. Mittaus: evidenssi asiakaskäytöstä

Laatuvaatimuksen nimi	Evidenssi asiakaskäytöstä
Laatupiirre	Luotettavuus (ISO/IEC 25010 Security) ja Toiminnallinen sopivuus (Functional suitability)
Alipiirre	Maturiteetti (Maturity) ja Toiminnallinen tarpeen kattavuus (Functional completeness)
Tavoitearvo	8 tai enemmän
Tuotteen laadun elinkaaren vaihe	Ohjelmistotuotteen laatu
Mittauksen tarkoitus	Ensisijaisesti: arvioida tuotteen maturiteettia aiemman käytön perusteella. Toissijaisesti: arvioida tuotteen toiminnallista kattavuutta aiemman käytön perusteella. Kolmanneksi: arvioida tuotteen markkina-asemaa käytön perusteella.
Päätöskriteerit	Huomioidaan vain tuotteen viralliselta sivustolta löytyvät tiedot peleistä ja pelien määristä. Mittaus päätetään arvolla 0, jos mitään mainintaa näistä ei löydy.
Indikaattori / visuaalinen esitys	Tuotteella tuotettujen pelien lukumäärä. 10 mikäli suurempi kuin 10 peliä. Laskettujen pelien nimet kirjataan ylös.
Mittausfunktio	Arvo määräytyy tuotteella tuotettujen pelien lukumäärän mukaan siten, että tuotteen omistajan tekemiä pelejä ei lasketa mukaan. Maksimi 10.
Käytetyt laatumittauselementit	Referenssipelit, jotka on dokumentoitu tuotteen verkkodokumentaatioon
Mittausmetodi	Referenssipeleihin liittyvän dokumentaation etsintä tuotteen verkkodokumentaatiosta. Tämän jälkeen referenssipelin statuksen selvittäminen – onko julkaistu. Kirjataan tuotteella tehtyjen pelien lukumäärä. Lasketaan ainoastaan pelit, joiden julkaisu tai viimeinen versiopäivitys on tapahtunut 1.1.2012 tai myöhemmin. Demoprojekteja, muita sovelluksia kuin pelejä tai toistaiseksi julkaisemattomia pelejä ei lasketa mukaan.
Tietolähteet	Tuotteen virallinen verkkodokumentaatio, viitattujen referenssipelien verkkodokumentaatio
Valintakriteeri: evidenssi mittauksen oikeellisuudesta	Laajan toteutuneen käytön voidaan olettaa parantavan tuotteen luotettavuutta (ISO 25010). Lisäksi peleillä on vaihtelevia vaatimuksia pelimoottorille, joten pelimoottori, jolla on tehty useita pelejä, on todennäköisesti toiminnallisesti kattavampi kuin tuote, jolla on tehty vain vähän pelejä. Useiden referenssipelien löytyminen tuotteen verkkosivustolta on vahva osoitus siitä, että tuotteella on tehty useita pelejä. Mikäli referenssejä ei etsittäessä löydy tai niiden puute mainitaan, indikoi se loogisesti päätellen että tuotteella ei ole tuotettu useita mainitsemisen arvoisia pelejä.
Valintakriteeri: evidenssi mittauksen luotettavuudesta	Mikäli referenssejä on dokumentoitu, ne korkealla todennäköisyydellä dokumentaation tarkastuksessa löytyvät. Dokumentoimattomat referenssit jäävät tarkastuksessa huomioimatta, joten on mahdollista, että referenssejä omaava mutta dokumentoimatta jättänyt, tuote saa alhaisen arvon mittauksessa. Tämä on tiedostettu ja hyväksytty.
Valintakriteeri: mittauksen kustannus	Alhainen. Tuotteen sivustolta etsitään tietoa tuotteella tuotettujen pelien määristä tai suoria viittauksia peleihin. Muualta kuin tuotteen sivustolta ei referenssejä aleta etsimään. Viitattujen pelien status selvitetään pelistä verkossa tarjolla olevin tiedoin (Googlea käytetään etsinnässä).
Käyttöskenaariot rooleittain	Laatuarvioinnin suorittaja käyttää mittaustuloksia jatkotutkimuksiin osallistuvien pelimoottoreiden pisteytykseen vertailun toisessa vaiheessa. Sadetta Oy:n työntekijät ja muut arvioinnin lukijat voivat käyttää mittaustuloksia pelimoottorituntemuksensa kartuttamiseen.

Pohja: ISO/IEC 25020

Liite 6. Mittaus: dokumentoitu alustatuki

Laatuvaatimuksen nimi	**Dokumentoitu alustatuki**
Laatupiirre	Siirrettävyys (ISO 205010 Portability)
Alipiirre	Mukautuvuus (Adaptability)
Tavoitearvo	5 tai enemmän
Tuotteen laadun elinkaaren vaihe	Ohjelmistotuotteen laatu
Mittauksen tarkoitus	Arvioida tuotteen mukautuvuutta uusille alustoille toteutuneen alustatuen laajuuden perusteella.
Päätöskriteerit	Huomioidaan vain tuotteen viralliselta sivustolta löytyvät tiedot tuetuista alustoista. Mittaus päätetään arvolla 0, jos mitään mainintaa näistä ei löydy.
Indikaattori / visuaalinen esitys	Tuotteen tukemien alustojen lukumäärä. Laskettujen alustojen nimet kirjataan ylös.
Mittausfunktio	Arvo määräytyy siten, että muista kuin Android- ja iOS tuesta 1 pistettä. (Android ja iOS tuki löytyy kaikista)
Käytetyt laatumittauselementit	Tuetut alustat, jotka on dokumentoitu tuotteen verkkodokumentaatioon. Huomioidaan myös suunniteltu tuki, jos suunnitelma vaikuttaa selkeältä.
Mittausmetodi	Dokumentoidun alustatuen etsintä tuotteen verkkodokumentaatiosta. Kirjataan tuotteen tukemien alustojen lukumäärä. Lasketaan ainoastaan alustatuki joka on jo olemassa tai selkeästi suunniteltu toteutettavaksi.
Tietolähteet	Tuotteen virallinen verkkodokumentaatio.
Valintakriteeri: evidenssi mittauksen oikeellisuudesta	Laaja toteutunut alustatuki on vahva osoitus siitä, että tuote ja sen toimittaja pystyvät mukautumaan uusille alustoille.
Valintakriteeri: evidenssi mittauksen luotettavuudesta	Tuetut alustat löytyvät korkealla todennäköisyydellä dokumentaation tarkastuksessa. Dokumentoimaton alustatuki jää tarkastuksessa huomioimatta, joten on mahdollista, että alustatukea omaava mutta dokumentoimatta jättänyt tuote saa alhaisen arvon mittauksessa. Tämä on tiedostettu ja hyväksytty. Toisaalta on myös mahdollista että sama alusta on mainittu tuettuna useilla eri nimillä ja mittauksen suorittaja ei tilannetta tunnista, jolloin mittaustulos saattaa olla liian korkea. Virheen todennäköisyyttä pidetään pienenä ja mahdollisia haittavaikutuksia hyväksyttävänä.
Valintakriteeri: mittauksen kustannus	Alhainen. Tuotteen sivustolta etsitään tietoa, eikä muita lähteitä aleta tutkimaan.
Käyttöskenaariot rooleittain	Laatuarvioinnin suorittaja käyttää mittaustuloksia jatkotutkimuksiin osallistuvien pelimoottoreiden pisteytykseen vertailun toisessa vaiheessa. Sadetta Oy:n työntekijät ja muut arvioinnin lukijat voivat käyttää mittaustuloksia pelimoottorituntemuksensa kartuttamiseen.

Pohja: ISO/IEC 25020

Liite 7. Mittaustulokset: toiminnallinen kattavuus

	TVS-01 3-ulotteinen grafiikka	TVS-02 3-Ulotteinen fysiikka	TVS-03 Ääni	TVS-04 3-Ulottinen animaatio	TVS-05 Käyttöliittymä	TVS-06 Verkkoliikenne	TVS-07 Peliohjaimet
Antiryad GX	Kyllä	Kyllä	Kyllä	Kyllä	Kyllä	Kyllä	Kyllä
BatteryTech Engine	Kyllä	Ei	Kyllä	Kyllä	Kyllä	Ei	Kyllä
Corona SDK	Ei	Ei	Kyllä	Ei	Kyllä	Kyllä	Kyllä
EDGELIB	Kyllä	Ei	Kyllä	Kyllä	Kyllä	Kyllä	Kyllä
Esenthel Engine	Kyllä	Kyllä	Kyllä	Kyllä	Kyllä	Kyllä	Kyllä
GameKit	Kyllä	Kyllä	Kyllä	Kyllä	Ei	Ei	Ei
GamePlay (beta)	Kyllä	Kyllä	Kyllä	Kyllä	Kyllä	Ei	Kyllä
IwGame Engine	Ei	Ei	Kyllä	Ei	Kyllä	Ei	Kyllä
Marmalade	Kyllä	Kyllä	Kyllä	Kyllä	Kyllä	Kyllä	Kyllä
Matali Physics Engine	Kyllä	Kyllä	Kyllä	Kyllä	Ei	Ei	Ei
MoSync Mobile	Ei	Ei	Kyllä	Ei	Kyllä	Ei	Kyllä
NME	Ei	Ei	Kyllä	Kyllä	Ei	Kyllä	Kyllä
openFrameworks	Kyllä	Ei	Kyllä	Ei	Ei	Kyllä	Kyllä
Orx	Ei	Ei	Kyllä	Ei	Ei	Ei	Kyllä
Proton SDK	Kyllä	Ei	Kyllä	Ei	Kyllä	Ei	Kyllä
ShiVa3D Game Engine	Kyllä	Kyllä	Kyllä	Kyllä	Kyllä	Kyllä	Kyllä
SIO2 Engine	Kyllä	Kyllä	Kyllä	Kyllä	Kyllä	Kyllä	Kyllä
UNIGINE Engine	Kyllä	Kyllä	Kyllä	Kyllä	Kyllä	Kyllä	Kyllä
Unity	Kyllä	Kyllä	Kyllä	Kyllä	Kyllä	Kyllä	Kyllä
Unreal Engine	Kyllä	Kyllä	Kyllä	Kyllä	Kyllä	Kyllä	Kyllä

Mittausmuistiinpanot on jätetty tästä liitteestä pois tilan puutteen vuoksi. Mittaukset suoritettiin 1.4. - 5.4.2013. Tarkemmat tiedot on toimitettu Sadetta Oy:lle.

Liite 8. Mittaustulokset: evidenssi käytöstä

	Mittaustulos käytöstä	Tarkastetut referenssipelit & kommentit (selvitykset tehty 7.4.2013)
Antiryad GX	0	Viimeiset asiakkaiden pelit julkaistu 2011.
Esenthel Engine	3	Lucius, Ineisis Online (oli hetken levityksessä), Super Hero Online (kiinalainen). Vain 1 oli selkeä referenssi.
Marmalade	9	Catch the Monkey, Tonga, Ghost Puzzle, Grave Defense HD, Draw Something, Cut the Rope, Need for Speed Shift, Backbreaker Vengeance 2, Golf Battle 3d
ShiVa3D Game Engine	10	Babel Rising, Non Flying Soldiers, Pinball Yeah!, Monkey Drum Deluxe, F1 Paper Racing, Block Rockin', Dust: Offroad Racing, Bob Orange, Chain3D, uDribble Basketball
SIO2 Engine	0	Referenssejä ei löytynyt viralliselta sivustolta. Foorumilta löytyi vuoden 2009 tienoille asti.
UNIGINE Engine	2	Tryst (win32), Demolicious (iOS)
Unity	10	Year Walk (iOS), Dead Trigger 2 (iOS, Android), The Room (iOS), Bad Piggies (iOS, Android, PC, Mac), Dungeonland (PC), Ravensword: Shadowlands (iOS, Android), Undead Slayer (iOS, Android), Momonga Pinball Adventure (iOS), Vampire Season – Monster Defense (iOS), Endless Space (PC, Mac)
Unreal Engine	10	Primal Carnage (PC), Dishonored (Xbox 360, PC, Playstation 3), Hawken (PC Public Beta), Transformers – Fall of Cybertron (Xbox 360, PS3), XCOM – Enemy Unknown (PC, Xbox 360, PS3), Borderlands 2 (PC, Xbox 360, PS3), Lollipop Chainsaw (Xbox 360, PS3), The Nightworld (iOS), Asura's Wrath (Xbox 360, PS3), Mass Effect 3 (Xbox 360, PS3, PC)

Liite 9. Mittaustulokset: dokumentoitu alustatuki

	Mittaustulos alustatuesta	Tuetut alustat, selvitykset tehty 14.4.2013.
Antiryad GX	4	Linux, Windows, OS-X, iOS, Android, PS3
Esenthel Engine	2	Windows, OS-X, Android, iOS
Marmalade	5	iOS, Android, BlackBerry PlayBook OS, bada (2.0), Windows, OS-X, LG Smart TV
ShiVa3D Game Engine	17	Web Player (x86/64), Windows, OS-X, Linux, Flash, Chrome NaCl, iOS, Android, Amazon Kindle, OUYA Game Console, WebOS, RIM Blackberry, Windows Phone 7, Windows Phone 8, Xbox 360, Nintendo Wii, Playstation 3, Playstation Portable, Playstation Vita. HUOM: voi olla päällekkäisyyksiä.
SIO2 Engine	2	iOS, Android, OS-X, Windows
UNIGINE Engine	4	Windows, Linux, OS-X, PS3, iOS, Android
Unity	8	iOS, Android, Windows, OS-X, Linux, Flash, Chrome NaCl, Playstation 3, Xbox, Wii U
Unreal Engine	7	iOS, Windows, Flash, Android, Xbox 360, OS-X, PlayStation 3, Playstation Vita, Wii U

Liite 10. Laadun kokonaisarviointi mittausten perusteella

	Mittaustulos käytöstä	Mittaustulos alustatuesta	Laatupisteet yhteensä
Antiryad GX	0	4	**4**
Esenthel Engine	3	2	**5**
Marmalade	9	5	**14**
ShiVa3D Game Engine	10	17	**27**
SIO2 Engine	0	2	**2**
UNIGINE Engine	2	4	**6**
Unity	10	8	**18**
Unreal Engine	10	7	**17**

Liite 11. Unityn laadun arviointi prototyyppiprojektin perusteella yrityksen näkökulmasta

Käytönaikainen laatu / (pelimoottori, tietokone, Sadetta)	**Subjektiivinen arvio prototyyppiprojektin perusteella**
Vaikuttavuus	**Hyvä: Unityllä voi esimerkkiprojektin perusteella tehdä erittäin näyttäviä pelejä.**
Tehokkuus	**Hyvä: Pelikehitys Unitylla on erittäin tehokasta**
Tyydyttävyys	**Hyvä**
Hyödyllisyys	Hyvä
Luottamus	Keskimääräinen: suurin ongelma oli MonoDevelop-ohjelmointityökalun satunnainen kaatuilu (noin kerran päivässä). Lisäksi havaittiin satunnaisia ongelmia asennuspakettien rakennusprosessissa sekä Unity Editorin kaatumisia harvakseltaan. Android-profiloinnin jälkeen käyttöliittymä jumittui aina kun pelin sammutti Android-laitteessa, tai laitteen irrotti PC:sta.
Nautinnollisuus	Hyvä
Mukavuus	Hyvä: kaikki toiminnot oli helposti saatavilla. Hieman häiritsevää oli käyttöliittymän jumittuminen pitkien asennuspakettien rakennusprosessien aikana.
Riskittömyys	**Ei huomioitu**
Taloudellisen riskin lieventäminen	Ei huomioitu
Terveydellisen ja turvallisuusriskin lieventäminen	Ei huomioitu
Ympäristöriskin lieventäminen	Ei huomioitu
Tarpeen kattavuus	**Hyvä**
Toiminnallinen kattavuus	Hyvä: kaikki prototyyppipelissä tarvittavat toiminnallisuudet oli helposti saatavilla skripteille tarjotusta rajapinnasta
Joustavuus	Hyvä: hyvin monimutkaisenkin toiminnallisuuden toteuttaminen onnistuu C#:lla, Unity Scriptillä ja Shader Labin avulla.

Ohjelmistotuotteen laatu (pelimoottori, pelimoottorilla tuotettu peli)	**Subjektiivinen arvio prototyyppiprojektin perusteella**
Toiminnallinen sopivuus	**Hyvä**
Toiminnallinen kattavuus	Hyvä
Toiminnallinen oikeellisuus	Hyvä: muutama bugi kirjastofunktioiden toiminnassa havaittiin, mutta pääosin laatu oli hyvä. Esimerkiksi Raycastin havaittiin tunnistavan pois päältä kytketty taso (plane), vaikka näin ei pitäisi tapahtua.
Toiminnallinen hyödyllisyys	Hyvä
Suorituskykytehokkuus	**Hyvä**
Aikakäyttäytyminen	Hyvä: peli toimi yllättävän tehokkaasti Androidilla
Resurssien käyttö	Hyvä: prototyyppipelin virrankulutus saatiin laskemaan hyväksyttävälle tasolle kun virkistystaajuuden tavoitearvo asetettiin arvoon 30/sek (oletus 60/sek), pelin asennuspaketti oli kooltaan odotusten mukainen.
Kapasiteetti	Hyvä
Yhteensopivuus	**Hyvä**
Yhteiselo	Hyvä: ei havaittu mitään ongelmia
Yhteistoiminta	Hyvä: äänien, musiikin, 3d-mallien ja 2d-grafiikan tuonti projektiin oli ongelmatonta ja erittäin sujuvaa
Käytettävyys	**Hyvä**
Hyödyllisyyden tunnistettavuus	Hyvä: dokumentaatio ja käyttöliittymä tekivät hyödyllisyyden tunnistettavuudesta varsin helppoa
Opittavuus	Hyvä
Hallittavuus	Hyvä
Käyttäjän virheiltä suojautuminen	Hyvä: virheet ilmoitettiin Unity Editorin käyttöliittymässä. Virheet peliolioiden määrityksissä myös usein näkee suoraan Unity Editorissa, koska määriteltävät pelioliot näkyvät editorissa kuten ne tulevat näkymään itse pelissäkin.
Käyttöliittymän esteettisyys	Hyvä
Saavutettavuus	Ei huomioitu
Luotettavuus	Keskimääräinen
Maturiteetti	Hyvä
Saatavuus	Ei huomioitu
Vikasietoisuus	Keskimääräinen: satunnaisia Unity Editorin kaatumisia ja pelin asennuspaketin rakennusprosessin kaatumisia havaittiin projektin aikana. Toisaalta peli ei kaadu skripteissä tapahtuviin poikkeuksiin, mikä on etu.
Palautettavuus	Hyvä: projektit voi pitää keskitetyssä versionhallinnassa ja Unity Editorin voi tarvittaessa asentaa uudelleen
Turvallisuus	**Ei huomioitu**
Luottamuksellisuus	Ei huomioitu
Eheys	Ei huomioitu
Kiistämättömyys	Ei huomioitu
Vastuullisen tunnistettavuus	Ei huomioitu
Aitouden tunnistettavuus	Ei huomioitu
Ylläpidettävyys	
Modulaarisuus	Hyvä: toiminnallisuuksien komponenttirakenne on modulaarinen
Uudelleenkäytettävyys	Hyvä: toiminnallisuuksien komponenttirakenne edistää uudelleenkäyttöä
Analysoitavuus	Hyvä: komponenttien toimintaa on erittäin helppo analysoida Unity Editorissa, koska sisäisten muuttujien arvot on nähtävissä ja muutettavissa pelin ajon aikana. Ajonaikaiset poikkeukset eivät myöskään johda pelin kaatumiseen vaan niistä tulee virheilmoitukset konsoliin. Virheilmoitusta tuplaklikkaamalla pääsee poikkeuksen aiheuttaneelle koodiriville ja stacktrace on konsolissa näkyvissä. Unity Pro:n mukana tulee myös profilointityökalu, jossa prosessorin ja grafiikkaprosessorin suoritusaikojen kulutusta pystyy analysoimaan.
Muokattavuus	Hyvä: toiminnalliset komponentit pysyivät ainakin prototyyppiprojekteissa kohtuullisen kokoisina ja erillisinä toisistaan. Muutosten vaikutukset rajoittuvat näin pienelle alueelle.
Testattavuus	Hyvä: Unity tukee testausta nopealla kehitys-testaus-syklillä ja läpinäkyvyydellä skriptien sisäisiin muuttujiin.
Siirrettävyys	
Mukatuvuus	Ei huomioitu
Asennettavuus	Hyvä (peli paketoidaan ja asentuu Androidille automaattisesti, Unity Editor ja sen päivitykset on helppo asentaa PC:lle)
Vaihdettavuus	Ei huomioitu

Liite 12. Unityn laadun arviointi prototyyppiprojektin perusteella pelaajan näkökulmasta

Käytönaikainen laatu (peli, pelilaite ja pelaaja)	**Subjektiivinen arvio prototyyppiprojektin perusteella**
Vaikuttavuus	**Hyvä**
Tehokkuus	**Ei huomioitu**
Tyydyttävyys	**Hyvä**
Hyödyllisyys	Ei huomioitu
Luottamus	Hyvä: tuotettu prototyyppipeli ei kaatunut Androidilla kertaakaan. Muutama pelin kaatuminen havaittiin PC:lla.
Nautinnollisuus	Hyvä
Mukavuus	Hyvä
Riskittömyys	**Ei huomioitu**
Taloudellisen riskin lieventäminen	Ei huomioitu
Terveydellisen ja turvallisuusriskin lieventäminen	Ei huomioitu
Ympäristöriskin lieventäminen	Ei huomioitu
Tarpeen kattavuus	**Ei huomioitu**
Toiminnallinen kattavuus	Ei huomioitu
Joustavuus	Ei huomioitu

Ohjelmistotuotteen laatu (peli)	**Subjektiivinen arvio prototyyppiprojektin perusteella**
Toiminnallinen sopivuus	Ei huomioitu
Toiminnallinen kattavuus	Ei huomioitu
Toiminnallinen oikeellisuus	Ei huomioitu
Toiminnallinen hyödyllisyys	Ei huomioitu
Suorituskykytehokkuus	**Hyvä**
Aikakäyttäytyminen	Hyvä
Resurssien käyttö	Hyvä
Kapasiteetti	Hyvä
Yhteensopivuus	**Hyvä**
Yhteiselo	Hyvä
Yhteistoiminta	Hyvä
Käytettävyys	**Keskimääräinen**
Hyödyllisyyden tunnistettavuus	Ei huomioitu
Opittavuus	Ei huomioitu
Hallittavuus	Ei huomioitu
Käyttäjän virheiltä suojautuminen	Ei huomioitu
Käyttöliittymän esteettisyys	Keskimääräinen, Unityn käyttöliittymäluokat eivät tarjonneet erityisen hyviä työkaluja käyttöliittymien toteuttamiseen. Esimerkiksi tekstin skaalaus animoidusti osoittautui hankalaksi ongelmaksi tarjottujen rajapintojen avulla. Unity on tiedostanut ongelman ja on kehittämässä uutta käyttöliittymäkirjastoa.
Saavutettavuus	Ei huomioitu
Luotettavuus	Hyvä
Maturiteetti	Ei huomioitu
Saatavuus	Ei huomioitu
Vikasietoisuus	Hyvä, ajonaikaiset poikkeukset skripteissä eivät kaada koko peliä
Palautettavuus	Ei huomioitu
Turvallisuus	**Ei huomioitu**
Luottamuksellisuus	Ei huomioitu
Eheys	Ei huomioitu
Kiistämättömyys	Ei huomioitu
Vastuullisen tunnistettavuus	Ei huomioitu
Aitouden tunnistettavuus	Ei huomioitu
Ylläpidettävyys	**Ei huomioitu**
Modulaarisuus	Ei huomioitu
Uudelleenkäytettävyys	Ei huomioitu
Analysoitavuus	Ei huomioitu
Muokattavuus	Ei huomioitu
Testattavuus	Ei huomioitu
Siirrettävyys	**Hyvä**
Mukatuvuus	Ei huomioitu
Asennettavuus	Hyvä: standardi asennuspaketti joka on odotetun kokoinen. Unity näyttää myös tukevan asennuspaketin osittamista, mutta tätä ei testattu.
Vaihdettavuus	Ei huomioitu

Liite 13. Arviointisuunnitelma

Allekirjoittanut suorittaa ensimmäisen ja toisen vaiheen laatumittaukset aikavälillä 1.4.2013 - 14.4.2013 ja käyttää mittausten suorittamiseen työaikaa tarpeen mukaan. Pelimoottorin valinta prototyyppivaiheeseen suoritetaan yhdessä yrityksen edustajan kanssa 15.4.2013 - 20.4.2013.

Prototyyppipeli tuotetaan valitulla pelimoottorilla pääosin allekirjoittaneen toimesta, mutta myös yrityksen edustaja osallistuu työajallaan prototyyppiprojektiin. Prototyyppiprojektissa pelikehityksen testiympäristöinä toimivat lähtökohtaisesti Windows 8 koneet. Peliprototyyppiä pyritään testaamaan Google Nexus 7:ssa, Samsung Galaxy S2:ssa ja iPhone 5:ssa. Prototyyppiprojektiin liittyvät mittaukset pyritään suorittamaan valmiiksi 31.5.2013 mennessä.

Arviointiin osallistuvat henkilöt ja arvioinnin suorittaneiden pätevyys

Arviointiprosessiin liittyvien tehtävien läpivienti ja arviointiraportin laatiminen ovat yksinomaan allekirjoittaneen suorittamia, ellei muuta erikseen mainita. Pelialan kokemusta allekirjoittaneelle on kertynyt harrastuneisuuden perusteella sekä seuraavista kaupallisista projekteista.

- Emancy: C-kielellä toteutettu PC-peli. Kehitys 1997-1999, julkaisu yksityishenkilöinä 1999.
- Bowling Evolution: C++-kielellä toteutettu PC-peli. Kehitys 2004-. Ensimmäinen julkaisu yksityishenkilöinä 2005. Kaupallinen julkaisu 2009. Pelin kehityksessä käytettiin useita avoimen lähdekoodin komponentteja hyödyksi (mm. Irrlicht ja Open Dynamics Engine).
- Motocross Masters: Qt:lla toteutettu mobiilipeli. Kehitys 2011-. Kaupallinen julkaisu 2011. Pelin kehityksessä käytettiin avoimen lähdekoodin komponentteja, joita oli saatavissa Qt-alustalle.
- Downhill Champion: Qt:lla toteutettu mobiilipeli. Kehitys 2012-. Kaupallinen julkaisu 2012. Pelin kehityksessä käytettiin avoimen lähdekoodin komponentteja, joita oli saatavissa Qt-alustalle.