



MOBIILIPELIN SUUNNITTELU JA JULKAISEMINEN WINDOWS PHONE 8 -ALUSTALLE

Erkka Hurske

Opinnäytetyö
Elokuu 2015
Tietojenkäsittely

TAMPEREEN AMMATTIKORKEAKOULU
Tampere University of Applied Sciences

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietojenkäsittely

HURSKE, ERKKA:

Mobiilipelin suunnittelu ja julkaiseminen Windows Phone 8 -alustalle

Opinnäytetyö 35 sivua
Joulukuu 2015

Peliteollisuus on maailman nopeimmin kasvava viihdeala. Kehittäjien on mahdollista nykyisten työkalujen avulla tehdä peli hyvin pienillä resursseilla. Aloittava pelinkehittäjä ei välttämättä tiedä, mitä kaikkea pelinkehitykseen sisältyy. Tämä opinnäytetyö on toteutettu tavoitteenaan antaa aloittavalle kehittäjäryhmälle käsitys siitä, mitä pelinkehittäminen on ja mitä kaikkea siinä tulisi ottaa huomioon. Tarkoituksena oli suunnitella, toteuttaa ja julkaista peli Windows Phone 8 -alustalle sekä tarkastella prosessiin liittyviä seikkoja teorian tasolla.

Opinnäytetyössä suunniteltiin, toteutettiin ja julkaistiin yksinkertainen mobiilipeli. Työssä käsitellään teorian tasolla sitä, mitä vaiheita kuuluu pelin yleiseen suunnitteluun ja hyvään sisältösuunnitelmaan. Työssä kerrotaan oman pelitoteutuksen suunnittelu. Itse toteutus tehtiin Unity-pelimoottorilla, jonka käyttöä ja ominaisuuksia käsitellään laajemminkin. Ohjelmointia käsitellään laajahkosti eikä lukijalta odoteta perehtymistä ohjelmointiin. Pelin julkaisua käsittelevässä osiossa kerrotaan Windows-kehittäjätilin luomisesta sekä siitä, mitä seikkoja oman pelin julkaisemisessa Windows Kaupassa tulee ottaa huomioon.

Lopputuloksena opinnäytetyössä on toimiva ja lähes julkaisuvalmis mobiilipeli Windows Phone 8 -alustalle. Pelistä on jouduttu karsimaan muutama suunnitelmassa kerrottu ominaisuus. Näiden ominaisuuksien sekä äänien puuttuminen ovat syynä pelin julkaisematta jättämiseen. Opinnäytetyöstä saa kuitenkin kuvan siitä, mitä pelinkehittäminen on ja mitä siinä kannattaa ottaa huomioon. Lisäksi voidaan todeta, että toimivan pelin tekeminen ei vaadi monijäsenistä kehitystiimiä vaan se voi onnistua yhdeltäkin henkilöltä.

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree Programme in Business Information Systems

HURSKE, ERKKA:

Designing and Publishing a Mobile Game for Windows Phone 8

Bachelor's thesis 35 pages
December 2015

Game industry is nowadays the fastest growing part of the entertainment industry. With modern tools the developers are able to make a game with very little resources. It may not be clear to everyone interested in game development what you must take into consideration when making a game. This thesis was made to answer this question. The purpose was to design, develop and publish a game for Windows Phone 8 and to view different aspects of the game developing process from a theoretical perspective.

A simple mobile game was designed, developed and published during this thesis project. The thesis answers the following questions: what steps are involved in good game design, and what does the game design document contain? The thesis also includes the design process of the game project under examination. The game itself was made using the Unity game engine, whose features and usage are observed more closely. A developer account is needed to publish a Unity game. The thesis tells how the account is created and how the game can be published.

The result was a playable and almost ready game. Some features are still missing from the game, which prevents publication of the game at this stage. These features keep the game from being published. The reader will get a general view on what needs to be taken into consideration when developing a game. Making a functional game doesn't need a big development team, it is possible to do it with just one person and a very small budget.

Keywords: unity, game development, windows phone 8

SISÄLLYS

1	JOHDANTO.....	5
2	PELIKEHITYS YLEISESTI.....	6
2.1	Historia.....	6
2.2	Pelinkehitys Suomessa.....	7
2.3	Tulevaisuus	7
3	PELIN SUUNNITTELU	9
3.1	Game Design Document (GDD).....	9
3.2	Kehitystiimi ja -työkalut	10
3.3	Prototyypin luominen	11
3.4	Oman projektin suunnittelu	11
3.4.1	Suunnittelu	11
3.4.2	Pelin kulku	12
4	PELIN TOTEUTUS	14
4.1	Pelimoottori	14
4.2	Unity	15
4.2.1	Yleistä	15
4.2.2	GameObject.....	18
4.2.3	Ohjelmointi	19
4.3	Muut kehitystyökalut	21
4.4	Testaus	21
4.5	Oman pelin toteutus	21
4.5.1	TouchController.cs.....	22
4.5.2	TriggerManager.cs	23
4.5.3	DataManager.cs.....	24
4.5.4	Valikot ja animaattori.....	24
4.5.5	Ongelmat toteutuksessa.....	27
5	PELIN JULKAISU.....	28
5.1	Kehittäjätili	28
5.2	Oman toteutuksen julkaisu.....	29
6	YHTEENVETO	33
	LÄHTEET	34

1 JOHDANTO

Pelit ja pelinkehitys ovat Suomessa tällä hetkellä kiinnostavampia kuin koskaan. Pelialan yrityksiä on ympäri Suomen yli kaksi sataa (Neogames: Alan toimijat 2015). Ala onkin nopeimmin kasvava viihdeala, jonka myynnin on arvioitu olevan jopa 65 miljardin dollarin luokkaa. Suomessa peliteollisuuden liikevaihdon vuonna 2014 arvellaan olevan 1800 miljoonan euron luokkaa, mikä on kaksi kertaa enemmän kun vuonna 2013 arvioitu summa. Peliteollisuuden tuotteista lähes kaikki päätyy vientiin, joten menestys on hyväksi myös Suomen kansantaloudelle ja viennille (Neogames: Tietoa toimialasta 2015). Suomi on saanut osakseen muutaman suuren menestystarinan peliteollisuudesta. Tekijät, kuten Supercell ja Rovio, ovatkin omalta osaltaan olleet siivittämässä Suomen peliteollisuutta nousuun ja suuren yleisön tietoisuuteen.

Suomalaisilla menee erityisen lujaa mobiilipelimarkkinoilla ja uusia tekijöitä tulee uusia jatkuvasti niin yksityisten kuin yritystenkin puolelle. Kilpailu on kovaa ja koskaan ei tiedä mistä ja milloin se seuraava uusi hittipeli ilmestyy. Pelinkehitys työkalujen ja laitteiden kehittyessä on kenen tahansa mahdollista aloittaa kehittämään omaa näkemystään tuosta seuraavasta hitistä. Aikaisemmin operaattorien ja eri julkaisijoiden hoitaessa mobiilipelien levitystä ja myyntiä, oli itsenäisten kehittäjien hyvin vaikea tuoda omaa peliään markkinoille. Lisäksi nämä ulkopuoliset toimijat ottivat suuren osan saaduista tuotoista. Nykyiset digitaaliset alustakohtaiset julkaisukanavat ovat auttaneet itsenäisiä pelinkehittäjiä tuomaan tuotteensa markkinoille hyvin pienillä julkaisukuluilla. Ilman erillistä julkaisija osapuolta ovat myös saadut tuotot jääneet kannattaviksi.

Pelialan ollessa tällä hetkellä todella kiinnostava, riittää alalle pyrkijöitäkin runsaasti. Olen osaltani huomannut, että kaikilla pelinkehityksestä kiinnostuneilla ei välttämättä löydy käsitystä mitä kaikkea kehitystyö pitää sisällään. Niinpä tämä opinnäytetyö onkin tehty tuohon avuksi. Selventämään pelinkehityksen vaiheita ja sitä millä tavoin voidaan toteuttaa ja julkaista peli, itse suosimalleni, Windows Phone -mobiilialustalle.

2 PELINKEHITYS YLEISESTI

Pelinkehitys on ohjelmistokehityksen alalaji, jonka lopputuotteena on videopeli. 2000-luvulla peliteollisuus on ollut tauotta nopeimmin kasvava viihdeteollisuuden ala. Vuonna 2014 peliteollisuuden maailmanlaajuisen arvon katsottiin olevan 100 miljardin euron luokkaa. (Hiltunen, K. Latva, S. Kaleva, P. 2013, 6)

Kehittäjiltä vaaditaan monia eri taitoja, kuten taiteilijoiden visuaalista silmää ja käsikirjottajan mielikuvitusta. Tärkeässä osassa kehityksessä on myös tekninen osaaminen, jotta suunniteltu visio saadaan toteutettua. (Crosby, T. 2008) Pelinkehittäjä terminä voi tarkoittaa montaa eri pelin tekemiseen tarvittavaa osa-aluetta, kuten esimerkiksi tuottaja, ohjelmoija, taiteilija tai suunnittelija. Nämäkin osa-alueet ovat jaettavissa vielä tarkempiin osiin. Ohjelmoija voi olla erikoistunut pelimoottori-, verkko- tai käyttöliittymäohjelmointiin ja taiteilija konseptitaiteeseen, animaatioihin tai 3D-malleihin.

2.1 Historia

Videopelien historia voidaan katsoa alkavaksi jo 1950-luvulta, jolloin niitä kehitettiin pääasiassa erilaisiin tutkimustarkoituksiin. Suuremman yleisön tietoisuuteen pelit tulivat 1970-luvulla, kun maksulliset peliautomaatit valtasivat markkinoita. Seuraavina vuosikymmeninä myös kuluttajat alkoivat saada omia kotikäyttöön tarkoitettuja tietokoneita, kuten vuonna 1982 julkaistut Commodore 64 ja ZX Spectrum. Tehot näissä ensimmäisissä tietokoneissa olivat vaatimattomia, mutta mahdollistivat silti pelien kehittämisen harrastuksena. Tästä lähtien pelien kiinnostus on vain kasvanut entisestään.

Kiinnostus on tuonut markkinoille monia eri konsoli sukupolvia, kun on siirrytty 8-bittisestä Nintendo Entertainment Systemistä (NES) aina nykyisiin Microsoftin Xbox Oneen ja Sonyn Playstation 4:een. Tällä välillä on pelien saralla koettu melkoinen muutos. 90-luvulla koettiin merkittävä innovaatio, kun siirryttiin bittigrafiikasta käyttämään 3D-grafiikkaa. Samoihin aikoihin uutta tuulta purjeisiin toivat julkaisijoiden yhteistyö, mikä mahdollisti myös suuremmat budjetit ja suuremmat kehittäjä ryhmät. Myös tietokoneiden

tehot ja mahdollisuudet kasvoivat, joka johti entistä hienompiin ja kunnianhimoisimpiin peleihin. Tämä kehitys on jatkunut aina näihin päiviin asti.

2000-luvulla kuviin astui myös mobiili pelaaminen, mikä jo vuonna 2003 oli ylittänyt miljardin dollarin rajan. Pelikonsoli markkinat saivat kovan kilpailijan kun Apple ja Google avasivat omille puhelimillensa tarkoitetut sovelluskaupat, joista kykeni hankkimaan sovelluksia ja pelejä edullisella hintalapulla varustettuna. Yksi kaikkien aikojen suosituimmista mobiilipeleistä on suomalaisen Rovio Entertainmentin tekemä Angry Birds, jota myytiin ensimmäisen puolen vuoden aikana yli 6,5 miljoonaa kappaletta Applen iPhoneilla, iPadilla ja iPadTouchilla (Takahashi, D. 2010). Älypuhelimet toimivatkin aloituspaikkana monille indie-kehittäjille niiden suosion ja helpon lähestyttävyyden vuoksi.

2.2 Pelinkehitys Suomessa

Suomessa menestyneimmät peliyritykset ovat keskittyneet pääasiassa mobiilipeleihin. Näistä toimijoista menestyneimpiä ovat edellä mainitun Rovio Entertainmentin lisäksi Supercell Clash of Clans -pelillään. Suomesta löytyy myös maailmalla menestyneitä pelintekijöitä, jotka ovat keskittyneet muille alustoille. Menestys on tuonut mukanaan kiinnostusta kansainvälisten sijoittajien suunnalta, vuosina 2011-2013 investointeja alalle tuli yli 160 miljoonana euron arvosta (Hiltunen, K. Latva, S. Kaleva, P. 2013, 29). Suomalaisen pelitalojen menestykselle ei näy loppua, vaan uusia tekijöitä ja menestystarinoita syntyy jatkuvasti lisää. Viimeisimpänä Colossal Order, jonka Cities: Skylines tietokonepeli myi neljännesmiljoona kappaletta vuorokauden sisällä julkaisustaan (Lappalainen, E. 2015).

2.3 Tulevaisuus

Pelien parissa käytetään entistä enemmän aikaa ja niiden sosiaaliset ominaisuudet mahdollistavat ihmissuhteiden luomisen maailmanlaajuisesti. Pelit alkavat olla tunnustettu ja niiden merkitys osana kulttuuria on koko ajan kasvavassa asemassa, pelit eivät enää ole

vain pienen ryhmän harrastelua vaan koko maailman yksi suosituimmista ajankäytön muodoista (Suomen Pelinkehittäjät ry. 2010, 4).

3 PELIN SUUNNITTELU

Suunnittelu on yksi pelikehityksen tärkeimmistä vaiheista ja sen epäonnistuminen voi johtaa koko peliprojektin kaatumiseen. Mitä tarkemmin kehittäjä hoitavat suunnitteluvaiheen, sitä helpommaksi pelinkehittäminen muuttuu.

Pelin suunnittelu alkaa usein ideasta. Pelinkehittäjällä voi olla useita ideoita pöytälaatikossa odottamassa oikeaa hetkeä aloittaa niiden kehittäminen tai sitten ideoita ei vain ole, jolloin kehittämistä on vaikea aloittaa. Kiinnostusta herättävässä pelissä on syytä olla lähtökohtaisesti kiinnostusta herättävä idea. Kiinnostavan idean löydyttyä voidaan aloittaa varsinainen suunnittelutyö.

3.1 Game Design Document (GDD)

Ideasta, oli se kuinka yksinkertainen tahansa, on hyvä tehdä yksityiskohtainen suunnitelma, jota yleisesti alalla kutsutaan Game Design Documentiksi (”GDD”). GDD on tarkoitettu yksinomaan pelinkehittäjille ja sen parissa työskenteleville. GDD:n lukijoiden koostuessa kehittäjistä on se hyvä kirjoittaa nimenomaan heille, eikä esimerkiksi pelaajille. GDD:n tulisi sisältää pelin idea ja millainen valmis peli tulisi olemaan, sekä rajata projektia ja auttaa kehittäjiä keskittymään pelin ja sen ominaisuuksien kehittämiseen. GDD:n tekemiseen ei kuitenkaan ole oikeaa tapaa, vaan lähinnä tekemistä auttavia sääntöjä. Hyvän GDD:n voi kiteyttää seuraavan laisesti (Schubert, D. 2008):

- Tunne kohdeyleisösi, tällöin dokumentista on enemmän hyötyä lukijalle, jos se on kirjoitettu nimenomaan heille.
- Pidä dokumentti lyhyenä, mikä helpottaa dokumentin asioiden ymmärtämistä.
- Priorisoi suunnitelma, näin sen noudattaminen helpottuu, jos asiat on aseteltu tärkeysjärjestykseen.
- Kuvita dokumentti, sillä ”kuva kertoo enemmän kuin tuhat sanaa”.
- Käytä selkeitä termejä, jotka on helppo ymmärtää.
- Kirjoita määrätietoisesti, pyri käyttämään vahvoja sanoja ja välttämään sanoja kuten ”ehkä”, ”mahdollisesti” ja ”voitaisiin”.

- Pyri perustelemaan tarvittaessa miksi tietty päätös tehtiin.

Ennen kehitystyön aloittamista on hyvä miettiä myös projektinhallintaa, mikä helpottaa, varsinkin laajemman, projektin kasassa pitämistä. Mitä tiukemmin projekti on rajattu ja mitä selkeämmät tavoitteet sillä on, sitä helpompaa projektin toteuttamisesta tulee. Peli-projektin on tapana saada runsaasti uusia ideoita ja ominaisuuksia kesken toteutuksen. Tästä ei sinänsä ole minkäänlaista haittaa, jos kehittäjät pysähtyvät miettimään ideoiden tarpeellisuutta pelin kannalta. Ei ole syytä lisätä peliin sellaisia asioita, jotka eivät paranna pelikokemusta tai tuo siihen mitään parempaa.

Jokainen kehittäjä haluaa varmasti pelinsä menestyvän, tästä syystä on jo suunnitteluvaiheessa syytä ottaa myös muutama seikka huomioon. On syytä lähteä kehitetään peliä nimenomaan pelaajalle eikä itselle. Pelistä halutaan tehdä mahdollisimman kiinnostava, jotta se jaksaa kiinnostaa pelaajaa. Pelin ominaisuuksia on syytä miettiä myös toteutuksen aikana, olisiko mahdollista parantaa jotain tai mahdollisesti poistaa pelikokemusta huonontavaa kohtaa. Pelikokemuksesta on pyrittävä luomaan puoleensavetävä ja pelaajaa palkitsevana.

3.2 Kehitystiimi ja -työkalut

Peli-idean ja suunnitelman on myös syytä vastata kehitystiimin vahvuuksia tai vastaavasti kehitystiimi on syytä kasata vastaamaan pelisuunnitelmaa. Jos tiimistä löytyy osaava 2D graafikko, ei ole viisainta lähteä ideoimaan tai kehittämään kolmiulotteista peliä. Myös ohjelmointikieli on syytä lyödä lukkoon ennen pelin toteuttamisen aloittamista, sillä kielten muuttaminen kesken projektin johtaa yleensä koko ohjelmointi työn alusta aloittamiseen.

Suunnittelu vaiheessa on myös syytä ottaa huomioon kehittämiseen vaadittavat työkalut, kuten mahdolliset kuvankäsittelyyn, 3D-mallintamiseen ja pelintekemiseen tarvittavat ohjelmat. Ohjelmat on syytä valita projektin budjetin ja tarvittavuuden mukaan, esimerkiksi pelimoottoreista on saatavilla pienempiin projekteihin ilmaisia versioita. Kannattaa myös muistaa, että pelit sisältävät erilaisia ääniä ja musiikkia, joiden tekemiseen saattaa halutessa tarvita laitteita ja ohjelmistoja. Internetistä löytyy kuitenkin useita sivustoja,

joista on mahdollista ladata äänitehosteita ja musiikkia täysin ilmaiseksi, kunhan muistaa ottaa huomioon lisenssit. Äänet kuitenkin ovat pelin kannalta jopa välttämättömät, sillä ne syventävät pelikokemusta ja niiden avulla on mahdollista luoda tiettyä tunnelmaa peliin.

3.3 Prototyypin luominen

Ennen kuin peliä aletaan kunnolla toteuttaa, on siitä suositeltavaa luoda suunnitelman pohjalta prototyyppi. Prototyypin avulla on helppo kokeilla onko suunniteltu peli tai jokin sen osa varsinaisen toteuttamisen arvoinen vai vaatiiko jokin vielä hiomista. Itse prototyypin ei tarvitse olla kovin erikoinen, kunhan se noudattaa pelin suunnitelmaa. Prototyypin voi tehdä aluksi vaikka paperille ja käydä pelin vaiheita läpi. Tässä vaiheessa ei kuitenkaan ole tarkoitus keskittyä visuaalisiin seikkoihin. Prototyypin onnistuessa hyvin on sitä mahdollista lähteä jatkokehittämään.

3.4 Oman projektin suunnittelu

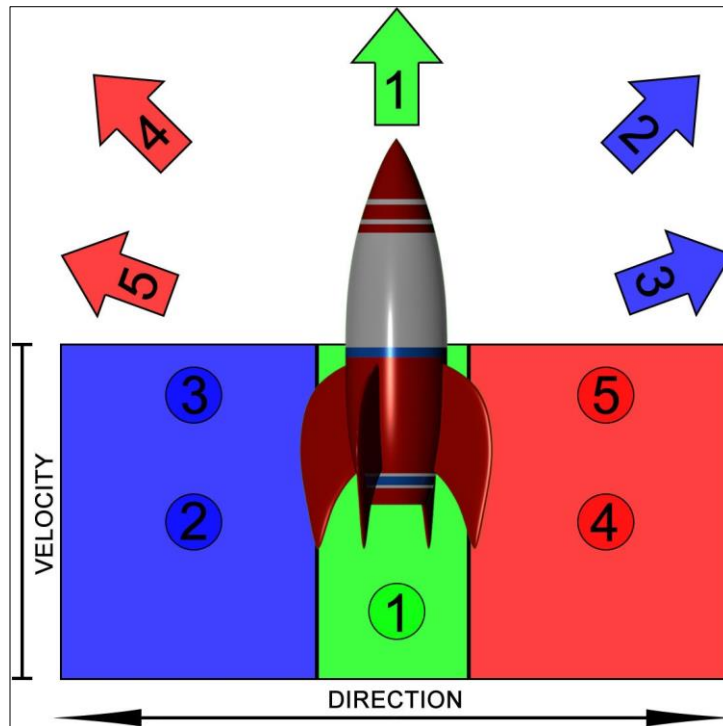
3.4.1 Suunnittelu

Projektin suunnittelu aloitettiin rajaamalla useasta ideasta se mitä peliä lähdetäisiin loppujen lopuksi toteuttamaan. Lähtökohtana pelille oli että sen tulisi olla suhteellisen yksinkertainen, jotta aikataulussa pysyminen onnistuisi mahdollisimman pienellä vaivalla. Yksinkertaisuus tuli kysymykseen myös siksi, että kehitystiimi koostui vain yhdestä henkilöstä. Vaikka taitoja ja ideoita monimutkaisemman projektin toteuttamiseen olisi myös ollut, päädyttiin juuri aikataulussa pysymisen vuoksi nyt toteutettuun ideaan. Peli toteutetaan nollabudjetilla, joten käytettävät työkalut pyritään pitämään ilmaisena.

Pelistä luotiin lyhyt GDD, joka toimi omana muisti- ja idealistana. Ominaisuuksia pelille pohdittiin pelaamalla samantyyllisiä pelejä. Näin saatiin koottua lista kriteereistä joita pelistä on hyvä löytyä. Lähinnä ominaisuudet keskittyivät siihen, mitä pelin valikoista tulisi

löytyä, miten pelissä määritellään pisteytys ja mitä siinä kerätään, sekä visuaalisen ilmeen miltä peli tulisi näyttämään. Pelimoottoriksi ja kehitysalustaksi valittiin Unity3D-ohjelma, jonka käytöstä löytyi eniten kokemusta. Lisäksi projektiin tarvittiin Cinema4D-, Photoshop- ja Visual Studio -ohjelmia, jotka kaikki oli saatavilla jollain tapaa ilmaiseksi, joko koulun tai muun tahon kautta.

3.4.2 Pelin kulku



Kuva 1 Yksinkertaistettu suunnitelma pelistä.

Toteutettu peli voidaan luokitella kuuluvaksi arcade-peleihin, sillä siinä on yksinkertaiset kontrollit, matala oppimiskäyrä ja lyhyet pelisessiot. Pelin idea yksinkertaisuudessaan on saada astronautti lentämään kuusta mahdollisimman korkealle. Astronautti hypähtää tiettyyn suuntaan ja tietyllä nopeudella näyttöä koskettamalla. Koskettamalla astronautin vasemmalle puolelle saadaan se lentämään oikealle päin ja päinvastoin. Hypyn voimakkuus määräytyy siitä, kuinka lähelle astronauttia kosketetaan. Kosketus aivan astronautin päällä seuraa lyhyeen hypähdykseen ja välimatkaa kasvattamalla myös hypyn voimakkuus kasvaa. Pelissä pyritään pääsemään mahdollisimman korkealle, jolloin saavutettu korkeus toimii pelin pisteytyksenä.

Peliä tuli päästä pelaamaan mahdollisimman nopeasti ja vaivattomasta, jolloin pelaajan olisi mahdollista palata pelin pariin aina kun hänellä olisi hetki aikaa. Peli koostuu lyhyistä peli sessioista, joita on mahdollista pelata nopeasti useita kertoja peräkkäin. Haluttiin myös, että pelistä löytyisi tilastot parhaista pelaajista ja heidän tuloksistaan, joita vastaan pelaaja voisi kilpailla ja pyrkiä pääsemään itse tuolle listalle.

4 PELIN TOTEUTUS

4.1 Pelimoottori

Itse pelin varsinainen kehittäminen alkaa toteutus vaiheessa. Kehittäjät voivat päättää tehdä kaiken itse, mutta vaihtoehtona on myös valita valmis pelimoottori. Pelimoottori on ohjelmistokehys, jotka useissa tapauksissa helpottavat kehitystä niiden valmiiden komponenttien vuoksi. Pelimoottoreiden sisältämät komponentit ovat tyypillisesti erilaiset luokkakirjastot ohjelmoinnin avuksi, fysiikkamoottori, törmäystunnistus, syötteiden hallinta. Moottori hoitaa myös grafiikan renderöinnin, eli 2D tai 3D grafiikan piirtämisen ruudulle, äänet, muistin hallinnan ja animaatiot. Suurin etu valmiissa pelimoottoreissa on kuitenkin, että kehittäjä voi keskittää kaikki voimansa pelinsä suunnitteluun ja toteuttamiseen (Ward, J. 2008).

Valmiista pelimoottoreita löytyy kuitenkin useita erilaisia ja kehittäjän tulee valita niistä se itselle sopivin. Osa pelimoottoreista tarjoaa kehittäjän käyttöön ainoastaan kasan työkaluja ja rajapintoja, joista on osattava valita ne itselle tärkeimmät. Osa taas tarjoaa käyttöliittymän ja valmiit työkalut oman pelin valmiiksi saamiseen mahdollisimman vähäisellä ohjelmoinnilla (Ward, J. 2008). Pelimoottoria valitessa kannattaa ottaa huomioon kehitystiimin kokemus ja taidot. Mikäli pelimoottori ei taivukkaan pelin tarpeisiin, voi se pahimmassa tapauksessa johtaa koko peliprojektin uudelleen aloittamiseen, tästä syystä on mietittävä tarkasti mitä ominaisuuksia siltä vaatii.

Nykypäivänä pelimoottoreita löytyy runsaasti valinnanvaraa ja niistä löytää varmasti omaan projektiinsa juuri sen oikean. Valinta ei kuitenkaan ole välttämättä helppo, sillä pelimoottorit sisältävät usein lähes identtisiltä kuulostavia ominaisuuksia, kuten tuki usealle alustalle ja ohjelmointikielelle. Aloittelevan kehittäjän on hyvä valita yksi suosituimmista pelimoottoreista, sillä niihin löytyy usein eniten opetusvideoita ja -dokumentteja, sekä niiden vertaistuki on usein kunnossa ja etsivä löytää näin helposti tarvitsevansa avun. Valmistajan sivuilta löytyy kattavasti dokumentteja ohjelman käytöstä ja sen teknisistä ominaisuuksista, joihin kannattaa tutustua ennen valinnan tekemistä. Asia mihin kannattaa kiinnittää myös huomiota, on pelien lisensointi, sillä pelimoottoreiden välillä niistä löytyy suuriakin rahallisia eroja.

4.2 Unity

Unity on Unity Technologiesin vuonna 2005 lanseeraama pelimoottori, jolla on mahdollista kehittää kaksi- ja kolmiulotteisia pelejä useille eri alustoille, kuten tietokoneille, konsoleille, mobiililaitteille ja verkkosivustoille. Alun perin ohjelma kantoi nimeä Unity3D, mutta nykyään se kulkee nimellä Unity. Unitystä julkaistiin 3. maaliskuuta 2015 ohjelman viides versio. Unity on maailmanlaajuisesti menestynyt pelimoottori, sillä se hallitsee 45% markkinoista ja jopa 47% pelinkehittäjistä käyttää sitä projekteissaan (Unity: Public Relations 2015). Unity valittiin oman projektin pelimoottoriksi sen ominaisuuksien, Windows Phone 8-tuen ja kehittäjän osaamisen vuoksi. Myös Microsoft suosittelee Windows pelien kehittämistä Unityllä sen pian sisältämän DirectX 12-tuen vuoksi (Windows Dev: Developing games on Windows 10. 2015).

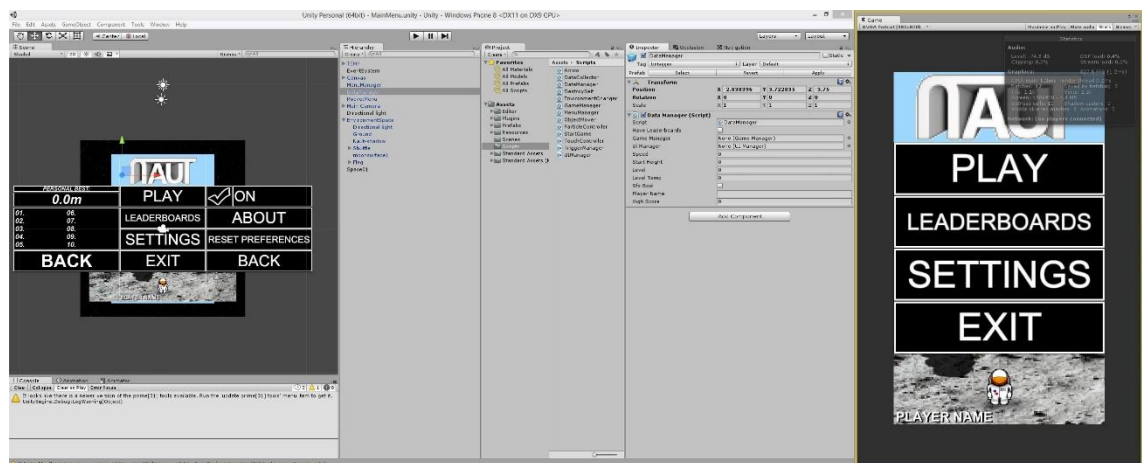
4.2.1 Yleistä

Unitystä löytyy monia monipuolisia työkaluja ja ominaisuuksia valmiina, mikä on yksi syy sen suureen suosioon. Tärkeimpiä ominaisuuksia ovat esimerkiksi sisäänrakennettu törmäystunnistus, monipuolinen Mecanim-animointityökalu ja oma fysiikkamoottori. Viides versio kasvatti ominaisuuksien listaa entisestään monilla toivotuilla asioilla, joita ovat muun muassa 64-bittinen editori, uusi audio-mikseri, reaali-aikainen global illumination, PhysX versio 3.3 ja WebGL esikatselun. Unityn vahvuuksia on myös tuki eri mallinnusohjelmille, joten kehittäjän on mahdollista helposti tuoda omat hahmot, rakennukset ja materiaalit omaan peliprojektiinsa.

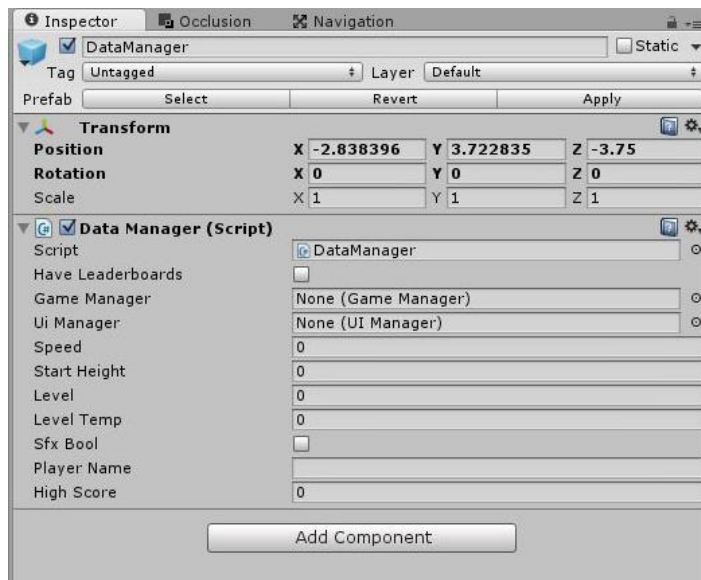
Unitystä on saatavilla ilmainen versio ja maksullinen Pro-versio, jonka saa käyttöönsä 1140€ tai 57€/kk. Joillekin alustoille on Unityn Pro-version lisäksi hankittava oma Pro-lisenssinsä, jotka maksavat saman verran kuin Unity Pro (Unity: Store 2015). Pro-versio avaa ilmaisversiossa lukittuna olleita ominaisuuksia joiden avulla kehittäjä voi tehdä pelistään omannäköisemmän, esimerkiksi kehittäjän itse määrittämällä käynnistysruudulla Unity-logon sijasta. Pro tarjoaa myös lukuisia työkaluja kehittämisen avuksi, kuten pelin

suorituskyvyn raportoinnin ja pilvipalvelussa tapahtuvan analyysin pelaajan käyttäytymisestä. (Unity: Get Unity 2015). Unity Technologies ei peri ilmaisversion käytöstä minikäänlaisia maksuja vaan sillä on mahdollista kehittää myös kaupallisia pelejä, kunhan kehittäjän liikevaihto on alle 100 000 dollaria (Unity: Unity Pro and Unity Personal Software License Agreement 5.x. 2015).

Itse pelin kehittäminen Unityssä tehdään graafisen editorin avulla. Unity-projekti koostuu palasista, jotka taas muodostavat pelin scenen. Usein kokonainen peli sisältää useita scenejä. Scenen voikin mieltää olevan kenttä tai valikko, joita Unityssä voi muokata ja testata erikseen suoraan editorissa. Editori voidaan jakaa erikokoisiin ikkunoihin tai niiden välilehtiin, joihin kehittäjä saa esille haluamansa työkalut. Kuvassa 1 näkyy oma Unity editorin ikkuna asettelu. Asetteleni sisältää kaikki oletus ikkunat, mutta hieman eri järjestyksessä. Vasemmassa laidassa on ikkunat kenttänäkymästä (Scene), jossa on mahdollista valita ja asetella projektin objekteja. Kenttänäkymän alla on konsoli-ikkuna (Console), mikä näyttää loki viestit, projektista löytyvät varoitukset ja virheet. Seuraavana oikealle siirryttäessä pidän projektin hierarkia- (Hierarchy) ja projekti-ikkunaa (Project). Inspector-ikkunasta kontrolloidaan valitun objektin ominaisuuksia, joita siihen on kiinnitetty. Oikeassa laidassa sijaitsee pelinäkymä (Game).

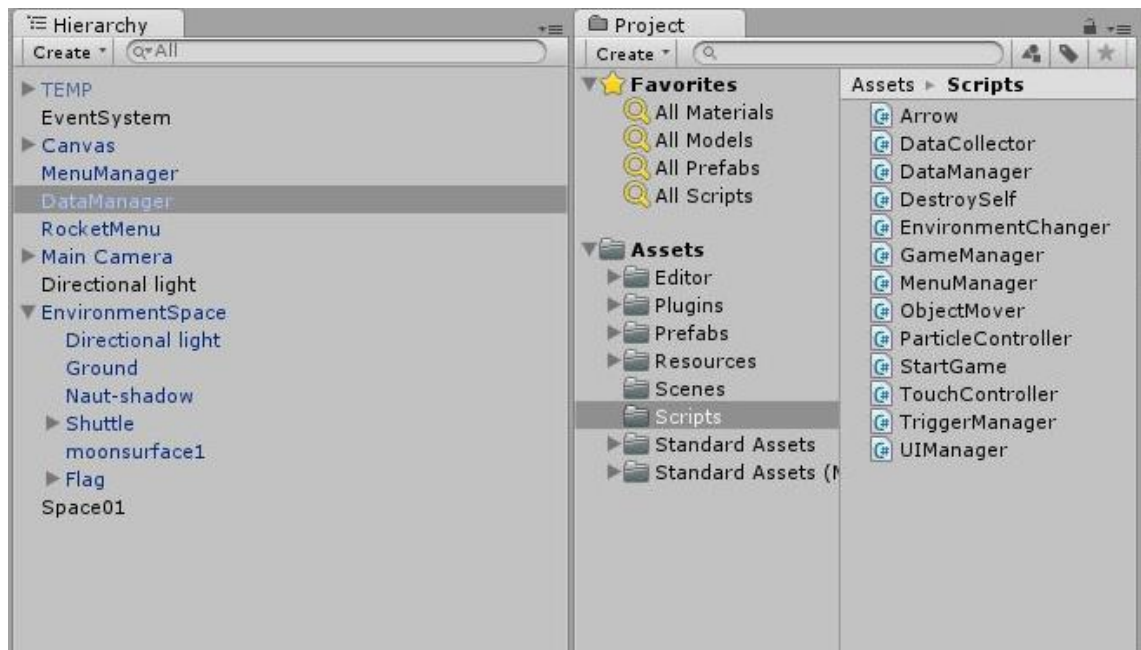


Kuva 2 Kuvakaappaus Unityn editorista.



Kuva 3 Kuvakaappaus Inspector-ikkunasta.

Unity projekti koostuu GameObjecteista, joihin on lisätty pelin tarvitsevia ominaisuuksia. Inspector-ikkuna näyttää tietoa valittuna olevasta GameObjectista. Mikäli GameObjectiin on lisätty skripti, kuten kuvassa 2, on sen julkisten muuttujien arvoja mahdollista muuttaa koodin sijaan suoraan Inspectorissa. Arvoja on mahdollista muuttaa myös pelin ollessa käynnissä ja näin hioa pelikokemus parhaaksi mahdolliseksi ilman jatkuvaa pelin uudelleen käynnistämistä. Inspector-ikkuna sisältää myös objektin sijainti-arvot (Transform). Transform sisältää tiedot objektin paikasta, suunnasta ja koosta pelimaailmassa. Inspector-ikkuna näyttää myös tiedot ja ominaisuudet projektiin tuoduista malleista ja materiaaleista.



Kuva 4 Kuvakaappaus hierarkia- ja projekti-ikkunoista.

Hierarkia-ikkunassa on lueteltu kaikki muokattavana olevan kentän (scene) objektit. Kyseessä voi olla esimerkiksi objekti mikä sisältää pelkän skriptin, suora viittaus johonkin projektin assettiin, kuten 3D-malliin. Objekteja on mahdollista siirtää toistensa alle, eli tehdä GameObjectista toisen GameObjectin lapsiobjekti (Unity Manual: Hierarchy). Projekti-ikkunassa on nähtävillä ja hallittavissa kaikki projektin sisältämät assetit. Assetit voivat olla mitä tahansa, kuten kuvia, 3D-malleja, animaatio-tiedostoja tai tekstuureita. Ikkunan vasen puoli näyttää kansiorakenteena projektin sisällön. Valitsemalla kansion pääsee tarkastelemaan ikkunan oikealla puolella yksityiskohtaisemmin kansion sisältöä. Projekti-ikkunan yläreunasta löytyy ”Create”-valikko, josta pystyy luomaan uusia kansioita tai asetteja projektiin (Unity Manual: Project Browser).

4.2.2 GameObject

Unityn kaikista tärkein objekti on GameObject, sillä kaikki mitä pelissä näkyy, on vähintäänkin osa GameObjectia. GameObjectit itsessään eivät tee mitään vaan niille tulee antaa komponentteja, jotta ne alkaisivat toimimaan kehittäjän haluamalla tavalla. Unity sisältää valmiita GameObjecteja, kuten erilaisia muotoja, mutta useimmin käytettävä GameObjecti minkä kehittäjä luo kenttäänsä on kuitenkin tyhjä. Tyhjälle GameObjectille

voidaan antaa esimerkiksi 3D-malli, kontrolli-skripti ja materiaali, ja näin on yksinkertaistettuna luotu kentälle kontrolloitava hahmo. GameObjectia voikin pitää ikään kuin säiliönä johon laittaa erilaisia komponentteja. GameObjecti sisältää kuitenkin tyhjänäkin Transform-komponentin, mikä määrittää sen sijainnin, suunnan ja koon (Unity Manual: GameObjects).

Pelistä on mahdollista luoda minkälainen tahansa lisäämällä erilaisia komponentteja GameObjectiin. Unitystä löytyy valmiita komponentteja, joiden avulla on mahdollista saada GameObjectille esimerkiksi fysiikat, törmäyksen tunnistus tai ääni. Komponenttien ominaisuuksia voi muokata inspector-ikkunassa, mistä voi myös katsoa mitä komponentteja valittu GameObject sisältää (Unity Manual: Using Components).

GameObjectista voidaan luoda Prefab, eli tallentaa se projektikansioon assetiksi. Projektikansion kautta voidaan tämän jälkeen muokata yhtä Prefabiä ja muutokset tulevat voimaan kaikkiin kyseisen Prefabin ilmentymiin, jolloin myös projektin hallittavuus helpottuu. Voit kasvattaa esimerkiksi kerättävien kolikoiden kokoa projektikansiosta ja kaikki kolikot pelimaailmassa kasvavat kyseiseen kokoon, jokaisen kolikoiden kokoa ei tarvitse siis erikseen käydä kasvattamassa. Prefabiä voidaan kutsua myös skriptissä, esimerkiksi ampumis-nappia painamalla voidaan synnyttää luoti-prefab ja laittaa se liikkumaan tiettyyn suuntaan (Unity Manual: Prefabs).

4.2.3 Ohjelmointi

Ohjelmointi Unityssä tapahtuu ulkoista ohjelmaa, esimerkiksi mukana tulevaa MonoDevelop tai normaalia tekstinkäsittely-ohjelmaa. Käytti mitä ohjelmaa tahansa, on lopputuloksena koodi, jota Unityssä kutsutaan skriptiksi. Skriptit toimivatkin Unityssä kehittäjän oma tekemänä komponenttina, mikä voidaan liittää GameObjectiin tuomaan sille vieläkin laajempia ominaisuuksia, verrattuna Unityn valmiisiin komponentteihin. Peli tarvitsee skriptin esimerkiksi pelaajan kontrollointiin. Pelin tekijä voi skriptien avulla kehittää ja liittää oman tekoälyn pelinsä hahmoille. Voikin sanoa, että vain mielikuvitus on rajana skriptien kehittämisessä ja käyttämisessä (Unity Manual: Scripting Section).

Unity tukee tällä hetkellä kolmea eri ohjelmointikieltä, joilla kehittäjän on mahdollista kirjoittaa skriptejään. Nämä kolme ovat C#, JavaScript ja Boo. Kehittäjän on mahdollista käyttää vaikka kaikkia kolmea kieltä yhdessä projektissa. Kuitenkin sillä varauksella, että eri kieliset skriptit eivät välttämättä keskustele toisiensa kanssa (Unity Manual: Scripting). Kielen valinta onkin kiinni hyvin pitkälle kehittäjän omista mieltymyksistä.

Oli kielivalinta mikä tahansa, toimivat Unityn kirjastot jokaisessa näissä. Itse olen valinnut ohjelmointi kieleksi C#:in, sillä se tarjoaa sopivassa suhteessa C++ monipuolisuutta ja JavaScriptin helppokäyttöisyyttä. JavaScript-kieli sopii hyvin vasta-alkajalle sen helppokäyttöisyyden vuoksi, lisäksi sen yleisyys helpottaa vastauksen löytämistä mahdollisiin ongelmatilanteisiin. Verkosta löytyy kattava dokumentaatio, josta löytää tietoa ja esimerkkejä Unityn eri toiminnallisuuksista. Dokumentaation skripti esimerkit on vaihdettavissa C# tai JavaScriptiksi. Vähäisen käytön vuoksi Boo-kieli on poistunut dokumentaatiosta viidennen Unity -version ilmestyttyä. Aktiivisen yhteisön ansiosta ratkaisua ongelmatilanteisiin voi etsiä Unityn foorumeilta tai Unity Answers -sivustolta (Unity Manual: Index).

Unityssä skriptit periytetään sen MonoBehaviour perusluokasta, mikä mahdollistaa sisäisten luokkien ja metodien käytön (Unity Script Reference: MonoBehaviour). Pelin käynnistyessä ajetaan Unityssä Awake- ja Start-metodit, joita käytetäänkin pääasiallisesti eri muuttujien alustamiseen ja viittausten luomiseen. Ajallisesti eri skriptien Awake-metodit ajetaan satunnaisesti, tämän jälkeen on Start-metodien vuoro. Tämä on syytä ottaa huomioon erityisesti viittauksia tehtäessä. Esimerkiksi muuttuja B on syytä alustaa Awake-metodissa, mikäli muuttuja A tarvitsee B:tä alustamiseensa. Tämän jälkeen suorittaa A muuttujan alustaminen Start-metodissa, kun B muuttuja jo löytyy (Unity Script Reference: MonoBehaviour.Start).

Unityn käytetyin metodi on Update, joka ajetaan jokaisella ruudunpäivityksellä. Metodin sisällä suoritetaan yleensä suurin osa skriptin toiminnoista, kun halutaan reaaliaikaista toimintaa peliin (Unity Script Reference: MonoBehaviour.Update). FixedUpdate-metodi on eräänlainen muunnos Update-metodista. FixedUpdatea suositellaan käytettäväksi fyysikkakomponenttien ohjaamiseen, koska ruudunpäivityksen sijasta, sitä kutsutaan ennalta määrätyin väliajoin (Unity Script Reference: MonoBehaviour.FixedUpdate).

4.3 Muut kehitystyökalut

Kehittäjän syytä ottaa huomioon myös muut pelinkehitykseen liittyvät työkalut. Yleensä pelissä on jonkinlaisia hahmoja ja materiaaleja, joiden tekemiseen vaaditaan työkaluja. 2D-hahmojen ja materiaalien tekeminen onnistuu millä tahansa piirto-ohjelmalla kuten Adobe Photoshop tai yksinkertainen Paint. 3D-malleja varten on luonnollisesti oltava ohjelma mikä kykenee kolmiulotteisten objektien luomiseen, tällaisia ohjelmia ovat esimerkiksi Cinema4D, Autodesk Maya tai Blender. Pelimaailmaa syventävät siitä löytyvät äänet, joiden itse tekemiseen tarvitaan myös jonkinlainen työkalu. Äänien ja materiaalien kohdalla on kuitenkin mahdollisuus käyttää hyväksi Internetistä löytyviä kirjastoja, joista voi helposti löytää tarvittavansa. Kirjastoja käytettäessä on syytä muistaa ottaa huomioon tekijänoikeudet ja perehtyä äänen tai materiaalin lisensiointiin.

4.4 Testaus

Ennen tuotoksensa julkaisemista on peliä syytä testata, jotta valmis tuote olisi mahdollisimman virheetön. Suurella todennäköisyydellä peliin jää joitain virheitä eli bugeja. Riittävällä testauksella voidaan estää, että ne eivät ole peliä kaatavia tai kokemusta pilaavia. Peli kannattaa antaa pelattavaksi jollekin ulkopuoliselle taholle, sillä kehittäjä tulee usein sokeaksi omalle tuotokselleen, eikä näin välttämättä huomaa kaikkea mikä on vialla. Julkaisua ei kannata liiaksi pitkittää virheiden vuoksi, vaan niistä kannattaa korjata kriittisimmät. Muiden pienempien virheiden korjaaminen on mahdollista hoitaa jälkikäteen päivityksinä. Tämän vuoksi kannattaa myös miettiä pelin elinkaarta julkaisun jälkeen, esimerkiksi kuinka peliin julkaistaan päivityksiä tai mahdollisia uusia ominaisuuksia.

4.5 Oman pelin toteutus

Pelin tuli alun perin olla kuvattu 2.5D näkymästä. Loppujen lopuksi päädyttiin toteuttamaan peli täysin kaksiulotteisena. 2D grafiikan luominen ei ollut kovin vahvalla mallilla,

joten päädyttiin renderoimaan 3D-mallista kaksiulotteisia kuvia, jolloin saatiin nopeasti luotua astronautista kuvia eri asennoissa. Näitä käsiteltiin kuvankäsittely ohjelmalla ja luotiin sprite-sheet, mikä sisälsi kaikki astronautin tarvittavat asennot.

Peli sisältää useita C#-skriptejä yksinkertaisuudestaan huolimatta. Suuri osa pelin toiminnallisuudesta on toteutettu käyttäen Unityn uutta valikko systeemiä (UI-system) ja animoimalla valikko-objekteja. Näistä käydään seuraavassa läpi pelin kannalta tärkeimmät kohdat.

4.5.1 TouchController.cs

TouchController-skripti on pelaajan komponentti ja se hoitaa pelaajan liikkeen kontrolloinnin, sekä sijainnin eri pelitilanteessa. Pelin alkaessa skripti tarkistaa, että pelaaja löytyy maailmasta ja siirtää sen tarvittaessa aloitus paikkaan.

Itse pelaamisen alettua skripti suorittaa korkeamman alku hypyn astronautille, minkä jälkeen kontrollit ovat pelaajan käsissä. Peli hyväksyy vain yhden kosketuksen kerrallaan, jotta pelaajan on ”näpyteltävä” ruutua mahdollisimman paljon. Kosketuksesta otetaan väliaikaiseen muuttujaan talteen sen sijainti ja muutetaan tämän jälkeen kameran sijaintia vastaavaan pelimaailman sijaintiin. Saatua sijaintia verrataan astronautin sijaintiin. Näin saadaan vertailtua mille puolelle astronauttia kosketus tapahtuu ja saadaan hypäytettyä astronauttia vastakkaiseen suuntaan Start-metodissa määritellyllä voimakkuudella. Voimakkuudesta vähennetään kuitenkin ennen hyppyä etäisyyden suhde, eli kuinka läheltä astronauttia on näpytetty. Mitä kauempana kosketus tapahtuu, sitä kovempaa hyppy tapahtuu. Astronauttia on siis helpompi kontrolloida mitä lähempää sitä koskettaa. Mikäli astronautti ajautuu kulkemaan näytön reunaa, on tilanne mahdollista pelastaa. Skripti tunnistaa tilanteen ja ottaa suunnan pois käytöstä ja vaihtaa sen niin että kosketuksesta astronautti tekee hypyn takaisin kohti pelialuetta. Astronautin tulee kuitenkin näkyä näytön reunalla, jotta pelastus on mahdollista.

```

109 // sprite change:
110 if(this.rigidbody.x > 0 && this.rigidbody.y > 0) {
111     nautRenderer.sprite = nautRight;
112 } else if(this.rigidbody.x < 0 && this.rigidbody.y > 0) {
113     nautRenderer.sprite = nautLeft;
114 } else {
115     nautRenderer.sprite = nautCenter;
116 }

```

Kuva 5 Astronautin sprite-kuvan kontrollointi.

Skripti hoitaa myös astronautin spriten, eli tässä tapauksessa hahmon kuvan vaihtamisen. Astronautin kuva vaihtuu kuvan 5 mukaisesti. Astronautin rigidbodyn sijainnin ollessa suurempi kuin nolla, eli sen liikkeessä oikealle, vaihdetaan astronautin piirtäjän (renderer) sprite vastaamaan oikealle liikkuvaan kuvaan (kuvassa muuttujaksi "nautRight"). Sama tehdään vasemmalle liikuttaessa, mutta kuva on vasemmalle liikkuja ("nautLeft"). Mikäli kumpikaan ei ehdoista ei täyty on astronautti jälleen keskiasennossa ("nautCenter").

4.5.2 TriggerManager.cs

TriggerManager-skripti hallitsee maailmassa sijaitsevia triggereitä, eli alueita joille mentäessä tai poistuttaessa on mahdollista saada jotain tapahtumaan. TriggerManager on myös liitetty pelaaja GameObjectin komponentiksi, jolloin se kulkee kokoajan mukana. Skriptissä on kolme metodia. Start-metodi missä alustetaan viittaus manageri-skriptiin.

OnTriggerEnter2D-metodi missä tarkistellaan onko GameObjecti koskettanut kyseistä triggeriä. Metodi sisältää tarkistukset, onko osuttu Deathzone-triggeriin. Mikäli näin on, vähennetään elämiä ja palautetaan peli takaisin alkutilaan. Metodissa on tarkistus edellisessä kappaleessa selvitetyle ruudun reunalla olemiselle. Ruudun reumat sisältävät triggerit, ja näihin osuttaessa muutetaan tätä tapahtumaa määrittävä boolean muuttuja true-tilaan.

Viimeinen OnTriggerExit2D-metodi sisältää tarkistukset onko triggerin vaikutusalueelta poistuttu. Pelissä metodia käytetään muuttamaan reunan tunnistus booleanit takaisin false-tilaan.

4.5.3 DataManager.cs

DataManager-skripti pitää sisällään peliin liittyviä kentille yhteisiä muuttujia ja arvoja, jotka kulkevat kentistä toiseen. Se huolehtii myös pelin tietojen tallentamisesta ja hake- misesta. Pelissä tiedot tallennetaan PlayerPrefs:iin, johon on mahdollista käyttää kätevästi asetusten tallentamiseen. Windows Phone 8:ssa PlayerPrefs sijaitsee sovelluksen paikalli- sessa kansiossa (Unity Script Reference: PlayerPrefs).

```
25 void Awake() {  
26     DontDestroyOnLoad(gameObject);  
27     if (FindObjectsOfType(GetType()).Length > 1)  
28     {  
29         Destroy(gameObject);  
30     }  
31 }
```

Kuva 6 DataManager.cs Awake-metodi.

DataManager sisältää kuvan 6 kaltaisen Awake-metodin. Tämän ansiosta GameObjecti on mahdollista siirtää kentästä toiseen, koska sitä ei poisteta latauksessa. Metodiin on upotettava tarkistus, ettei kyseisiä GameObjecteja löydy useampaa, näin tapahtuessa ky- seinen objekti poistetaan.

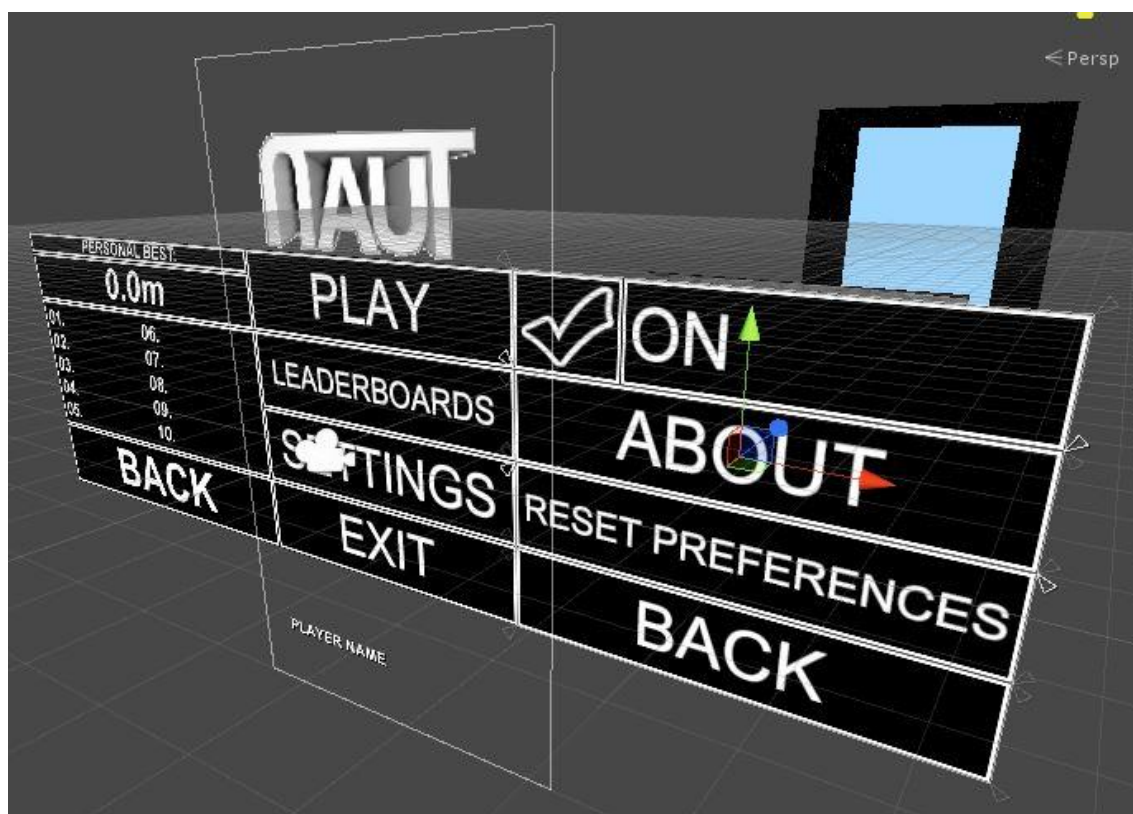
Start-metodissa tarkistetaan löytyykö pelaajan huipputulos PlayerPrefsistä. Tuloksen löy- tyessä siirretään se muuttujaan talteen, muussa tapauksessa alustetaan nolla tulos PlayerPrefsiin ja ladataan se. Skripti myös alustaa viitaukset eri skripteihin riippuen ken- tästä, jolloin niihin päästään käsiksi helposti. DataManager toimii huipputuloksen tarkis- tajana ja tietokantaan siirtäjänä, tässä on kuitenkin ongelmia palveluiden maksullisuuden kanssa, joten ominaisuus on poistettu käytöstä.

4.5.4 Valikot ja animaattori

Suurta osaa projektissa yksinkertaisuutensa vuoksi näyttelevät valikot, jotka on toteutettu käyttäen UI-työkalua ja objekteja, sekä Unityn animaattoria. UI (User Interface) on täysin

uudistettu ja toimii nykyään samalla GameObject periaatteella. Aiemmin valikot jouduttiin Unityssä tekemään täysin koodissa omassa metodissaan, mutta uudistuksen myötä niiden luominen on nopeampaa. Valikoista on myös mahdollista tehdä entistä visuaalisempia ja interaktiivisempia.

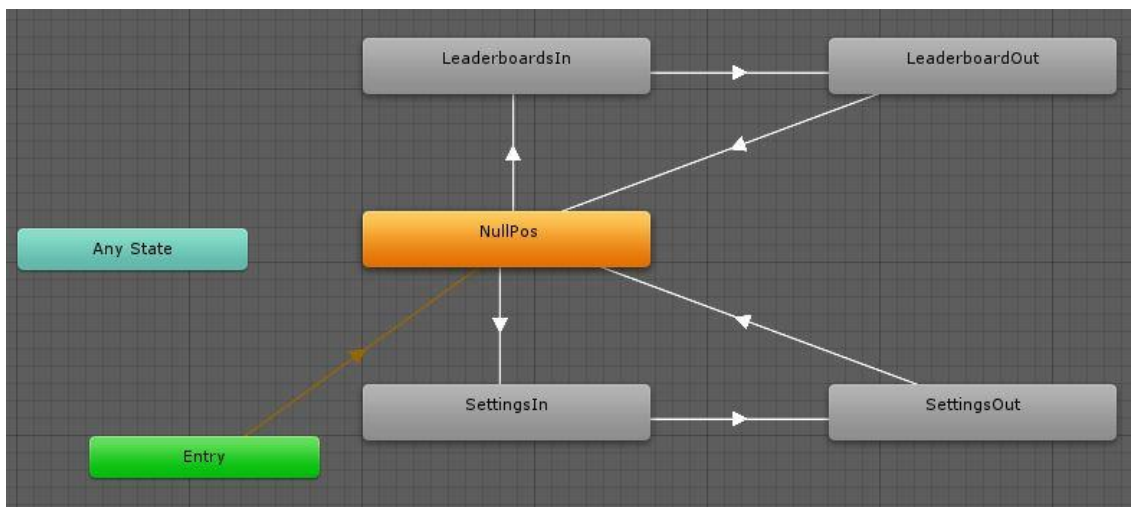
Unityn UI rakentuu Canvas-objektin, GameObjectin millä on canvas-komponentti, sisään. Canvas-komponentti sisältää eri näyttö tiloja (render mode), mitkä määrittävät näytetäänkö UI esimerkiksi pelikamerassa vai pelimaailmassa. UI on mahdollista asettaa perinteisesti näkymään kaikkien objektien päällä, tietylle etäisyydelle kamerasta. Mahdollisuutena on myös asettaa UI pelimaailmaan, jolloin on esimerkiksi helposti mahdollista luoda vihollisten päällä sijaitsevia energia mittareita. Mahdollisuuksia tämän uuden systeemin ansiosta on lukuisia.



Kuva 7 Pelin alkuvalikko editori-ikkunassa.

Itse UI-objektit, kuten kuvat, tekstit ja napit, sijoitetaan Canvas-objektin lapsiksi. Jokainen UI-objekti saa Rect Transform -komponentin. Tässä komponentissa on määritelty objektin koko ja sijainti canvas-objektin suhteen. Lisäksi on mahdollista ankkuroida ob-

jekti pysymään tietyssä paikassa. Tällöin objekti voidaan saada myös skaalautumaan canvas-objektin mukana, mikä helpottaa UI:n suunnittelua esimerkiksi mobiililaitteiden näyttöjen resoluutioiden vaihtuessa.



Kuva 8 Valikon animaatio rakenne Mecanimissa.

Valikon UI-objekteja on mahdollista animoida samalla tavalla kuin muita GameObjecteja Unityssä. Animointi tapahtuu luomalla uusi animaatio, johon yksinkertaisuudessaan annetaan esimerkiksi objektin aloituspaikka ja halutun ajan kuluttua lopetuspaikka. Animaatioiden kontrollointi tapahtuu Unityssä sen omalla Mecanim animaatio systeemillä.

Kuvassa 7 näkyvä valikko toimii Kuvan 8 tavoin. Kuvassa 6 ollaan kuvan 7 tilanteessa ”NullPos”. Kuvassa 7 näkyvien tilojen välillä oleville nuolille on mahdollista antaa ehtoja, joiden käydessä toteen siirrytään nuolen osoittamaan tilaan. Esimerkiksi painettaessa ”settings”-nappia täyttyy ”NullPos” ja ”SettingsIn” välisen nuolen ehto ja näin siirrytään toistamaan ”SettingsIn”-animaatiota. Tätä animaatiota toistetaan kunnes täytetään ehto siirtyä ”SettingsOut”-animaatioon. ”SettingsOut” ei sisällä ehtoa, jolla päästään takaisin ”NullPos”-tilaan, vaan animaatio on määritetty vain toistamaan itsensä kerran ja siirtymään eteenpäin. Valikoiden animointia helpottaa myös, että animaatioita on mahdollista toistaa väärinpäin, jolloin ei tarvitse animoida paluu animaatiota erikseen. Samalla tavalla pystytään luomaan pelihahmon animaatiot. Hahmon animaattori siirtyy pyörittämään juoksu-animaatiota, kun nopeus nousee tietyn pisteen yli ja palaa takaisin alkutilaan nopeuden laskiessa.

4.5.5 Ongelmat toteutuksessa

Peliä kehittäessä on syytä varautua ongelmiin, sillä niitä tulee melko varmasti. Toteutus onkin jatkuvaa ongelmanratkaisua. Suurimmilta osin toteutus onnistui odotetulla tavalla, kiitos hyvän suunnittelu työn. Oman toteutuksen ongelmat kohdistuivat lähinnä toiminnallisuuksiin, kuten globaaleihin tulostauluihin ja projektin kokonaisuuden huonoon jaotteluun.

Suunnitelmissa oli luoda järjestelmä maailmanlaajuisten tulosten listaamiseksi. Tämä suunnitelma kaatui lähinnä siihen seikkaan, että pitkäaikaiseen tallennus tilaan ei ollut resursseja. Niinpä tästä jouduttiin luopumaan jo melko hyvissä ajoin projektin toteutusta. Pelistä kuitenkin löytyy melko valmis systeemi tälle, mikäli se jossain kohtaa päätetään toteuttaa. Toteutuksen aikana kokeiltiin saada tulokset tallentumaan Microsoftin Azure - palveluun, mutta tämä hylättiin palvelun muuttuessa maksulliseksi.

Lisäksi alun perin toteutus oli suunniteltu hieman monimutkaisemmaksi kaikkine ominaisuuksineen. Tämän hetkinen lopputulos on toimiva ja halutun kaltainen, mutta sitä olisi mahdollista yksinkertaistaa huomattavasti. Kyseinen peli olisi mahdollista toteuttaa vain yhteen kenttään (scene) yhdistämällä päävalikko itse pelikenttään. Tämä toteutetaan todennäköisesti ennen varsinaista julkaisua.

Peli on tällä hetkellä beta-vaiheessa, sillä se ei sisällä vielä minkäänlaisia ääniä. Äänille on osaksi luotu valmiita paikkoja. Tarkoituksena ja toiveena on opetella itse tekemään ja muokkaamaan ääniä.

5 PELIN JULKAISU

Projektin julkaisulle on tässä tapauksessa oikeastaan vain yksi vaihtoehto. Koska peli on tehty nimenomaan Windows Phonelle, on peli lähes pakko julkaista Windows Kaupassa. Alustakohtaiset digitaaliset julkaisukanavat ovatkin mahdollistaneet sen, että pelinkehittäjä voi itse toimia myös julkaisijana. Koska välikäsiä on mahdollisimman vähän, jää kehittäjälle itselleen suurempi osa tuotoista. Lisäksi Windows Kaupan tyylisten julkaisukanavien etuna itsenäiselle kehittäjälle on, että maksujärjestelmä on suoraan integroitu niihin (Hiltunen, K. Latva, S. Kaleva, P. 2013, 23). Maksullisena julkaistun sovelluksen tuotoista Windows Kauppa ottaa 30%, mikä vastaa tasoltaan muiden alustojen omia kanavia (Microsoft: Getting Paid).

5.1 Kehittäjätili

Pelin julkaisemiseen Windows Kaupassa vaaditaan kehittäjätili. Kehittäjätili maksaa Suomessa yksityiseltä kehittäjältä 14€ ja yritykseltä 75€. Mahdolliset tuotot on mahdollista saada suoraan pankkitilille tai PayPal-tililleen (Microsoft: Account types, locations, and fees). Kehittäjäksi rekisteröitymiseen vaaditaan Microsoft-tili, jolla palveluun kirjaututaan sisään. Tämän jälkeen täytetään erinäisiä tietoja, kuten maa missä asut tai toimit. Tässä kohtaa on mahdollista valita myös julkaisija nimi, jonka alla sovellukset julkaistaan kaupassa. Joudut myös täyttämään maksutiedot, jolta kehittäjätilin hinta veloitetaan.

Mikäli suunnitelmissa on julkaista maksullisia sovelluksia, on täytettävä tiedot payout-tiliä ja vero profiilia varten. Ilman näitä ei voi saada sovelluksiensa tuottoja omille tileille. Payout-tilillä tarkoitetaan sitä pankkitiliä johon mahdolliset tuotot maksetaan. Pankin on sijaittava samassa maassa, jonka on kehittäjätililleen määrittänyt. Verotusta varten on täytettävä tiedot asuin maasta ja asiaan kuuluva vero-lomake. Suomessa asuvan kehittäjän on tuottoa saadakseen täytettävä Yhdysvaltojen verolomake IRS W-8. Yhdysvaltojen ulkopuolelta rekisteröityneeltä pidätetään 30% tuotoista, tätä on mahdollista pienentää mikäli asuin maalla on tulovero sopimus Yhdysvaltojen kanssa (Microsoft: Setting up your payout account and tax forms).

Account settings

Account details

[Redeem code](#)

Basic info

Account status: Active

Verification status: Authorized

Account type: Individual

Publisher IDs

Seller ID

Windows publisher ID

Windows Phone publisher ID

Contact info

[Update](#)

Public info

Publisher display name

Seller contact info

Name

Email

Phone

Address

Kuva 9 Kehittäjätilin näyttämät tiedot (arvot poistettu).

5.2 Oman toteutuksen julkaisu

WMAAppManifest.xml

Use this designer to set or modify some of the properties in the Windows Phone app manifest file.

Application UI

Capabilities

Requirements

Packaging

Use this page to set the UI details that identify and describe your application.

Display Name:

Description:

Navigation Page:

App Icon:

...

Supported Resolutions:

☒ wvga

☒ wxga

☒ 720p

Tile Template:

TemplateFlip

☒ Support for large Tiles

Tile Title:

Tile Images:

Small:

...

Medium:

...

Large:

...

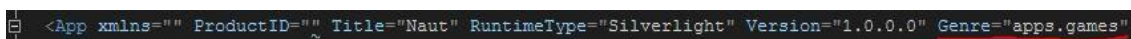
Kuva 10 Projektin tietoja Visual Studiassa.

Ennen oman pelin julkaisua on ensin Unityssä suoritettava pelin build, mikä luo Windows Phone -projektissa Visual Studio -projektin. Luodusta Visual Studio -projektin kansioista löytyvät muun muassa pelin kuvakkeet, jotka on syytä muokata kuvan käsittely ohjelmalla vastaamaan peliä. Visual Studiossa on syytä avata myös projektin asetukset ja täyttää ne (kuva 12). Peliin lisätään kuvassa 11 myös ominaisuus nopealle palaamiselle pelin pariin. Mikäli pelin aikana joudutaan esimerkiksi vastaamaan puhelimeen, voidaan peliä jatkaa suoraan siitä mihin jäätiin uudelleen käynnistämisen sijasta. Myös pelin tyyppi on syytä vaihtaa sovelluksesta peliksi.



```
<Tasks>
  <DefaultTask Name="_default" NavigationPage="MainPage.xaml" ActivationPolicy="Resume" />
</Tasks>
```

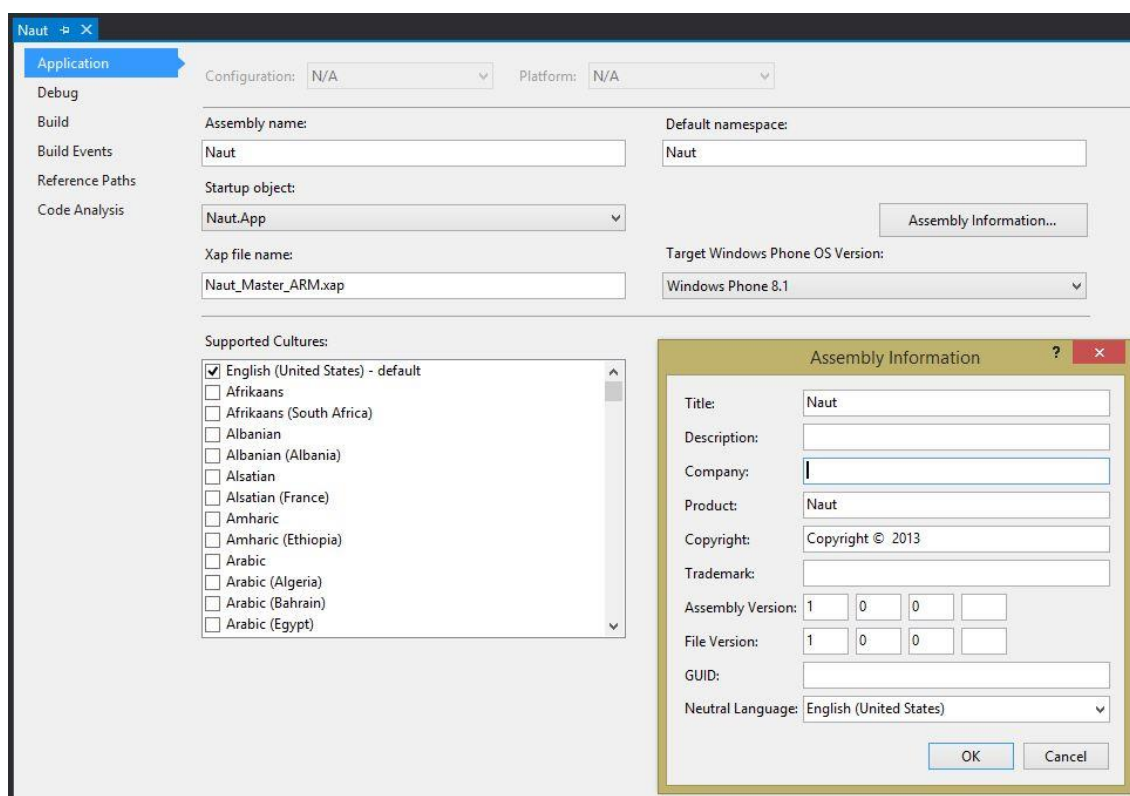
Kuva 11 Activation policyn muuttaminen Visual Studiossa.



```
<App xmlns="" ProductID="" Title="Naut" RuntimeType="Silverlight" Version="1.0.0.0" Genre="apps.games"
```

Kuva 12 Määrittäminen peliksi Visual Studiossa.

Visual Studiosta löytyy Windows App Certification Kit -ominaisuus, mikä on myös hyvä ajaa läpi ennen pelin lähettämistä julkaistavaksi. Windows App Certification Kit käy läpi projektista eri asioita varmistaen, että peli on mahdollisimman valmis julkaistavaksi (Microsoft: Windows App Certification Kit tests for Windows Store apps).



Kuva 13 Projektin asetukset Visual Studiassa.

Pelistä on löytyy Visual Studio -projektin kansioista .xap-päätteinen tiedosto, joka on nyt mahdollista ladata kauppa paikalle. Lisättäessä sovellusta varataan sille aluksi nimi, tämän voi tehdä heti kun nimi on tiedossa, sillä se säilyy varattuna yhden vuoden. Kun nimi on saatu, on mahdollista alkaa täyttämään muita sovelluksen julkaisuun vaadittavia tietoja (kuva 13).

Pelille on määriteltävä hinta ja se missä maissa se on saatavissa. Hinta haarukka on asetettavissa ilmaisesta 999.99 Yhdysvaltain dollariin. On myös määritettävissä, milloin pelin julkaisu tapahtuu. Julkaistaanko peli heti hyväksynnän jälkeen vai halutaanko se mahdollisesti itse julkaista. Pelille on määritettävä genre, mihin se kuuluu, sekä sallittu ikäryhmä (Microsoft: App submissions). Kun tietoihin ollaan tyytyväisiä, voidaan peli lähettää eteenpäin hyväksyttäväksi. Hyväksymis prosessissa käydään läpi, ettei peli sisällä haittaohjelmia tai viruksia. Prosessista saa palautteen, missä kerrotaan onko peli hyväksytty kauppaan, vai onko siinä jotain korjattavaa. Windows Kauppaan julkaistavien sovellusten ja pelien tulee noudattaa kehittäjä ehdoissa hyväksyttyjä toimintaperiaatteita, kuten asiakkaalle vastinetta ja arvoa sovelluksen juuri sinun kehittämän pelin lataamisesta. Käyttäjän tai asiakkaan harhaanjohtaminen ja huijaaminen on kiellettyä (Microsoft: Windows Store Policies). Ehtoihin on hyvä tutustua kehittäjätiliä luodessaan.

Naut

App overview

Analytics ▾

Submissions

Submission 1

IAPs

Monetization ▾

Services ▾

App management ▾

← Dashboard overview

Submission 1

Delete

Pricing and availability	Not started	<input type="radio"/>
App properties	Not started	<input type="radio"/>
Packages	Not started	<input type="radio"/>
Descriptions	Not started	<input type="radio"/>
You'll be able to edit your descriptions after you upload packages.		
Notes for certification	Optional	<input type="radio"/>

Kuva 14 Sovelluksen lähettämiseen vaadittavia tietoja.

6 YHTEENVETO

Opinnäytetyön tavoitteena oli antaa aloitteleville ja pelinkehityksestä kiinnostuneille yleiskuva siitä mitä pelinkehittäminen on ja mitä siinä on syytä ottaa huomioon. Tavoitteessa onnistuttiin mielestäni hyvin ja opinnäytetyöstä saa poimittua asioita ja vinkkejä oman peliprojektin aloittamiseen. Opinnäytetyöstä saa tiedon siitä minkälaisia tietoja ja taitoja yksinkertaisen pelin suunnittelu ja tekeminen vaatii, sekä mitä tietoja tarvitaan sen julkaisemiseen Windows Storessa.

Tarkoituksena toteutettiin peli Windows Phone 8 -alustalle ja saada se julkaistua Windows Storessa. Tässä onnistuttiin vaihtelevasti, sillä tämän hetkinen versio pelistä on periaatteessa ääniä ja pientä viilaamista vaille valmis julkaistavaksi. Alun perin suunniteltuja ja haluttuja ominaisuuksia on kuitenkin jouduttu jättämään pelistä pois. Maksullisuudesta johtuen toivottua globaalia tulostaulua ei saatu toteutettua suunnitellulla tavalla, käyttäen Microsoft Azure -palveluita. Pelistä puuttuu myös äänet, mikä on pääasiallinen syy julkaisematta jättämiselle. Toiveissa on myös elävöittää pelimaailmaa erilaisilla tausta grafiikoilla, kuten esimerkiksi liikkuvalla maapallolla ja erilaisilla avaruudesta löytyvillä objekteilla. Peli saadaan helposti julkaisua ilman tulostauluja, kun edellä mainitut asiat on saatu korjattua. Jatkokehityksen kannalta peli on suunniteltu niin, että siihen on julkaisun jälkeen helppo tuoda lisää ominaisuuksia päivitysten avulla. Toiveissa on löytää vaihtoehtoinen tapa toteuttaa globaalit tulostaulut ominaisuus ja tuoda se peliin päivityksellä myöhemmin.

Opinnäytetyöstä on helposti todettavissa se, että nykyään pelinkehittämisen aloittaminen on helppoa ja onnistuu pieneltäkin ryhmältä vaivattomasti, mikäli kiinnosta löytyy. Budjettikaan ei tule olemaan mikään este sillä saatavilla on monia ilmaisia kehitystyökaluja, joiden avulla saa varmasti luomuksensa aikaiseksi. Tässä työssä käytetty Unity on erittäin aloittelijaystävällinen ja sen käyttöön löytyy runsaasti apuja verkosta. Unity ei myöskään laadullisesti häviä maksullisille ja paremmaksi pidetyille pelimoottoreille. Valmiin teoksen julkaiseminen on yksinkertaista ja eikä myöskään vaadi kehittäjiltä suuria resursseja. Nykyiset alustakohtaiset kauppapaikat antavat myös itsenäisille kehittäjille mahdollisuuden tienata peleillänsä helpommin.

LÄHTEET

Neogames: Alan toimijat. 2015. Luettu 17.7.2015. <http://www.neogames.fi/tietoa-toimialasta/alan-toimijat/>

Neogames: Tietoa toimialasta. 2015. Luettu 17.7.2015. <http://www.neogames.fi/tietoa-toimialasta/>

Hiltunen, K. Latva, S. Kaleva, P. 2013. Peliteollisuus kehityspolkuna. Tekes. Luettu 15.5.2015. http://www.tekes.fi/globalassets/julkaisut/peliteollisuus_kehityspolku.pdf

Crosby, T. 2008. How Making a Video Game Works. Luettu 15.5.2015. <http://electronics.howstuffworks.com/making-a-video-game.htm>

Takahashi, D. 2010. Angry Birds sells 6.5M units on iPhone and flies to new smartphones. Luettu 15.5.2015. <http://venturebeat.com/2010/08/13/angry-birds-sells-6-5m-units-on-iphone-and-flies-to-new-smartphones/>

Lappalainen, E. 2015. Cities: Skylines -hittipeliyhtiön johtaja: "Tavoitteena ei ole tehdä omistajille huikeita voittoja". Luettu 15.5.2015. <http://www.talouselama.fi/Kasvuyritykset/cities+skylines+hittipeliyhtion+johtaja+tavoitteena+ei+ole+tehda+omistajille+huikeita+voittoja/a2299203>

Suomen Pelinkehittäjät ry. 2010. Suomen pelitoimialan strategia 2010 -2015. Neogames. Luettu 15.5.2015. <http://www.neogames.fi/wp-content/uploads/2013/05/Pelistrategia-2010-2015.pdf>

Schubert, D. 2008. How to Write Great Design Documents. Luettu 15.5.2015. http://www.zenofdesign.com/wp-content/uploads/2014/12/Writing_Design_Docs_2008.pdf

Ward, J. 2008. What is a Game Engine?. GameCareerGuide. Luettu 10.6.2015. http://www.gamecareerguide.com/features/529/what_is_a_game.php?page=1

Unity: Public Relations. 2015. Unity3D. Luettu 10.6.2015. <http://unity3d.com/public-relations>

Windows Dev: Developing games on Windows 10. 2015. Luettu 10.6.2015. <https://dev.windows.com/en-us/games/develop>

Unity: Store. 2015. Unity3D. Luettu 10.6.2015. <https://store.unity3d.com/?currency=EUR>

Unity: Get Unity. 2015. Unity3D. Luettu 10.6.2015. <http://unity3d.com/get-unity>

Unity: Unity Pro and Unity Personal Software License Agreement 5.x. 2015. Unity3D. Luettu 10.6.2015. <http://unity3d.com/legal/eula>

Unity Manual: Hierarchy. 2015. Unity3D. Luettu 11.6.2015. <http://docs.unity3d.com/Manual/Hierarchy.html>

Unity Manual: Project Browser. 2015. Unity3D. Luettu 11.6.2015. <http://docs.unity3d.com/Manual/ProjectView.html>

Unity Manual: GameObjects. 2015. Unity3D. Luettu 11.6.2015. <http://docs.unity3d.com/Manual/GameObjects.html>

Unity Manual: Using Components. 2015. Unity3D. Luettu 11.6.2015. <http://docs.unity3d.com/Manual/UsingComponents.html>

Unity Manual: Prefabs. 2015. Unity3D. Luettu 11.6.2015. <http://docs.unity3d.com/Manual/Prefabs.html>

Unity Manual: Scripting Section. 2015. Unity3D. Luettu 11.6.2015. <http://docs.unity3d.com/Manual/ScriptingSection.html>

Unity Manual: Scripting. 2012. Unity3D. Luettu 11.6.2015. <http://docs.unity3d.com/401/Documentation/Manual/Scripting.html>

Unity Script Reference: MonoBehaviour. 2015. Unity3D. Luettu 11.6.2015. <http://docs.unity3d.com/ScriptReference/MonoBehaviour.html>

Unity Script Reference: MonoBehaviour.Start. 2015. Unity3D. Luettu 11.6.2015. <http://docs.unity3d.com/ScriptReference/MonoBehaviour.Start.html>

Unity Script Reference: MonoBehaviour.Update. 2015. Unity3D. Luettu 11.6.2015. <http://docs.unity3d.com/ScriptReference/MonoBehaviour.Update.html>

Unity Script Reference: MonoBehaviour.FixedUpdate. 2015. Unity3D. Luettu 11.6.2015. <http://docs.unity3d.com/ScriptReference/MonoBehaviour.FixedUpdate.html>

Unity Script Reference: PlayerPrefs. 2015. Unity3D. Luettu 11.6.2015. <http://docs.unity3d.com/ScriptReference/PlayerPrefs.html>

Microsoft: Getting Paid. 2015. Luettu 26.7.2015. <https://msdn.microsoft.com/en-us/library/windows/apps/mt148536.aspx>

Microsoft: Account types, locations, and fees. 2015. Luettu 24.7.2015. <https://msdn.microsoft.com/en-us/library/windows/apps/jj863494.aspx>

Microsoft: Setting up your payout account and tax forms. 2015. Luettu 24.7.2015. <https://msdn.microsoft.com/en-us/library/windows/apps/bg124529.aspx>

Microsoft: Windows App Certification Kit tests for Windows Store apps. 2015. Luettu 24.7.2015. <https://msdn.microsoft.com/en-us/library/windows/apps/jj657973.aspx>

Microsoft: App submissions. 2015. Luettu 25.7.2015. <https://msdn.microsoft.com/library/windows/apps/hh694062.aspx>

Microsoft: Windows Store Policies. 2015. Luettu 25.7.2015. <https://msdn.microsoft.com/en-us/library/windows/apps/dn764944.aspx>