Marko Haaja

TILTAGON 2.0-PELIN KÄÄNTÄMINEN WINDOWS-ALUSTALLE

Opinnäytetyö

Tietotekniikka / Peliohjelmointi

Elokuu 2016

KYAMK

University of Applied Sciences

KYAMK
University of Applied Sciences

Tekijä/Tekijät	Tutkinto	Aika
Marko Haaja	Insinööri	Elokuu 2016

Opinnäytetyön nimi

Tiltagon 2.0-pelin kääntäminen Windows-alustalle — 51 sivua

Toimeksiantaja

Kiemura

Ohjaaja

Lehtori Niina Salmi

Tiivistelmä

Opinnäytetyön tavoitteena oli tehdä käännöstyö peliyritys Kiemuran Tiltagon-nimisen pelin päivitykselle. Tiltagonin ideana on vierittää palloa läpi vaihtelevien tasojen muuttamalla puhelimen asentoa. Peli on tehty monelle eri alustalle ja se ylitti miljoonan latauksen rajan jo kesällä 2015.

Projektissa käytettiin monia erilaisia työkaluja. Tärkeimpinä olivat Microsoft Visual Studio ja Tiltagonin pelimoottorina toimiva Unity3D sekä sen plug-init eli liitännäiset. Plug-inejä ja ohjeita toimeksiantaja laittoi pääasiassa Google Driveen, joka toimi sähköpostin ohella tärkeänä tiedostojen jakamiseen käytettävänä apuvälineenä. Prosessinhallintaan projektissa käytettiin SourceTree-nimistä ohjelmaa.

Projektin pääpaino oli mainosverkkojen integroinnissa. Tämä oli myös kaikkein työläin osuus, koska Windows-alustalla on puhelimille hiipuvat markkinat. Tämä tarkoittaa myös sitä, että liitännäisiä ja mainosverkkoja ei tehdä Windows-puhelimille monenkaan kehittäjän toimesta. Muita keskeisiä osa-alueita projektissa oli sosiaalisen median liittäminen peliin, pelin sisäiset ostot sekä liitännäisten uudelleensovittaminen projektiin.

Lopulta ennen työtä sovitut tehtävät saatiin valmiiksi. Tiltagonin päivitys on nyt viimeistelyä vaille valmis lisättäväksi Windows Storeen mukanaan kaikki sille vaaditut ominaisuudet. Oma osaamiseni kehittyi ja osaamisalue laajentui erityisesti Microsoft Visual Studion ja Unity3d:n välisen kommunikoinnin ymmärtämisessä. Myös tietoisuus monetisaatiosta, eli rahan tienaamisesta pelien avulla kasvoi huomattavasti.

Asiasanat

Tiltagon, Unity3d, Windows Store, monetisaatio, Microsoft Visual Studio, SourceTree, liitännäinen, Windows Phone

KYAMK
University of Applied Sciences

Author (authors)	Degree	Time
Marko Haaja	Bachelor of Engineering	August 2016

Thesis Title

Porting a Tiltagon 2.0 update to Windows Phone — 51 pages

Commissioned by

Kiemura

Supervisor

Niina Salmi, Senior Lecturer

Abstract

Objective for this thesis was to port an update for a game named Tiltagon. Tiltagon is an arcade game where player has to roll the ball through variable platforms by tilting the phone to the direction where the ball should go. The game has been designed for multiple different platforms and it hit the marker for one million downloads in summer 2015.

Several tools, mainly Microsoft Visual Studio and Unity3d with its plug-ins, were used in the project. Unity is the game engine used in Tiltagon. Plug-ins and directives were added to either Google Drive or email by the thesis subscriber. SourceTree was used in process control.

Ad network integration was the primary stress in this project. This was also the most laborious part because the market for Windows Phone is fading. This also means that there are not many developers that focuses on the plug-ins and ad networks for Windows Phone. Other essential parts for this update were the adding of social media and in app purchases along with the reinstalling the plug-ins.

Eventually all the pre-prescribed tasks were finished. Update for Tiltagon is now nearly finished for being ready to be released in the Windows Store with all of its required features. Project was mostly about communication between Microsoft Visual Studio and Unity3D. Project was successful for both me and the subscriber. Works were finished and I gained valuable information.

Keywords

Tiltagon, Unity3d, Windows Store, monetization, Microsoft Visual Studio, SourceTree, plug-in, Windows Phone

SISÄLLYS

TERMIT JA LYHENTEET

Building	Buildingilla tarkoitetaan ohjelman irtonaisten osien kasaamista yhdeksi kokonaisuudeksi. Buildingilla koottua projektia sanotaan buildiksi.
C#	C# on yksi käytetyimmistä ohjelmointikielistä. Kieli on oliopohjainen ja luotiin yhdistämään helppokäyttöisyys ja tehokkuus.
Debuggeri	Tietokoneohjelma, jota käytetään kehitettävissä sovelluksissa virheiden jäljittämiseen ja korjaamiseen.
DLL	DLL (dynamic-link library) on Microsoftin toteutus kirjastolle, joka sisältää koodia sekä tietoa, ja jota pystyy käyttämään useampi ohjelma samaan aikaan. Nämä kirjastot toimivat Microsoft Visual Studio -ohjelman liitännäisinä.
IAP	IAP (in-app purchase) eli pelin sisäinen osto on termi, jota käytetään mobiilisovelluksissa tapahtuvista ostoista, jolla saa peliin joko hyötyä tai visuaalista muutosta. IAP on mobiilipelien suurimpia tulonlähteitä.
IntelliSense	Tehokkuutta parantava koodin apuväline, joka yrittää arvata, mitä käyttäjä on kirjoittamassa.
Käyttöliittymä	Käyttäjän näytöllä näkyvä rajapinta, jolla käyttäjä on yhteydessä sovelluksen toimintaan.
Liitännäinen	Liitännäinen (plug-in) on kolmannen osapuolen tekemä ohjelmiston osa, joka parantaa sovellusten ominaisuuksia.
Monetisaatio	Rahan ansainnan keinot. Peleissä nämä ovat esimerkiksi pelin sisäiset ostot sekä mainokset.
Pelimoottori	Ohjelmointiympäristö, joka on suunniteltu pyörittämään pelien kehittämistä.

SDK	SDK (software development-kit) on sovellusten kehittämiseen tarkoitettu työkalu.
Skripti	Kooditiedosto, joka sisältää komentoja, joilla määritetään pelin toimintoja.
WNS-ilmoitukset	Kehittäjien tiedonlähetystapa sovelluksiinsa
WSA	WSA (Windows Store application) on Microsoftin alusta Windows 8.1:tä uudemmille laitteille. WSA:lle suuntautuessa loppuu WP8:n eli Windows Phone 8:n tuki. Unity3D-pelimoottorissa WSA:lle tulee monia lisäominaisuuksia WP8:aan nähden, kuten videomainokset.

1 JOHDANTO

Kiemura on 2015 perustettu pelialan yritys, jonka perustajina toimivat kotkalaiset Jyri ja Piia Kilpeläinen. Heidän kehittämä ja kanadalaisen Noodlecake Studiosin julkaisema Tiltagon on puhelimen kallisteluun perustuva pulmanratkontapeli (Kiemura 2016). Pelissä on kaksi pelitilaa. Peliä voi mennä taso kerrallaan läpi tai vaihtoehtoisesti siinä on myös loputon peli. Tässä pelataan niin kauan kunnes tippuu tasolta. (Tiltagon 2016.)

Opinnäytetyön tavoitteena oli tehdä jo Androidille ja iOS-käyttöjärjestelmälle tehdyn päivityksen kääntäminen Windows Phone -käyttöjärjestelmälle. Päivityksessä tehtyihin uudistuksiin lukeutui peliin lisätty tasotila, uusintojen hallinta, tuloslistat sekä monia käyttäjäkokemuksiin liittyvää uutta materiaalia, joista tärkeimpinä 36 uutta erilaista palloa sekä uudenlaiset tasot.

Itse päivityksessä keskityttiin vain asioihin, jotka eivät toimineet suoraan Android ja iOS-versiosta muutettaessa Windows Phonelle. Näihin lukeutui moni pelimoottori Unity3D:n liittännäisistä. Nämä pitivät asentaa uudelleen, sekä osaa niihin liittyvistä koodeista täytyi kirjoittaa uudelleen Windows Phonelle soveltuviksi. Tämän lisäksi projektiin piti lisätä pelin sisäisten ostojen mahdollisuus sekä integroida videomainokset ja koko ruudun mainokset.

Työ toteutettiin Unity3D-pelimoottorilla sekä Microsoftin ohjelmankehitysympäristöllä nimeltä Microsoft Visual Studio. Versionhallintaan käytetty SourceTree auttoi mahdollisiin virheisiin, koska sen sovelluksesta pystyi lataamaan edellispäivien virheettömät versiot, eikä koko projektia tarvinnut haravoida läpi etsittäessä virheiden lähdettä.

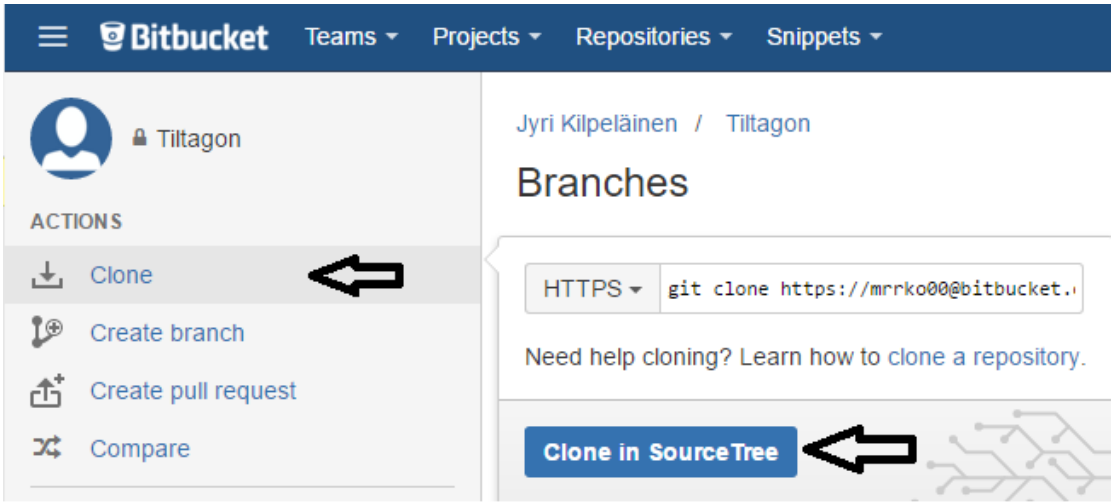
2 KEHITYSYMPÄRISTÖ

Projektin tekeminen keskittyi pääosin Unity3D:hen ja Microsoft Visual Studioon sekä niiden väliseen kommunikointiin, mutta apuvälineitä oli muitakin. Osa apuvälineistä oli edellä mainittujen ohjelmien sisällä olevia liitännäisiä, osa itsenäisiä ohjelmia ja laitteita.

2.1 SourceTree

SourceTree on Atlassianin tekemä ohjelmisto, jota käytetään versionhallintaan. Versionhallinta mahdollistaa projektin virheiden minimoimisen, koska sen avulla projektista voi ongelmien sattuessa ladata edellisten päivien toimivat versiot. (SourceTree 2016)

SourceTreen käytön aloittamiseksi on tehtävä tili Bitbucket-nimiselle verkkosivulle, joka isännöi SourceTreen palveluja netin välityksellä. Bitbucketissa on ladattavissa pohjakoodi, jonka tilaaja oli sinne laittanut. Ennen pohjakoodin lataamista SourceTreehen on valittava haara (branch). Haaroista ylin on master-haara. Master on versionhallinnassa aina se haara, mihin alemmat haarat lopuksi yhdistetään niiden valmistuttua. Projektin lataamiseen on valittava ensin oikea haara, jonka jälkeen valitaan Clone ja Clone in SourceTree. Tämä käynnistää SourceTreen ja lataa projektin (kuva 1).



Kuva 1. Projektin kloonaaminen SourceTreehen

SourceTreehen pitää Bitbucketin lisäksi vielä erikseen kirjautua sisään. Ohjelman toimimaansaamiseksi on käyttäjän vielä täytettävä omat tietonsa. Ne saa täytettyä kohdasta Tools->Options->General.

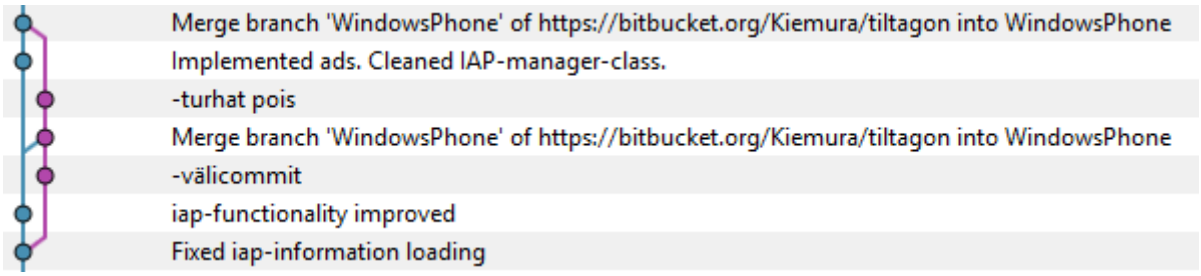
SourceTreen toimintoja:

- Clone/New: Uuden ”repositoryn” (säilytyspaikka) luonti
- Commit: Tiedostomuutosten lisäys
- Discard: Tiedostojen poisto
- Stash: Muutettujen tiedostojen piilottaminen
- Fetch: Muiden henkilöiden committien hakeminen
- Pull: Korvaa omat tiedostot muiden henkilöiden commiteilla
- Push: Työntää oman commitin muille näkyviin
- Branch: Tekee uuden haaran olemassa olevaan haaraan
- Merge: Yhdistää nykyisen haaran ylempään haaraan

SourceTreessä näkee lisäyksen jälkeen, mitä projektissa on muutettu. Vihreällä on merkitty projektiin lisätyt, punaisella poistetut ja keltaisella muokatut tiedostot (kuva 2).

Kuva 2. Commit, missä näkyy lisäyksen jälkeiset muutokset

SourceTreessä pystyy myös tekemään monia erillisiä haaroja, jotka lopulta yhdistetään. Tämä mahdollistaa työmäärän jakamisen eri osa-alueisiin, jotka säännöllisesti sitten yhdistetään samaan haaraan (kuva 3)



Kuva 3. SourceTreen haarojen erotus ja yhdistäminen

2.2 Unity3D

Unity3D on ilmainen ja monialustainen pelimoottori, jonka ensimmäinen versio julkaistiin jo 2005. Projektin tekoon käytetyt versiot olivat 5.3.2f1 ja 5.2.4f1. Unity tukee kolmea ohjelmointikieltä: vähän käytettyä Boota, Javascriptiä sekä yleisimmin käytettyä C#:ia. Unity tukee yli kahtakymmentä alustaa mukaan

lukien suurimmat konsolialustat, käyttöjärjestelmät sekä mobiilikäyttöjärjestelmät.

Unityn päänäytössä on projekti jaettuna eri osiin. Projektin sisäisten tiedostojen näkymässä kaikki projektin pelikomponentit (assets) on järjestetty kansioihin. Pelikomponentteihin kuuluu projektiin kuuluvat visuaaliset tiedostot, ääniefektit, pelin kohtaukset (scenes), liitännäiset (plug-ins), pelin skriptit sekä muut peliä edistävät tiedostot. Projektin peliobjektit on omassa hierarkiassaan Unityn päänäytössä. Peliobjektit voidaan siellä lisätä toisen objektin sisälle, jolloin kaikki hierarkiassa alempana olevat peliobjektit perivät ylempänä olevan ominaisuudet. Objektien tarkastelua varten näkymässä on myös oma osio. Kohtausnäkymä on pelin esikatselua varten. Peliä pystyy myös pelaamaan Unityllä, jota varten siinä on kohtausnäkymä. Pelin virheet ja pelin tarkasteluun tehtävät lokimerkinnät näkyvät Unityn konsolissa (kuva 4).

Kuva 4. Unityn perusnäkymä

Skriptin luomisessa on otettava huomioon, että skripti täytyy olla liitettynä peliobjektiin toimiakseen. Koodin jäädessä jumiin on Debug.Log-komento hyödyllinen. Tämä komento antaa viestin Unityn konsoliin, jos ehdot komennolle ovat täyttyneet (kuva 5).

```
Debug.Log("....----Adspot id changed, Type set to interstitial----....");
CacheInterstitial(); //Hakee mainosta heti pelin alussa
Debug.Log("....----Cache started----....");
```

Kuva 5. Debug.Log-esimerkki. Lokiin laitettu merkintä ennen funktiota ja sen jälkeen

OnEnable()- ja OnDisable()-funktiot käynnistyvät objektin sallimisen mukaan. Niiden tapahtumat (events) aktivoivat niihin liitetyt funktiot, jotka aloittavat oman toimintonsa (kuva 6).

```
void OnEnable()
{
    /*InAppPurchaseManager.OnPurchaseCompleted += OnPurchaseCompleted;*/
    GameManager.OnGameOverCompleted += OnGameOverCompleted;
    UIManager.OnGameStarted += OnGameStarted;
    UIManager.OnMenuOpened += OnMenuOpened;
    UIManager.OnMenuClosed += OnMenuClosed;
    UIManager.OnInventoryOpened += OnInventoryOpened;
    UIManager.OnInventoryClosed += OnInventoryClosed;
}

void OnDisable()
{
    /*InAppPurchaseManager.OnPurchaseCompleted -= OnPurchaseCompleted;*/
    GameManager.OnGameOverCompleted -= OnGameOverCompleted;
    UIManager.OnGameStarted -= OnGameStarted;
    UIManager.OnMenuOpened -= OnMenuOpened;
    UIManager.OnMenuClosed -= OnMenuClosed;
    UIManager.OnInventoryOpened -= OnInventoryOpened;
    UIManager.OnInventoryClosed -= OnInventoryClosed;
```

Kuva 6. OnEnable- ja OnDisable-funktiot

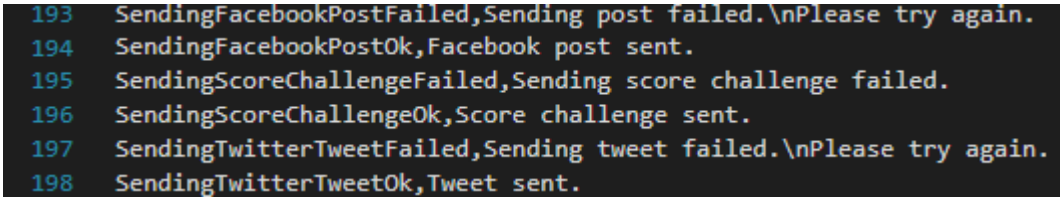
2.2.1 Unityn liitännäiset

Unityllä on laaja valikoima Unityn käyttöä helpottavia liitännäisiä. Liitännäisten ideana on projektin nopeutus. Suurin osa niistä maksaa, mutta niiden hinta korvaantuu nopeasti ajassa, joka niiden tekoon menisi. Unityllä on oma valikoima liitännäisiä Asset Storessa. Keskeneräistä projektia siirrettäessä päätteeltä toiselle oli liitännäiset yleensä asennettava uudelleen, jotta ne toimivat täysin. (Unity Asset Store 2016a.)

LeanTween on ilmainen tween-moottori. Tämä helpottaa Unityn peleissä tapahtuvaa animaatiota. LeanTween helpottaa myös skriptien silmukoiden tekemistä sekä pelin paussille laittamista, koska siinä on esimerkkikohtauksia ja –skriptejä, joista voi ottaa mallia. (Unity Asset Store 2016b.)

NGUI on todella suosittu käyttöliittymän liitännäinen. Suosio NGUIssa perustuu siihen, että se tukee kaikkia samoja alustoja kuin Unity. NGUI:ssa on kaikkiin sen ominaisuuksiin selkeät esimerkkikohtaukset ja ominaisuuksista suuressa käytössä projektissa oli NGUIn mahdollistama lokalisaatio. (Unity Asset Store 2016c.)

Lokalisaatio helpottaa pelin kääntämistä muille kielille. Lokalisaation avulla skripti hakee tekstitiedostosta hakusanalla tarvittavan tekstin. Tarvittaessa sillä voi muuttaa kielenkin, mutta se helpottaa myös Tiltagonin kaltaisen yksikielisen pelin tekstien asettamista (kuva 7)



Kuva 7. Lokalisaatio-esimerkki. Ennen pilkkua hakusana, pilkun jälkeen asetettava teksti

Lokalisaatio haetaan skriptistä komennolla: Localization.Get(”hakusana”);.

Prime31 on monipuolinen, pääosin mobiilikäyttöjärjestelmiin keskittyvä liitänäisten valmistaja. Prime31 tekee liitännäisiä mainoksien integrointiin, sosiaalisen median lisäämiseen sekä pelin sisäisiin ostoihin. Prime31 luopui Windows Phone 8:n tuesta, joten osa projektista yritettiin välissä vaihtaa muilla liitännäisillä toimiviksi, mutta lopulta saatiin Prime31 toimimaan vanhan Tiltagonin ensimmäisen version liitännäisellä. (Prime31 2016a.)

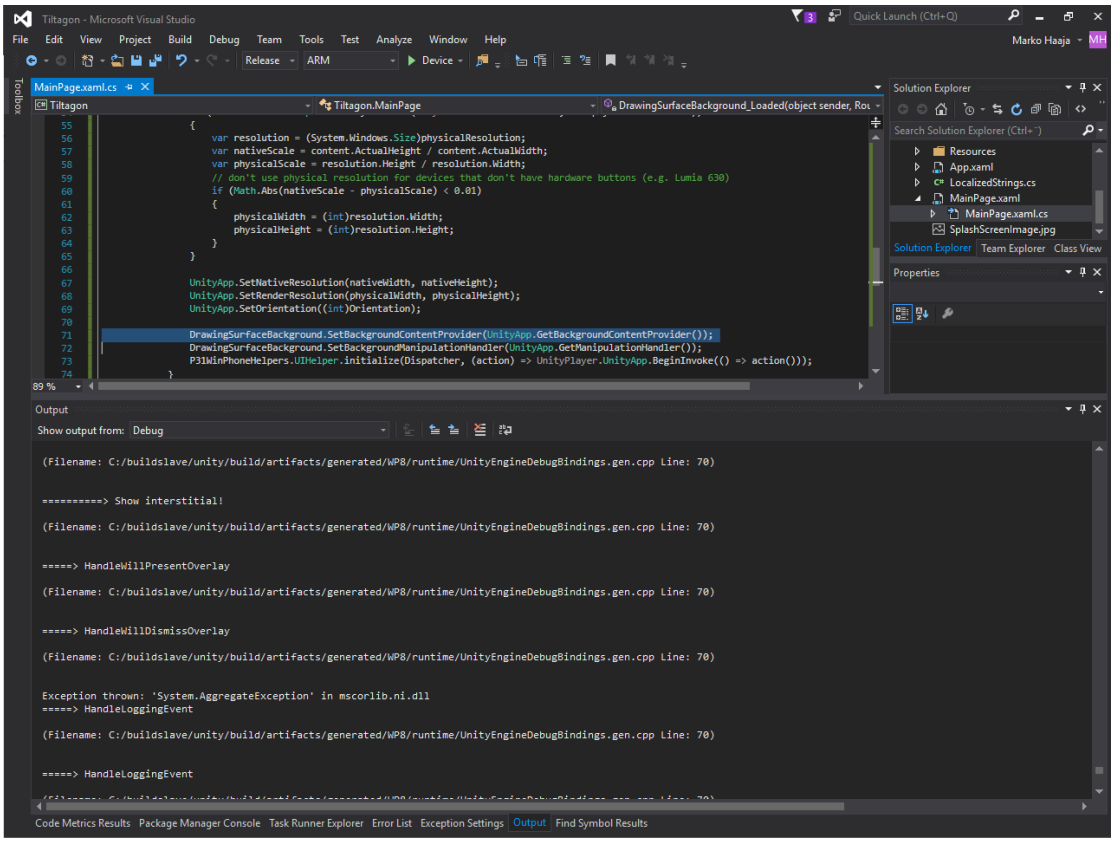
UniRate on mainostamiseen tarkoitettu liitännäinen. Sen ideana on parantaa pelin arvosteluja kaupoissa parantamalla sovelluksen arvioinnin ajankohtaa. UniRaten avulla sovellus kysyy käyttäjää arvioimaan sovelluksen vasta, kun sitä on käytetty paljon. Tämän takia tulee oletettavasti enemmän positiivisia arvioita, mikä johtaa suurempaan latausmäärään. (Unity Asset Store 2016d.)

Vungle on mainosverkko sekä liitännäinen, joka keskittyy videomainoksiin jotka avaavat pelissä ominaisuuksia ja palkintoja. Vunglen videomainokset tukevat vain Windows Phonen versioita 8.1:stä lähtien, joten Vunglen videomainokset jäivät pelin lopullisesta versiosta pois (Vungle 2016).

Easy Save 2 on tiedostojen lataamiseen, tallentamiseen, jaksottamiseen, salaamiseen ja varastoimiseen käytettävä Unityn liitännäinen. Unityn omaan tallennusjärjestelmään, PlayerPrefsiin verratessa Easy Save 2 on kevyempi ja nopeampi. (Unity Asset Store 2016e.)

2.3 Microsoft Visual Studio

Microsoft Visual Studio on Microsoftin kehittämä ohjelmankehitysympäristö, joka tukee monia ohjelmointikieliä. Unityltä siirrettäessä projekti on kirjoitettu C#-kielellä, kuten Unityssäkin. Visual Studio sisältää IntelliSenseä tukevan koodieditorin sekä integroidun debuggerin, joka toimii sekä lähdetasolla, että konetasolla. Visual Studio sisältää myös monia työkaluja ja se tukee monipuolisesti liitännäisiä, mitkä parantavat sen toimintaa (kuva 8). Ohjelmaa voi käyttää niin yksityinen harrastaja kuin yrityksetkin. (Visual Studio 2016.)

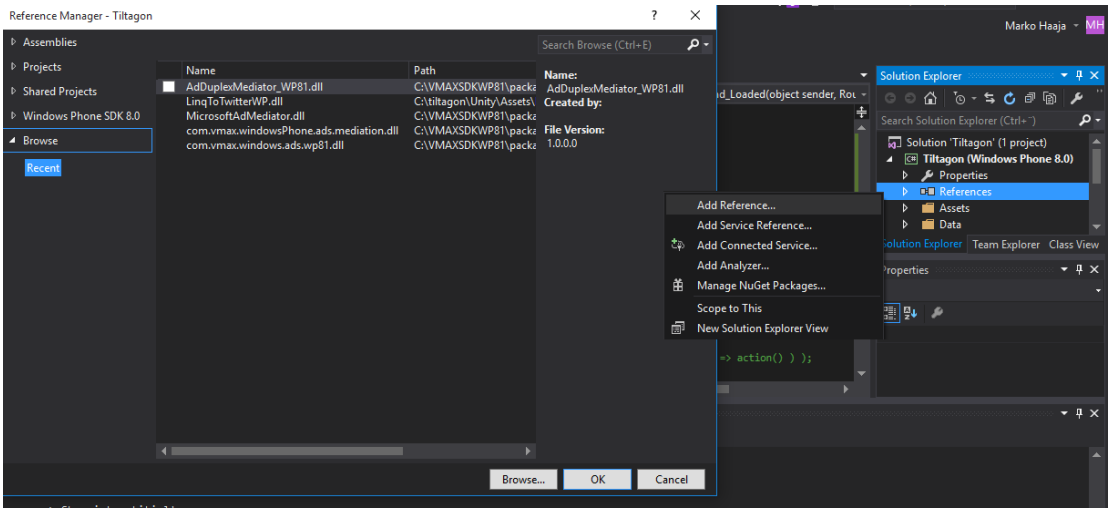


Kuva 8. Visual Studion perusnäkymä, jossa on koodieditori, debug-näyttö sekä erilaisia työkalupalkkeja

2.3.1 Visual Studion viitteet

Visual Studion viitteillä tarkoitetaan erilaisia kirjastoja, mitä projektiin voi lisätä. Kirjastot ovat Visual Studion laajennuksia, jotka eivät ole käytettävissä jos ei niitä erikseen lisää. Tämä keventää projekteja, koska siihen ei tarvitse liittää ominaisuuksia, joita ei itse tarvitse. Viitteen voi lisätä Solution Explorer -kohdasta klikkaamalla kohtaa References oikealla hiiren napilla, ja valita valikosta Add Reference (kuva 9). Viitteen voi myös lisätä NuGet-menetelmän

avulla. NuGet on avoimen lähdekoodin pakkauksenkäsittelyjärjestelmä, joka pyrkii yksinkertaistamaan kirjastojen lisäämistä projektiin. (Microsoft Developer Network 2016a.)

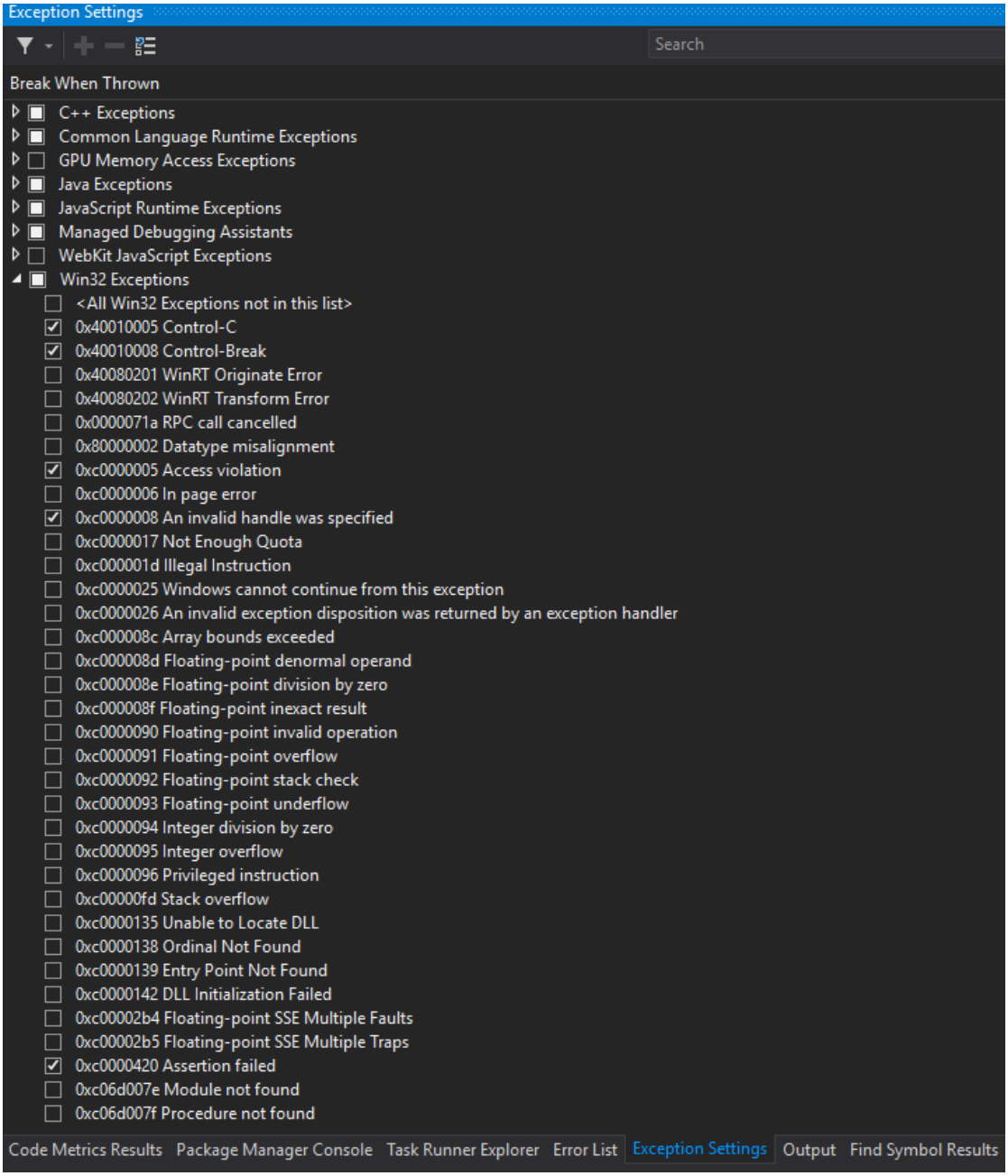


Kuva 9. Viitteiden lisääminen sekä Reference Manager

2.3.2 Projektin building-vaihe

Projektin ollessa valmis buildaukseen pitää valita kohdelaite sekä buildin laatu, jossa vaihtoehtoja on kolme: debug, release ja master. Debug-buildi näyttää projektin ulostulossa kaiken, mitään karsimatta. Se on ainoa build-tyyppi, jonka kautta näkee symboliset debug-tiedot, ja jonka avulla näkee, missä kohtaa koodia debuggeri toimii. Release-buildi näyttää myös Unityn projektissa lisättyjä Debug.Log-komentoja, mutta tämän lisäksi release on täysin optimoitu eli se on karsinut koodia, mikä nopeuttaa buildin toimimista. Master-buildi on julkaisuvalmis buildi, mikä on optimoitu ja kaikki debug-viestit poistettu. (Unity Technologies 2016a.)

Projektia debugatessa tärkeä työkalu on Exception Settings. Exception Settings -valikosta pystyy valitsemaan, minkä virheetyypin kohdalla projekti pysäyttää toimintansa ja tarkistaa tarkemmin onko kyseiseen ongelmaan tarjolla ratkaisua Visual Studiossa itsessään (Microsoft Developer Network 2016b). Exception Settings-valikko löytyy debug-näytön alareunasta (kuva 10).

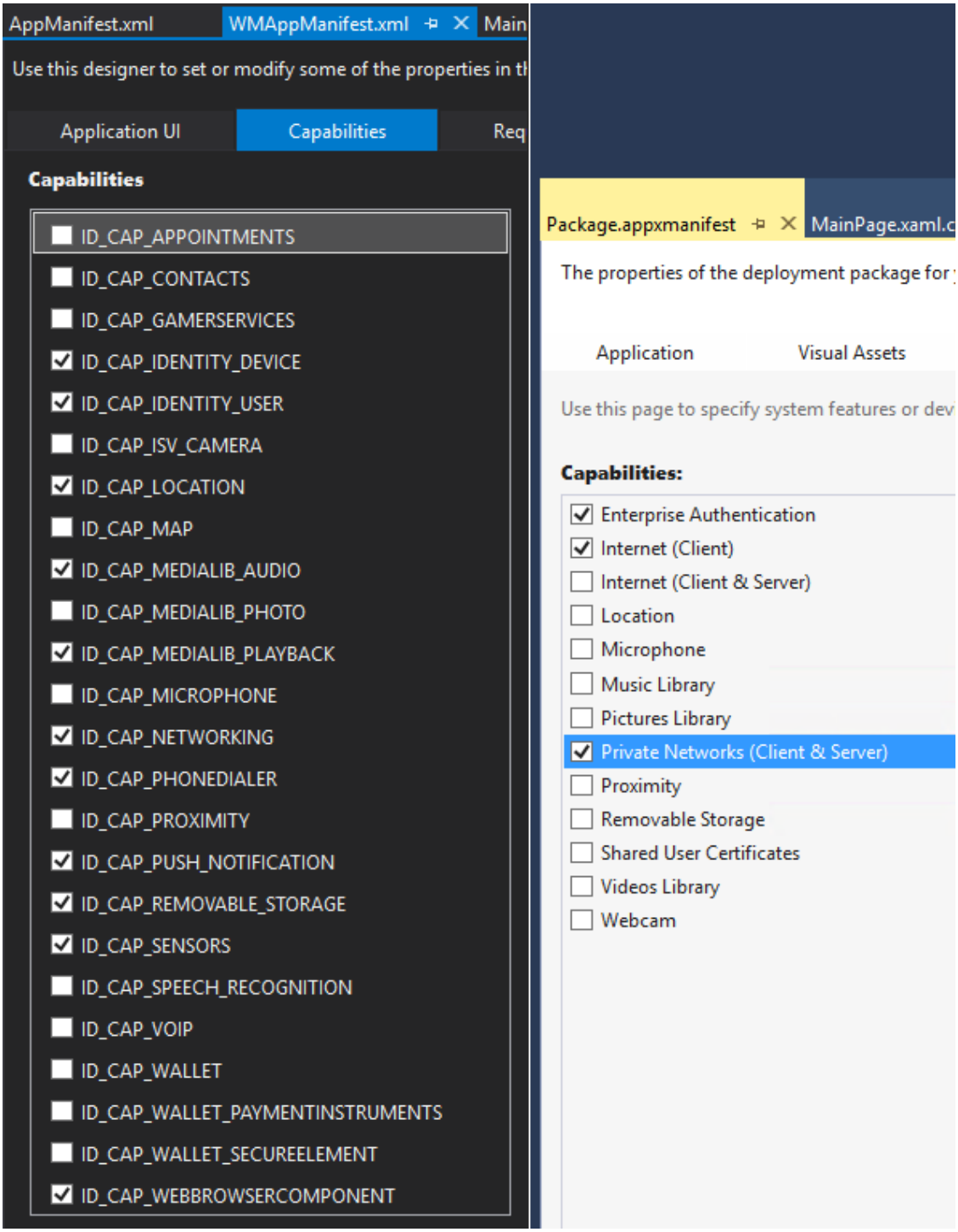


Kuva 10. Exception Settings -valikko

2.3.3 Appxmanifest ja WMAppManifest

Appxmanifest ja WMAppManifest ovat projektin .xml-päätteisiä tiedostoja, joissa määritetään projektin sisäisiä asetuksia, jotka ovat erityisen tärkeitä projektin julkaisuvaiheessa sekä eri monetisaatiotapojen liittämisessä peliin, koska tiedostossa määritettävät id:t pitävät täsmätä serverillä olevien kanssa, jotta ne voivat kommunikoida keskenään. (Järvinen 2012, 224-225.) Appxmanifest ja WMAppManifest on samaan tarkoitukseen käytettävä tiedosto, mutta Appxmanifest on uudempi, WSA-projektille suunnattu tiedosto,

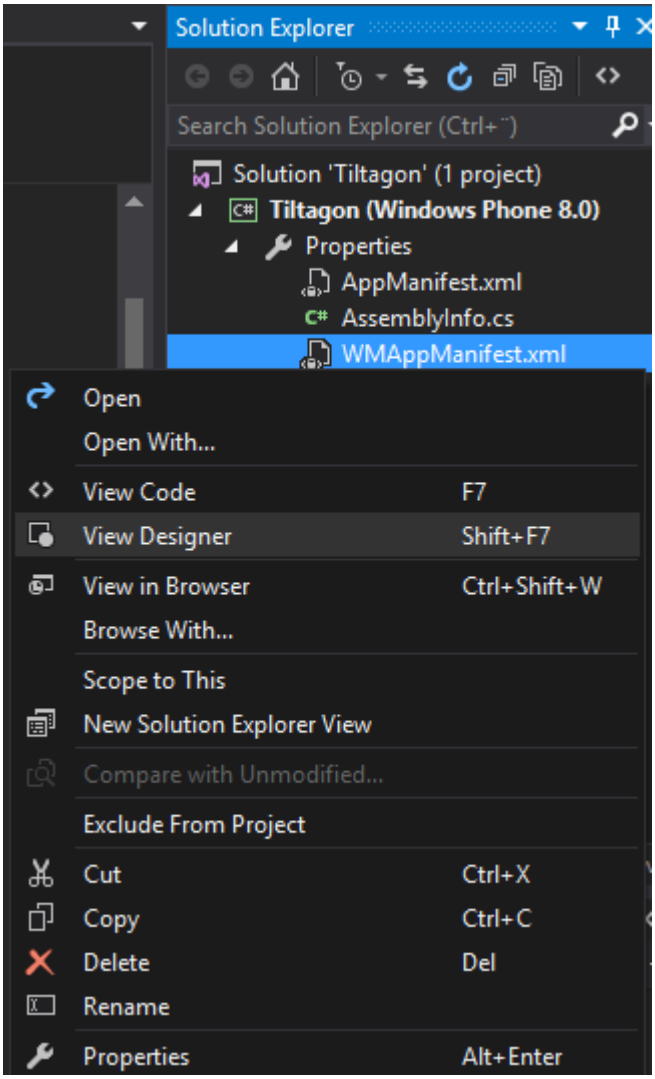
missä osa valinnoista on selkeytetty Windows Phonen vanhempaan WMAppManifestiin verrattuna (kuva 11).



Kuva 11. Capabilites-välilehti WMAppManifestissa ja Appxmanifestissa

Appxmanifestiin ja WMAppManifestiin suurimman osan tiedoista voi lisätä avaamalla Visual Studion kautta tiedoston designer Solution Explorerista näpäyttämällä oikeaa hiirenpainiketta valitun tiedoston kohdasta ja valitsemalla View Designer -kohta. Joitain tietoja kuitenkin näkee vain koodin

kautta, joka avataan muuten samalla tavalla, mutta valitaan valikosta View Code (kuva 12).



Kuva 12. WMAppManifest-tiedoston koodin ja designerin avaaminen

Appxmanifestissa ja WMAppManifestissa määritettävät tiedot:

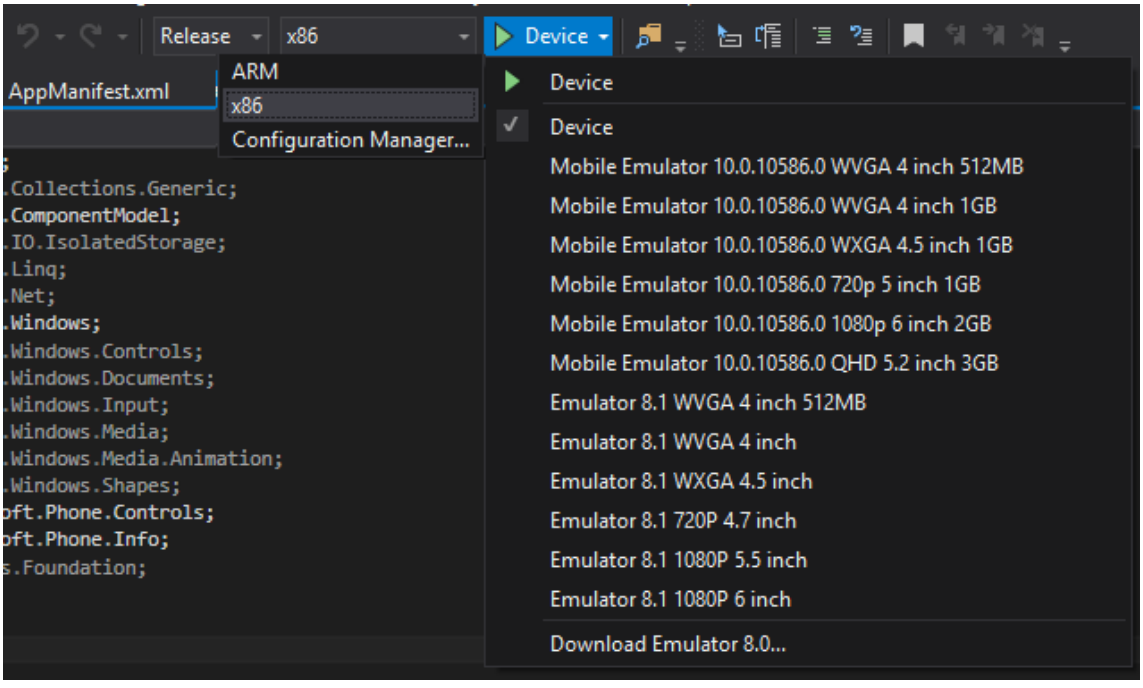
- Display Name: Pelin nimi
- Package Name: Koko projektin paketin nimi
- Product ID: Projektin id, joka on myös URL-lataussivun loppuosa
- Publisher ID: Julkaisijan id
- Author: Tekijän nimi
- Publisher Display Name: Julkaisijan nimi
- Package SID: Security Identifier, eli id, jonka avulla peliin lähetetään WNS-ilmoituksia
- Version: Version numero
- Capabilities: Eri ominaisuuksia, joita peliin voi liittää

Pelin käyttämät ominaisuudet:

- IDENTITY_DEVICE: Antaa pääsyn käytetyn laitteen id:lle ja mallille
- IDENTITY_USER: Tunnistaa käyttäjän Microsoft-tilin kautta
- LOCATION: Antaa käyttäjän sijainnin
- MEDIALIB_AUDIO: Antaa pääsyn audio-ominaisuuksille mediakirjastossa
- MEDIALIB_PLAYBACK: Antaa pääsyn medialle, joka on samaan aikaan päällä
- NETWORKING: Sallii internetin käytön pelissä
- PHONEDIALER: Antaa tuen mainosten soitto-ominaisuuksiin
- PUSH_NOTIFICATIONS: Sallii ilmoitukset sovellukseen
- REMOVABLE_STORAGE: Antaa pääsyn erilliselle säilytystilalle kuten puhelimen SD-kortille
- SENSORS: Sallii Windows Phonen sensoreiden käytön
- WEBBROWSERCOMPONENT: Antaa pääsyn verkkoselaimelle

2.4 Testilaitteet (CPU)

Testilaitteeksi saatiin koululta kolme Windows Phone 8.1 -käyttöjärjestelmän omaavaa Nokia Lumia -puhelinta. Niiden versiomallit olivat 630, 635 ja 735. Tämän lisäksi opinnäytetyön tilaaja antoi kaksi testipuhelinta ja Microsoft Visual Studiossa on oma emulaattori, jossa on samat ominaisuudet kuin puhelimessa, vaikkei siinä varsinaista matkapuhelinverkkoa olekaan. (Järvinen 2012, 87-90.) Emulaattorissa on monipuolisesti erilaisia versioita Windows Phone -laitteista (kuva 13).



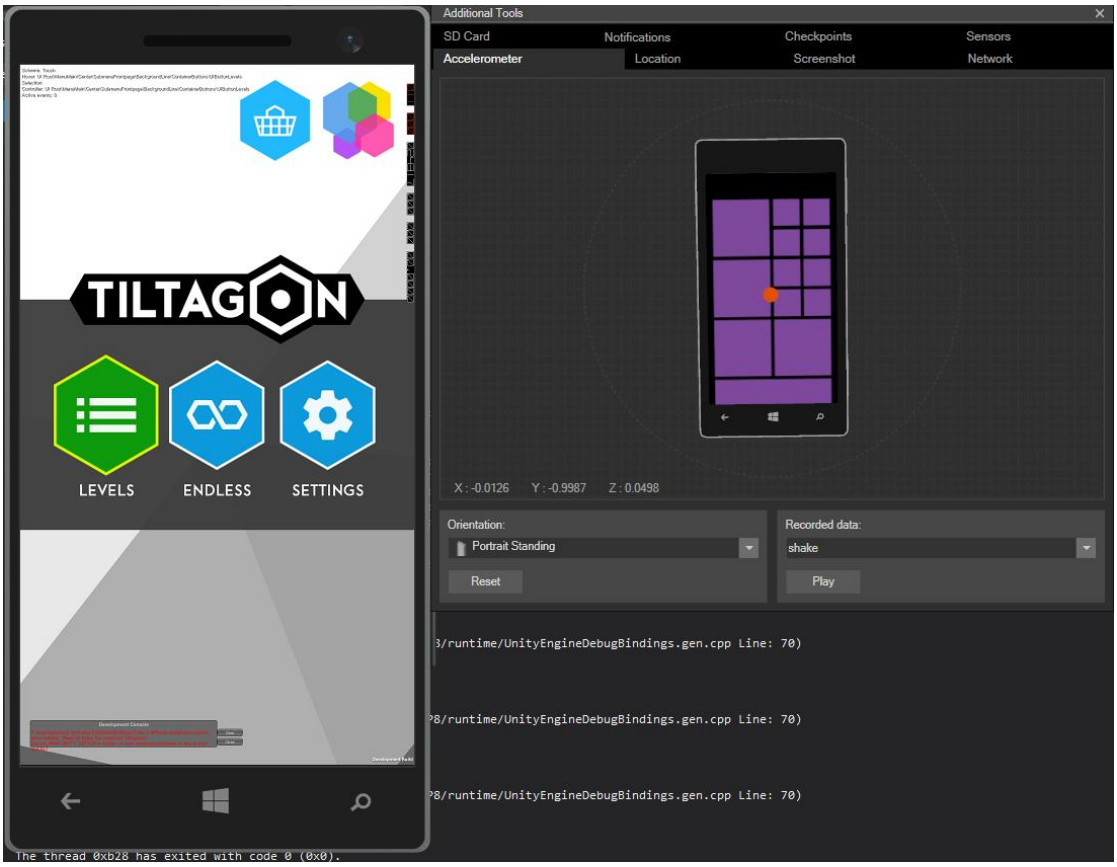
Kuva 13. Visual Studion emulaattorilista

2.4.1 ARM-testilaitteet

ARM (Advanced RISC Machines) on 32-bittinen suoritinarkkitehtuuri, jota käytetään matkapuhelimissa. Projektissa ARM-laitteita olivat kaikki fyysiset testilaitteet. Testilaitteissa oli oletuksena poissa päältä kehittäjän tila (developer mode), mikä mahdollistaa kehitysvaiheessa olevien ohjelmistojen testaamisen laitteella. Laitteissa sai olla enintään kaksi kehitysvaiheessa olevaa ohjelmistoa. (Techopedia 2016.)

2.4.2 x86-emulaattorit

Windows Phone -laitteille kehittäessä voi testaamiseen käyttää myös Visual Studion omia emulaattoreita. Windows Phonen eri malleja mallintavat x86-suoritinarkkitehtuurin emulaattorit voivat mallintaa monia älylaitteiden ominaisuuksia, kuten laitteen kallistusta (kuva 14).



Kuva 14. Emulaattori ja sen kallistusominaisuus

3 ESIVALMISTELU

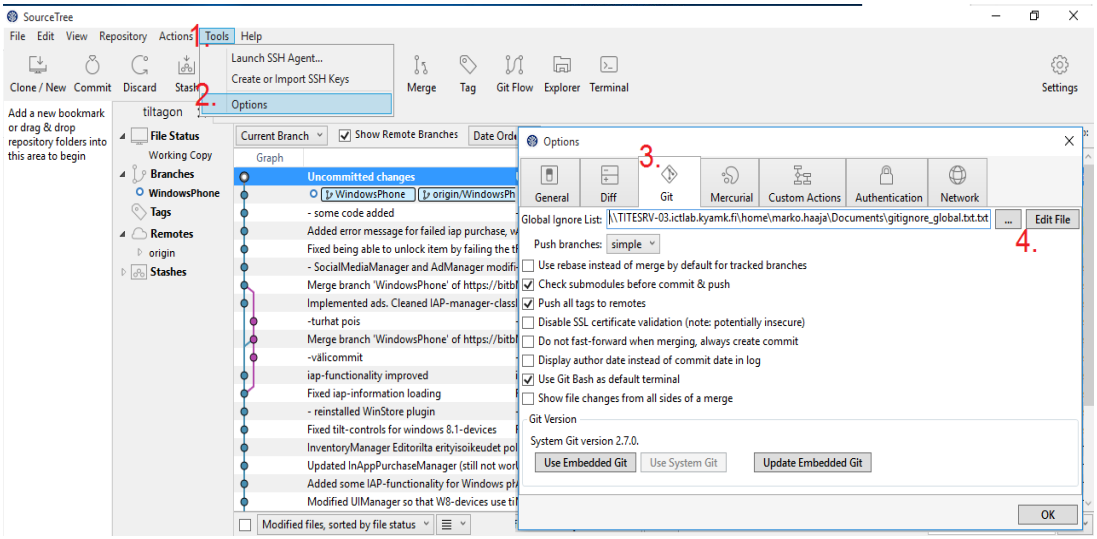
Ennen projektin aloitusta toimeksiantajan kanssa sovittiin projektin laajuus ja tehtävät. Tämän lisäksi allekirjoitettiin salassapitosopimukset. Pohjaksi annettu Tiltagon 2.0:n Android- ja iOS-alustoille suunnattu projekti vaati monta mukautusta ennen kuin ensimmäinen, hyvin pelkistetty versio saatiin Windows Phonelle buildattua.

3.1 Tehtävänanto

Tehtävistä tärkeiksi mainittiin mainosverkot ja niiden integrointi peliin. Suuri osa, jopa joissain ilmaispeleissä ainoa tulonlähde on mainokset (Boshell 2016). Tiltagon 2.0 päivitykseen oli myös lisätty muitakin monetisaatiota parantavia ominaisuuksia. Tehtävänä oli lisätä peliin IAP-ominaisuudet erilaisten pallojen mahdollisiin ostoihin. Monetisaatioon liittyvien lisäysten lisäksi projektiin täytyi lisätä toimivuus sosiaalisen median kanssa kommunikoinnin mahdollistaviin painikkeisiin. Näin pelin tuloksen pystyisi jakamaan Facebookissa sekä Twitterissä. Lopuksi peliä pitäisi testata, jotta se olisi valmiina julkaisuun.

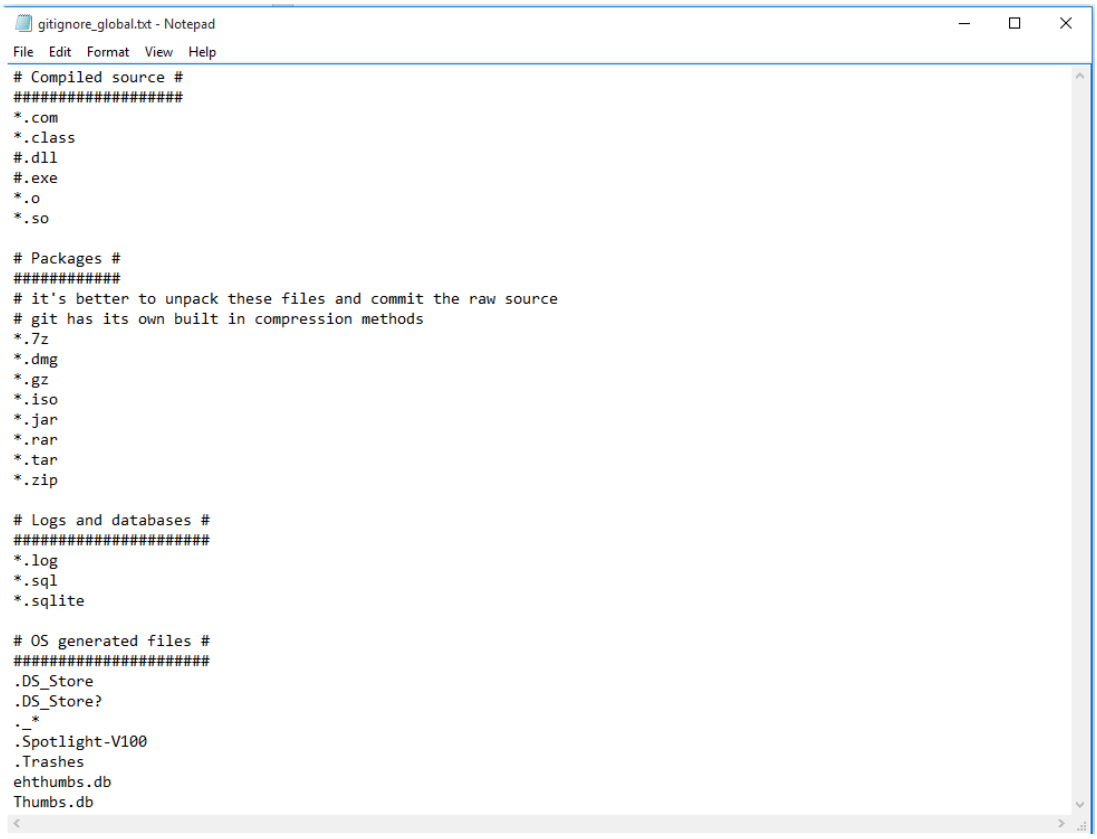
3.2 SourceTreen käyttöönotto

SourceTreen käytössä ilmeni alussa ongelmia, koska Bitbucket (SourceTreen käytön mahdollistava palvelu) sivuutti paljon tärkeitä tiedostoja. Kaikista tärkeimmät tiedostot, joita se sivuutti olivat .dll-loppuiset liitännäistiedostot. Jotta kaikki tarpeelliset tiedostot saataisiin läpi, oli muokattava Global Ignore Listiä. Tässä listassa määritetään, mitkä tiedostot Bitbucket päästää läpi (kuva 15).



Kuva 15. Tiedoston gitignore_global.txt, löytäminen SourceTreessä

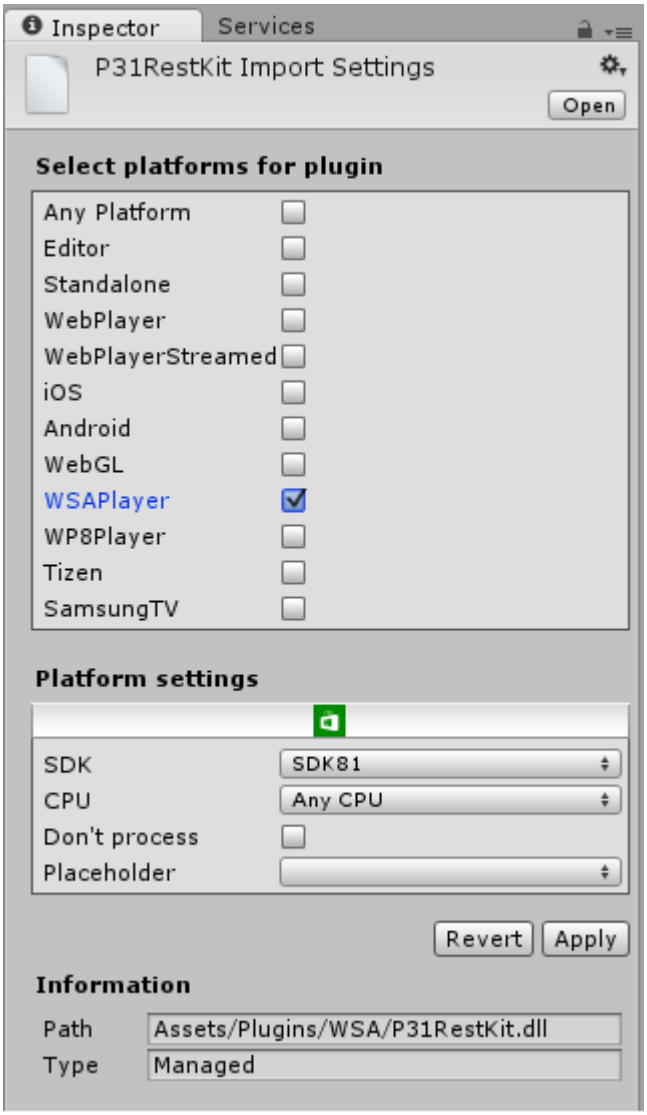
Global Ignore List suodattaa tiedostoja tiedostopäätteiden mukaan. Jos listaan on merkitty merkki * tiedostopäätteen eteen, sen päätteen omaavia tiedostoja ei päästetä läpi tietoturvasyihin vedoten. Kun tiedostopäätteen edessä on #-merkki, on esto kommentoitu pois, ja nämä tiedostot voivat näin kulkea vapaasti (kuva 16)



Kuva 16. Esimerkkitiedosto gitignore_global.txt

3.3 Koodin puhdistaminen Unityssä

Koska aluksi saatu pohjaprojekti oli tarkoitettu muille alustoille, täytyi siihen tehdä monia muutoksia ja karsimisia. Liitännäisistä poistettiin iCloud- ja Steamworks-liitännäiset, jotka haittasivat pelin toimintaa alustakohtaisuuden vuoksi. Projektin muut liitännäiset asennettiin uudestaan, koska niiden toiminta oli puutteellista. Lisäksi jokaisella liitännäisellä piti olla oikea vaihtoehto eli WSAPlayer valittuna, jotta pelimoottori osaa hakea oikeat asetukset kyseiselle alustalle (kuva 17).



Kuva 17. Alustan valitseminen liitännäiselle

Myös pelin koodeissa tehtiin isoja muutoksia ennen kuin projektin raaka versio saatiin käynnistettyä. Koodeista oli poistettava kaikki poistettuihin liitännäisiin viittaavat kohdat ja samalla lisätä koodia Windows Phonelle soveltuvat korvaavat koodit. Unity3D:llä on tuki multiplatformiin, mikä mahdollistaa

saman koodiin käyttämisen jokaiselle eri alustalle. Tämä toimii käyttämällä alustakohtaisissa koodin osissa #if-komentoa (kuva 18).

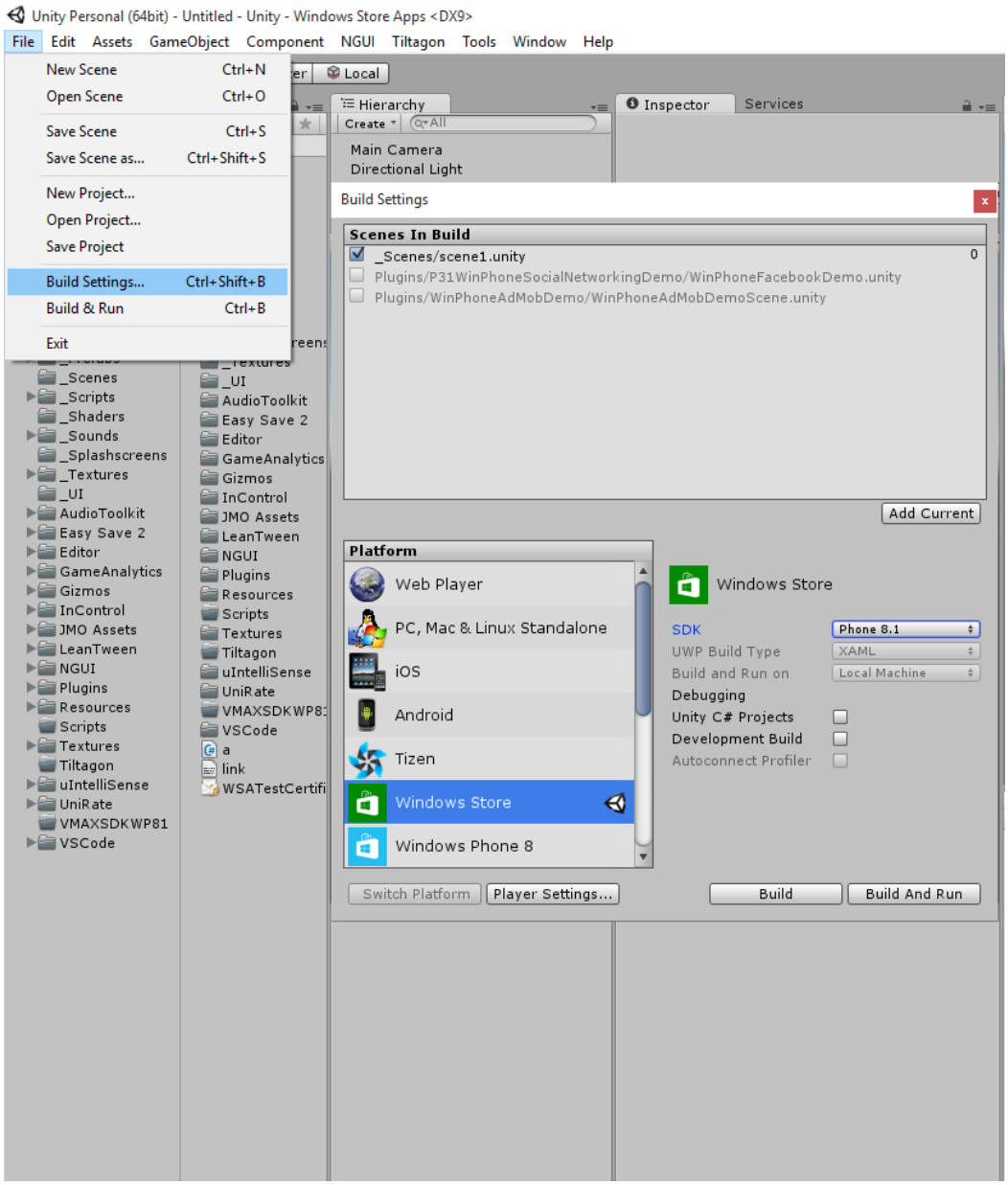
```
#if UNITY_IPHONE || UNITY_ANDROID
using Prime31;

#elif UNITY_WP8 || UNITY_WP8_1 || UNITY_WSA
using Prime31;
using Prime31.WinPhoneSocialNetworking;
#else
    using Prime31;
#endif
```

Kuva 18. Esimerkki koodin jaottelusta eri alustoille

3.4 Sovelluksen ensimmäinen buildaus

Windows Phone -alustalle buildatessa on kaksi vaihetta. Ensin projekti buildataan Unity3D:stä Microsoft Visual Studiolle, jonka jälkeen se buildataan vielä erikseen Visual Studiosta puhelimeen. Unityn buildauksessa projektissa ei saa olla mitään kriittistä virhettä, joka estää peliä pyörimästä. Build käynnistetään Build Settingsistä, josta on myös vaihdettava lopullinen alusta jolle peli luodaan (kuva 19).

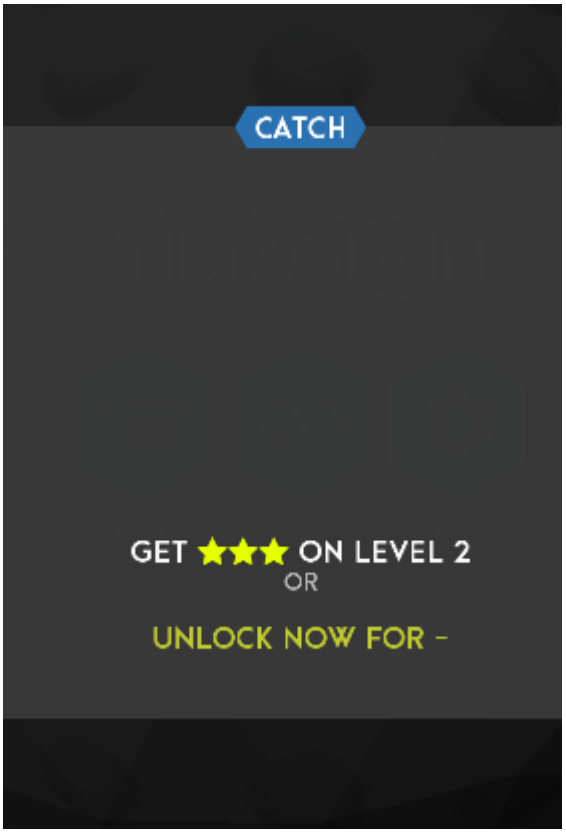


Kuva 19. Buildin käynnistäminen

Visual Studiossa projekti saatiin suoraan siirrettyä puhelimeen. Tämä ensimmäinen puhelimeen saatu versio ei tehnyt oikeastaan mitään. Siinä aukesi valikot ja asetukset, mutta itse peli ei toiminut. Lisäksi versiosta puuttui ostettavien pallojen ulkoasu ja niistä oli pelkät nimet nähtävillä.

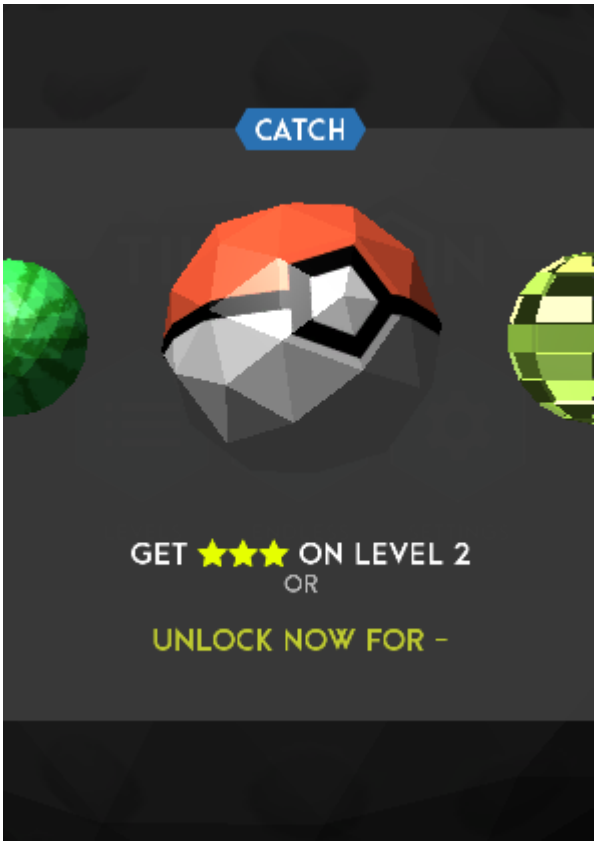
4 PROJEKTIN VAIHEET

Ensimmäistä buildia pidettiin projektin pohjana ja siitä alettiin korjaamaan puutteita sekä lisäämään eri toimintoja. Ensin päätettiin korjata buildissa olevat selkeät puutteet. Ostoruudun pallojen ulkoasujen näkyväksi saamiseen ei ollut mitään yksiselitteistä tapaa (kuva 20).



Kuva 20. Pallon ostoruutu puuttellisella pallojen ulkoasulla

Lopulta huomattiin, että verkkolevyllä Unity näytti toimivan huonommin, joten siirrettiin projekti tietokoneen fyysiselle C-levylle. Tämän jälkeen opinnäytetyön tilaaja kertoi, että hänelläkin oli ollut vastaavia ongelmia, ja että siihen auttoi 3d-mallinnusohjelma Blenderin lataaminen koneelle. Blenderin asennuksen jälkeen ulkoasut alkoivat toimimaan (kuva 21).



Kuva 21. Pallon ostoruutu toimivalla pallojen ulkoasulla

Peliä käynnistäessä näytölle tuli kontrollien valintavalikko. Tämä piti saada pois, koska ainoa Windows Phonea tukeva kontrollimalli on kallistus. UIManager.cs-tiedostosta oli etsittävä ja poistettava ominaisuus (ohjelmalistaus 1).

```
#elif UNITY_WP8 || UNITY_WP8_1 || UNITY_WSA
        CloseMenuSelectControls();
#endif
```

Ohjelmalistaus 1. Suljetaan kontrollien kysyminen tehden näin oletukseksi pelkän kallistuksen

Tässä vaiheessa pelin puutteet oli korjattu, joten voitiin keskittyä ominaisuuksien lisäämisiin. Sovittiin tilaajan kanssa, että keskitytään tekemään peli vain Windows Phone -alustalle eikä Windows Universal Store -alustalle, koska sen ominaisuudet eivät olleetkaan samankaltaisia pohjana käytetyn projektin kanssa. Windows Universal Store -alustalle tekeminen olisi mahdollistanut saman projektin käyttämisen sekä älylaitteilla että tietokoneella (taulukko 1).

Taulukko 1. Unityn eri alustavaihtoehdot Windowsin eri alustoille (Unity Technologies 2016b)

UNITY_WP_8	Suunnattu ainoastaan Windows Phone 8:lle
UNITY_WP_8_1	Suunnattu ainoastaan Windows Phone 8.1:lle
UNITY_WSA	Suunnattu Windows Store -sovelluksille
UNITY_WSA_8_0	Vain Windows Store -sovelluksille, jotka on suunnattu SDK 8.0:aa käyttäville alustoille
UNITY_WSA_8_1	Vain Windows Store -sovelluksille, jotka on suunnattu SDK 8.1:tä käyttäville alustoille
UNITY_WSA_10_0	Vain Windows Store -sovelluksille, jotka on suunnattu Universal-käyttöön
UNITY_WINRT	Yhdistelmä, joka mahdollistaa yhteensopivuuden UNITY_WP_8:n ja UNITY_WSA:n välillä
UNITY_WINRT_8_0	Yhdistelmä, joka mahdollistaa yhteensopivuuden UNITY_WP_8:n ja UNITY_WSA_8_0:n välillä
UNITY_WINRT_8_1	Yhdistelmä, joka mahdollistaa yhteensopivuuden UNITY_WP_8_1:n ja UNITY_WSA_8_1:n välillä
UNITY_WINRT_10_0	Mahdollistaa Universal-käytön

4.1 Pelin sisäiset ostot

Tiltagonissa on mahdollista saada kaikki pallot, jos on tarpeeksi hyvä. Jos kuitenkin näkee mieluisensa pallon, jolla haluaa pelata, voi sen myös ostaa. Pallojen lisäksi pelistä voi ostaa täyden, mainoksettoman version.

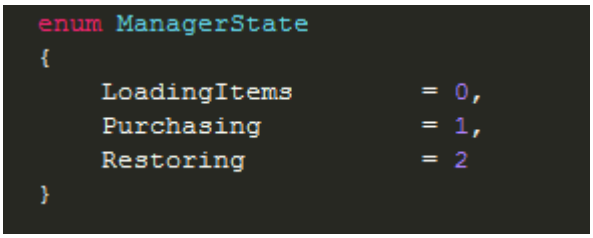
Pelin sisäiset ostot tehtiin Prime31:n liitännäisellä. Windows Phone 8:lle ja Windows Phone 8.1:lle tehtäessä ovat kirjastot sekä komennot erinimisiä. Tämän vuoksi väärälle alustalle tehtäessä koodia on vaikea korjata, koska virheilmoituksia ei tule, koska Prime31 tunnistaa komennot, mutta ominaisuudet ei myöskään toimi. Windows Phone 8:ssa käytetyn kirjaston nimi on Prime31.WinPhoneStore, kun Windows Phone 8.1:ssä se on Prime31.WinRTStore. (Prime31 2016b.)

Ostoja testattiin laittamalla Prime31:n liitännäisen mukana tulleesta demoprojektista kohta, mikä luettelee tuotteet lokimerkintöjen kautta näkyville (ohjelmalistaus 2)

```
Store.loadTestingLicenseXmlFile("license.xml", list-
ingInfo =>
{
Debug.Log("loading up listing. printing results");
Debug.Log("price: " + listingInfo.formattedPrice);
Debug.Log("description: " + listingInfo.description);
Debug.Log("dumping productListings: ");
foreach (var p in listingInfo.productListings)
Debug.Log(p.Key + ": " + p.Value);
            });
```

Ohjelmalistaus 2. Tuotelistan tulostaminen lokimerkintöihin

Pelin sisäisiä ostoja hallinnoivassa InAppPurchaseManager.cs-tiedostossa on määritetty managerin tiloja ostamishetkellä. Tilat menevät päälle tapahtuman muuttuessa ja ne täytyi aina tapahtuman päättyessä sulkea, koska peli voi lakata toimimasta, jos tiloja on samaan aikaan päällä (kuva 22).



Kuva 22. InAppPurchaseManager.cs-tiedoston managerin tilat

Ostettavat tuotteet ja niiden lisätiedot piti saada Tiltagonin omasta products-listauksesta Prime31:n InAppPurchase-listaukseen. Tämä tehtiin Prime31:n kirjastossa olevalla funktiolla OnProductListRequestSuccess(ListingInformation _productListing), mikä hakee tuotelistauksen Microsoftin servereiltä. Tämän funktion sisällä on tuotteet lueteltu foreach-lauseella, joka hakee tuotelistan jokaisen tuotteen. Tämän lauseen sisällä tapahtuu listojen yhdistäminen (ohjelmalistaus 3).

```
products[i] = new InAppPurchase
  (
id,
product.Value.productId,
product.Value.name,
product.Value.description,
1.00f,
product.Value.formattedPrice
);
```

Ohjelmalistaus 3. Foreach-lauseen sisällä oleva tuotteiden tietojen lisääminen

Tuotteen osto tapahtuu funktiossa BuyItem(string _id), jossa id:n avulla haetaan tuotteen SKU (stock keeping unit), mikä tarkoittaa tuotteen yksikköä, ja mikä sisältää tuotteen tietoja kuten hinnan, nimen ja kuvauksen (Business Dictionary 2016). Tuotteen oston jälkeen koodi tuottaa onnistuessaan kuitin tai epäonnistuessaan virheilmoituksen (ohjelmointilistaus 4).

```
if( !managerState[(int)ManagerState.Purchasing] )
{
//Start purchasing
managerState[(int)ManagerState.Purchasing] = true;
Store.requestProductPurchase( sku, ( receipt, error )
=>
{
//Debug.Log("requestProductPurchase done");
// we will either have a receipt or an error
if( receipt != null )
{
       //purchase completed with receipt
       OnPurchaseSuccess(sku);
}
else if( error != null )
{
        //error purchasing product
        OnPurchaseFailed(sku, error);
}
});
```

Ohjelmalistaus 4. Foreach-lauseen sisällä oleva tuotteiden tietojen lisääminen

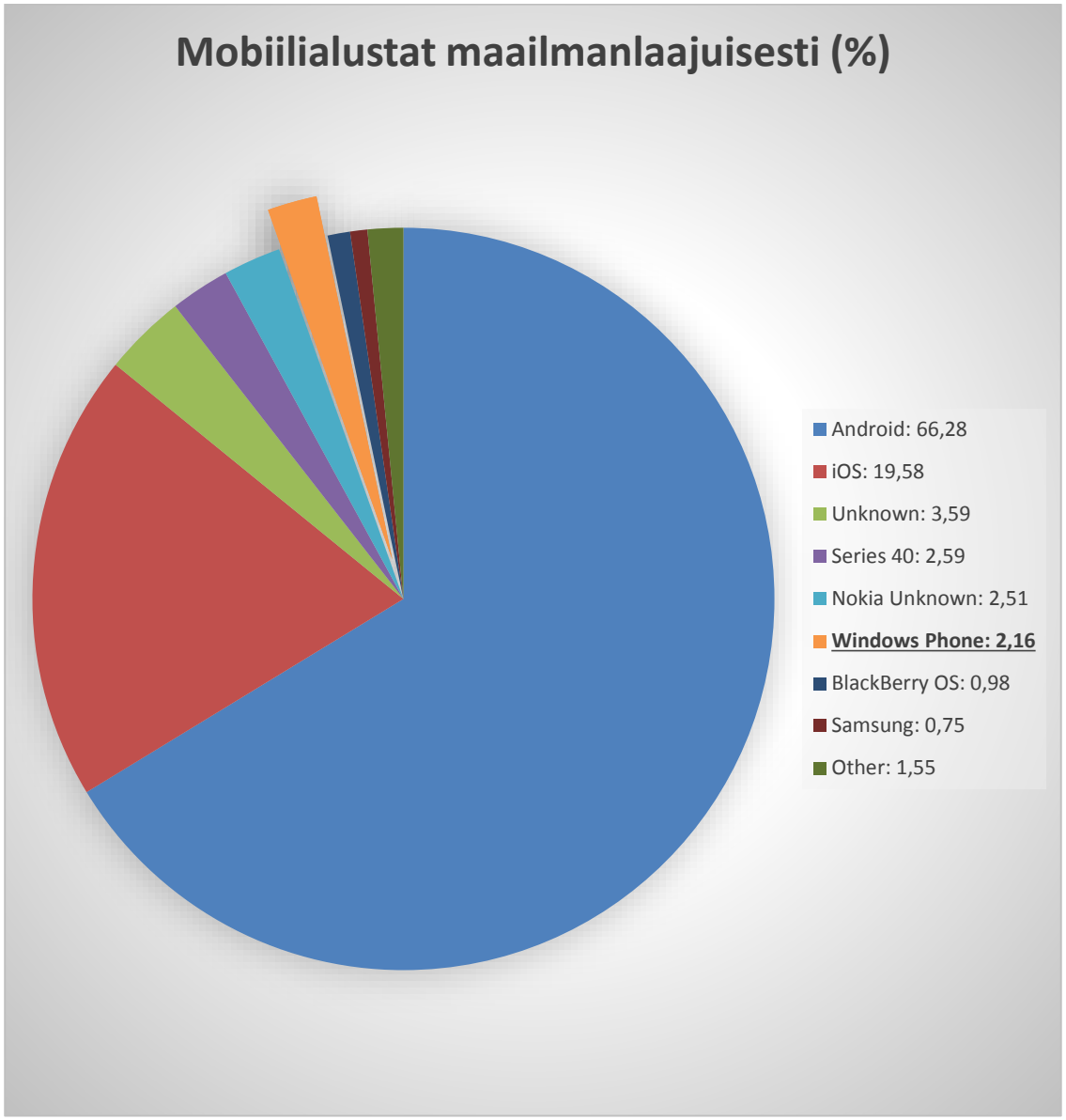
Pelin sisäiset ostot toimivat, mutta tuotetta ostaessa pystyi peruuttamaan tilauksen, mutta silti sai tuotteen omaan peliinsä. Tämä piti korjata koodissa varmistamalla, että oston tila on onnistunut (ohjelmalistaus 5)

```
if(receipt.status == ProductPurchaseStatus.Succeeded)
{
       OnPurchaseSuccess(sku);
}
else
{
       OnPurchaseFailed(sku, error);
}
```

Ohjelmalistaus 5. Oston onnistumisen tarkistaminen ennen sen hyväksymistä

4.2 Mainosten integrointi

Mainosten integraatiosta muodostui koko projektin haastavin työ. Tämä johtui siitä, että Windows Phone -alustalle ei löydy montaa Unity-liitännäistä. Myös monesta liitännäisestä, mille on joskus löytynyt Windows Phone -tuki, on vetänyt sen pois markkinoilta, koska liitännäisen päivittäminen on vähäisen tuoton takia kannattamatonta (kuva 23). Vaikka Windows Phone -alustalle on ennakoitu nousevia markkinoita, ei tämän hetken myyntiluvuilla ole tuottoisaa pitää liitännäisiä markkinoilla (Globalnerdy 2016).



Kuva 23. Kaavio Windows Phonen markkinaosuudesta tammikuulta 2016 (StatCounter 2016)

4.2.1 Videomainokset

Videomainoksiin liitännäisenä käytettiin Vunglea. Videomainosta käytettiin pelin loputtomassa pelimuodossa. Ideana oli, että jos pelissä saa vähintään viisi pistettä, saa peliä jatkaa kerran, vaikka olisikin pudonnut, katsottuaan mainoksen.

Vunglella on hyvin kattava tutoriaali, minkä mukaan oli helppo edetä. Kaikista Vunglen liitännäisen mukana tulleista tiedostoista oli valittu WSAPlayer alustaksi. Koska mainokset haetaan internetin välityksellä, piti package.appmanifest-tiedostosta sallia pelin internet-yhteys.

Koodissa Vungle piti alustaa komennolla Vungle.init(”AppID”), jonka jälkeen pystyy jo soittamaan mainoksen laittamalla haluamaansa kohtaa komennon Vungle.playAD(). Koska mainoksen katsomalla pelaaja saa hyvityksen, laitettiin koodiin boolean-arvo, mikä palauttaa arvon todeksi, jos mainos on katsottu (ohjelmalistaus 6).

```
bool watchedAd = false;

Vungle.onAdFinishedEvent += (adFinished) =>
{
        watchedAd = adFinished.IsCompletedView;
};

void Update()
{
        if(watchedAd)
        {
                OnRewardedVideoEvent(true);
        }
}
```

Ohjelmalistaus 6. Katsotun mainoksen boolean-arvon käyttäminen pelaajaa palkitessa

Tehtiin oma testitunnus Vungleen, jota käytettiin videomainosten testaamiseen. Kun testitunnus vaihdettiin oikeaan tunnukseen, mainokset eivät enää toimineet. Selvisi, että testatessa Vunglen asetuksissa pitää sallia testikäyttö.

4.2.2 Koko ruudun mainokset

Koko ruudun mainokset ovat tasaisin väliajoin tulevia mainoksia, joita tulee joka tapauksessa, jos ei ole ostanut pelin mainoksetonta versiota. Mainokset integroidaan joko suoraan Unityn kautta, jolloin Visual Studioon buildaamiseen ei tarvitse lisätä mitään, tai buildauksen jälkeen Visual Studiossa, jolloin valmista projektia ei saa suoraan lähetettyä opinnäytetyön tilaajalle. Tilaajan kanssa sovittiin, että olisi parempi, jos löytyisi Unityn kanssa yhteensopiva liitännäinen, jotta saisi lopuksi lähetettyä kokonaisen Unity-projektin sen sijaan, että joutuisi lähettämään Visual Studio –projektin, koska siinä tapauksessa projektia ei pystyisi enää muuttamaan samalla tavalla.

4.2.2.1 Mainosten integrointi Windows Phone 8.1 -versioon

Mainosten integrointiin koitettiin monia eri liitännäisiä sekä mainosverkkoja. Tästä huolimatta sekä Googlen omistaman mainosverkon liitännäinen AdMob, että eri mainosverkkoja hyväksikäyttävä liitännäinen VMAX, ovat Windows 8.1 -alustalla Visual Studiolla hallinnoitavia liitännäisiä. Unityllä hallinnoitavia toimivia liitännäisiä ei löydetty ollenkaan Windows Phone 8.1 -alustalle. Tämän vuoksi tilaaja päätti, että siirretään peli Windows Phone 8 -alustalle. Tämän vuoksi projekti piti siirtää vanhemmalle, Unityn 5.2.4f1-versiolle, jonka takia moni asia piti asentaa ja rakentaa uudestaan. Windows Phone 8 ei myöskään tue Vunglen videomainoksia, joten ne jäivät pois lopullisesta versiosta. Positiivinen asia muutoksessa oli kuitenkin opinnäytetyön tilaajan mukaan alemman tuloluokan maiden markkinat, joilla on Windows Phone laitteista huomattava määrä Windows Phone 8-laitteita. Windows Phone 8 -alustalle tehdyt pelit toimivat kaikissa uudemmissa käyttöjärjestelmissä, mutta Windows Phone 8.1 -alustan pelit eivät toimi enää Windows Phone 8:ssa. (VMAX 2016.)

4.2.2.2 Mainosten integrointi Windows Phone 8 -versioon

Integroidessa Windows Phone 8 -alustalle, VMAX sisälsi mahdollisuuden hallinnoida mainoksia Unityn kautta. Unityn AdManager.cs-tiedoston aloitus metodiin piti alustaa liitännäinen (ohjelmalistaus 7).

```
using VMAXPlugin;
void Start() {
pluginVMAXAd = new VMAXWP8Plugin();
pluginVMAXAd.SetAdspotId("ID");
pluginVMAXAd.SetAdUXType(VMAXWP8Plugin.AdUxType.Inter-
stitial);
}
```

Ohjelmalistaus 7. Mainosliitännäisen alustaminen

Jo pelin käynnistyessä peli hakee ensimmäistä kertaa mainosta. Tämä ominaisuus lisättiin peliin käyttäjätyytyväisyyden ylläpitämiseksi, ettei käyttäjä joudu odottamaan aina mainoksen lataamista. Mainos ladataan valmiiksi komennolla pluginVMAXAd.CacheAd();. Jos mainos on jo valmiina, voidaan se näyttää pelaajalle komennolla pluginVMAXAd.ShowAd();. Jos mainosta ei halua ladata etukäteen, voi sen vaan ladata ja näyttää suoraan komennolla pluginVMAXAd.LoadAd();.

Mainosten testaamista varten vaihdettiin koodissa olleet numeeriset arvot: private long interstitialAdInterval sekä private long interstitialFirstBootWaitTime pieniksi, jotta heti ensimmäisen pelikerran jälkeen näki toimiko mainokset. interstitialAdInterval käsitteli mainosten välistä aikaa ensimmäisen mainoksen ilmestymisen jälkeen. interstitialFirstBootWaitTime puolestaan käsitteli aikaa, jolloin ensimmäinen mainos ilmestyy.

Visual Studion MainPage.xaml.cs-koodiin piti tehdä myös muutama muutos, ennen kun mainokset alkavat toimimaan (ohjelmalistaus 8). MainPage.xaml.cs-tiedosto on yhteydessä käyttöliittymään, joka on määritetty MainPage.xml-tiedostossa (Arvai, Balãssy, Fulop & Novák 2012, 98).

```
// Alussa kohdistava using-lause
using VMAXPlugin;

// Konstruktoriin:
VMAXPlugin.VMAXWP8Plugin.DSBG =
DrawingSurfaceBackground;

// PhoneApplicationPage_BackKeyPress-funktioon, jotta
paluu-nappi toimisi oikein
private void PhoneApplicationPage_BackKeyPress(object
sender, CancelEventArgs e)
{
        if (!VMAXPlugin.VMAXWP8Plugin.isBrowserOpened)
        {
            e.Cancel = UnityApp.BackButtonPressed();
        }
}
```

Ohjelmalistaus 8. MainPage.xaml.cs-tiedostoon tehtävät lisäykset

4.2.2.3 Interop

Interop on Visual Studion ja Unityn kommunikointiin käytettävä C#-kooditiedosto. Interopin kanssa toimittiin niiden mainosliitännäisten toiminnan helpottamiseksi, joiden koodinmuokkaukset oli tehtävä Visual Studion projektin kautta. Käytännössä Interop-tiedostossa on tehty tapahtuma, joka on yhdistetty projektissa olevaan funktioon (Microsoft Developer Network 2016b). Koodin ehdon täyttyessä käynnistyy Visual Studiossa tapahtuma, joka lähettää tiedon WinRTBridgen eli sillan kautta Unityyn (Unity Technologies 2016c). Tämä laukaisee siihen liitetyn funktion (ohjelmalistaus 9).

```
public static Action OnShowInterstitial = delegate { };
public static Action OnPreloadInterstitial = delegate {
};

static Interop()
{
OnShowInterstitial = delegate { };
OnPreloadInterstitial = delegate { };
}

public static void PreloadInterstitialAd()
{
OnPreloadInterstitial();
}
public static void ShowInterstitialAd()
{
OnShowInterstitial();
}
```

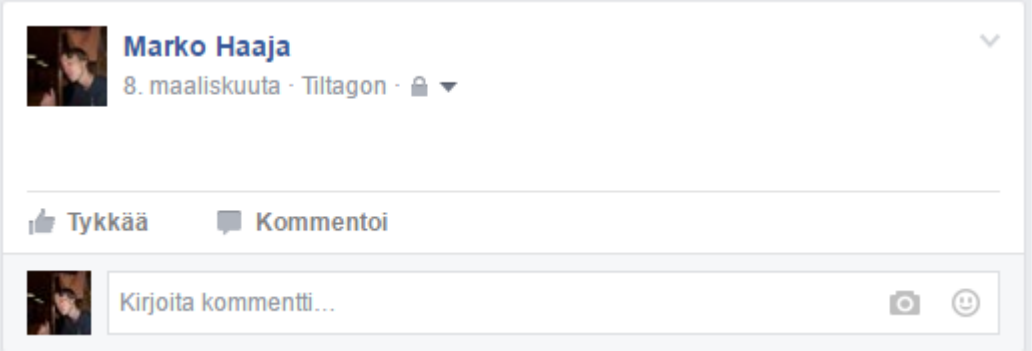
Ohjelmalistaus 9. Interop-tiedostossa luodut tapahtumat, funktiot sekä konstruktori

4.3 Sosiaalisen median liittäminen peliin

Sosiaalisen median tuomaa voimaa voidaan käyttää hyväksi pelin markkinoinnissa. Tiltagonissa mahdollisuutena on tehdä omasta huipputuloksestaan päivitys Facebookiin tai Twitteriin. Päivityksessä on lisäksi pelin latauslinkki, joten jokainen jako lisää potentiaalisia pelaajia, koska jokaisesta pelaajan kaverista voi tulla uusi pelaaja. Tämä on siis toisin sanoen ilmaista mainontaa.

Projektissa sosiaalisen median liittämiseen käytettiin Prime31:n WinPhoneSocialNetworking-liitännäistä. Projektin pohjaversiossa kaikki siihen liittyvä oli poistettu tai kommentoitu pois, koska Steam-versio ei käytä niitä, joten sosiaalisen median pohjakoodi haettiin vanhasta, ensimmäisestä Windowsin versiosta.

Vanhassa versiossa oli kuitenkin eroavaisuuksia nykyiseen verrattuna, osien komentojen ollessa erinimisiä. Facebookiin jakaminen ei toiminut alussa kunnolla. Päivitystä laittaessa kaikki meni samalla tavalla, kuin jo testatussa vanhassa versiossa, mutta päivitys oli hyvin pelkistetty versio. Siinä luki vain pienellä Tiltagon, eikä mitään muuta siihen viittaavaa (kuva 24).



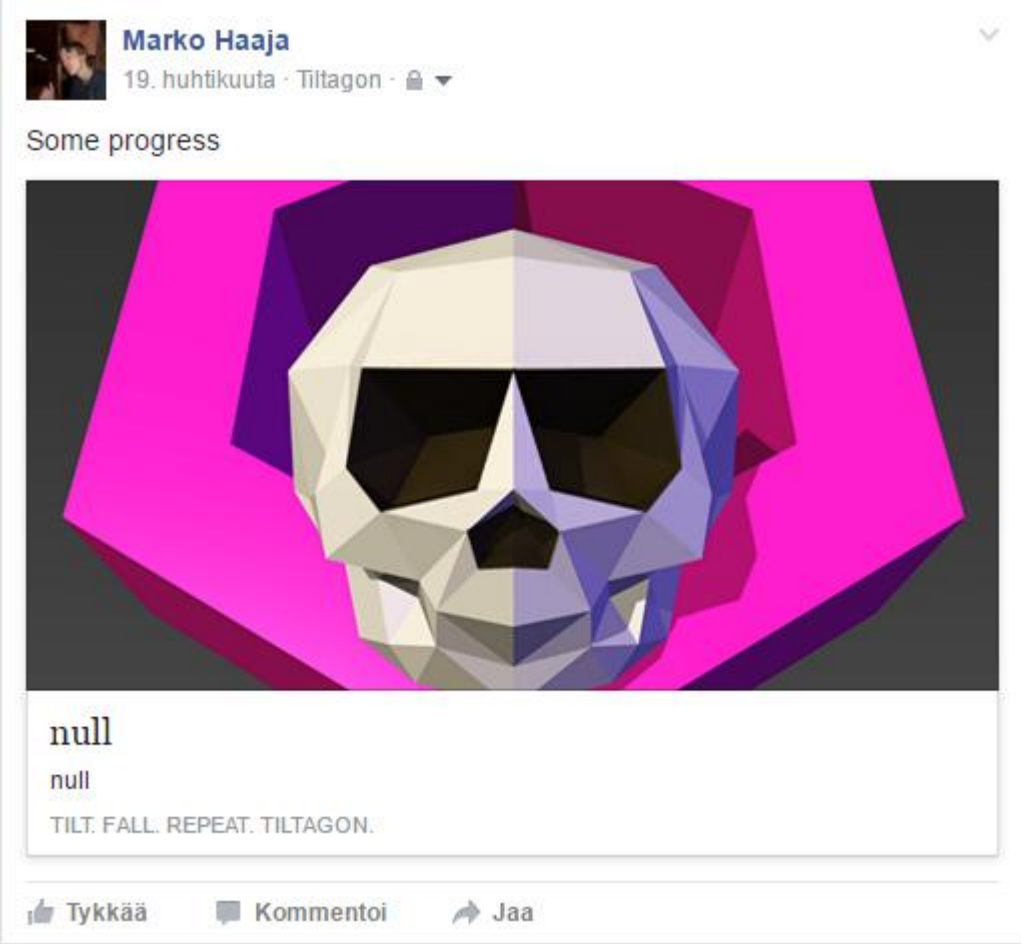
Kuva 24. Tiltagonin Facebook-päivityksen ensimmäinen versio

Tähän vaiheeseen sosiaalinen media jätettiin loppuviimeistelyyn asti. Koodia tarkastelemalla huomattiin virhe kohdassa, joka määrittää päivityksen sisällön. Vanhan version parametrit olivat eriäviä uuden version kanssa (ohjelmalistaus 10).

```
var parameters = new Dictionary<string, object>
       //VANHA
{
{ "title", _title },
{ "description", _message },
{ "caption",  caption},
{ "imageURL", "http://www.tiltagon.com/share/facebook-
score.png" },
{ "contentURL", link }
};
       //UUSI
{
{ "name", _title },
{ "description", _message },
{ "caption",  caption},
{ "picture", "http://www.tiltagon.com/share/facebook-
score.png" },
{ "link", link }
};
```

Ohjelmalistaus 10. Vanhat parametrit SocialMediaManager.cs-tiedostossa

Kaikki vanhat parametrit muutettiin samannimisiksi kuin uudetkin. Tällä tavalla päivitykseen saatiin jo pohja valmiiksi, mutta muista koodeista haettavat arvot vielä puuttuivat (kuva 25).



Kuva 25. Toimiva Facebook-päivityksen pohja ilman haettuja arvoja

Jäljitetiin arvot sisältävä tiedosto. Se oli nimeltään UIManager.cs. Sieltä löytyi virheen sisältävä kohta samantien. Alustakohtaisten #if-ehtolauseiden Windows-kohdista puuttui määritelmä UNITY_WP8 eli Windows Phone 8:n määritys. Ne lisättyä Facebookiin jakaminen alkoi toimimaan moitteettomasti (kuva 26).

Kuva 26. Valmis Tiltagonin Facebook-päivitys

Käytännössä Facebook- ja Twitter-päivitysten toiminnallisuuden luonti toimi samalla tavalla Prime31:n liitännäistä käyttäen. Alussa piti laittaa managerstate-muuttujia enum-funktiolla (ohjelmalistaus 11). Enum-funktio toimii hyvin managerin tilan määrittämiseen, koska siinä voi olla vain yksi tila kerrallaan päällä, niin kuin pelikin tekee vain yhtä asiaa kerrallaan. (Dot-netperls 2016.)

```
enum ManagerState
{
TwitterLoggingIn = 0,
TwitterFailedLoginClosing = 1,
TwitterPosting = 2,
FacebookLoggingIn = 3,
FacebookFailedLoginClosing = 4,
FacebookPosting = 5,
FacebookReauthorizing = 6,
FacebookSuccessfulPostClosing = 7,
FacebookFailedPostClosing = 8,
TwitterSuccessfulPostClosing = 9,
TwitterFailedPostClosing = 10
}
```

Ohjelmalistaus 11. SocialMediaManager.cs-tiedoston funktio enum ManagerState

Start-funktiossa sekä Facebook, että Twitter täytyi alustaa. Facebookin sovellustunniste määritettiin lauseella FacebookAccess.ApplicationId = ”tunniste”. Twitterissä tehtiin muuttuja, joka alusti Twitterin ja tarkisti myöhemmässä funktiossa, onko käyttäjä kirjautunut aiemmin sisään. Tämä muuttuja luotiin lauseella: var didLogin = TwitterAccess.init(”tunniste”).

Päivityksen tekeminen Facebookiin luotiin funktiossa PostToFacebook(title, message), jonka sisällä muutettiin managerin tilaa, jonka jälkeen funktio kävi läpi virheitä tarkastavia ehtolauseita. Niiden jälkeen, jos ei mitään virheitä ollut ilmentynyt, käynnistyy toinen funktio nimeltä Share(title, text), joka hakee arvot ja jakaa tiedot Facebookiin.

Twitterissä tehtiin funktio PostToTwitter(text) yhdistää ensin Twitteriin, jonka jälkeen se tarkistaa onko Twitteriin kirjauduttu. Ensimmäisellä kerralla Twitteriin kirjautuessa peli ei koskaan lopettanut latausta, vaan jäi jumiin kirjautumiskohtaan, mutta vanhaa versiota testatessa lopulta huomattiin, että tuo sama virhe oli myös siinäkin. Selvisi, että virhe oli liitännäisessä, joten emme voineet korjata sitä. Kun oli jo kirjautunut sisään, niin toisella yrityksellä päivityksen lähetys onnistuu (kuva 27). Lopuksi päivitys tehdään funktion Tweet(text) kautta. (Prime31 2016b.)

Kuva 27. Onnistunut Twitter-päivitys

Windows Phone -alustalle tehtäessä funktioiden kirjoittaminen oli huomattavasti työläämpää, kuin projektissa jo valmiina olleet Anroid- ja iPhone-funktiot. Siinä missä muilla alustoilla oli funktioita tiivistettynä yhteen lauseeseen, oli Windows Phonelle kirjoitettava monta riviä (ohjelmalistaus 8).

```
#if UNITY_IOS
FacebookBinding.showFacebookShareDialog(parameters);

#elif UNITY_ANDROID
FacebookAndroid.showFacebookShareDialog(parameters);

#elif UNITY_WP8 || UNITY_WP8_1 || UNITY_WSA
FacebookAccess.showDialog("stream.publish", parameters,
dialogResultUrl =>
{
if (dialogResultUrl == null || dialogResultUrl == "?er-
ror=user canceled" || dialogResultUrl == "#_=_")
{
if (!managerState[(int)ManagerState.FacebookFailedPost-
Closing])
{
statusUpdated = managerState[(int)ManagerState.Face-
bookFailedPostClosing] = true;
}
}
else
{
if (!managerState[(int)ManagerState.FacebookSuccessful-
PostClosing])
{
statusUpdated = managerState[(int)ManagerState.Face-
bookSuccessfulPostClosing] = true;
}
}
});
#endif
```

Ohjelmalistaus 12. Esimerkki Windows Phonen ja muiden alustojen funktioiden eroista

5 VIRHEILMOITUKSET JA NIIDEN RATKAISEMINEN

Pelin pohjana pidettyä projektia oli käytetty viimeksi peliin tehdyn Steam-version pohjana. Tämän vuoksi projektissa oli useita tiedostoja, jotka eivät olleet yhteensopivia Windows Phone -version kanssa. Näitä tiedostoja täytyi muokata tai poistaa (taulukko 2).

Taulukko 2. Steamworks-kirjastoon liittyvä virhe sekä ratkaisu

Virhe	Ratkaisu
Assets\Plugins\Steamworks.NET\InteropHelp.cs(57,45): error CS0234: The type or namespace name 'Win32' does not exist in the namespace 'Microsoft' (are you missing an assembly reference?)	Poistettiin projektista koko Steamworks-kirjasto, koska projekti ei käytä sitä

Unityn liitännäiset aiheuttivat aluksi paljon virheilmoituksia projektissa. Ratkaisuna suurimpaan osaan virheistä toimi liitännäisten uudelleen asennus (taulukko 3). Kun uudelleenasennus ei auttanut, täytyi jokaisesta liitännäisen tiedostosta katsoa, että niissä oli paikanpitäjä (placeholder) joka on pakollinen asettaa liitännäisten tiedostoihin sen toiminnan varmistamiseksi (Unity Technologies 2016d.)

Taulukko 3. Liitännäisten virheilmoitukset ja ratkaisut

Virhe	Ratkaisu
FsmStateAction: Assets/Playmaker/Actions/ES2PlayMaker.cs(11,42): error CS0246: The type or namespace name `FsmStateAction' could not be found. Are you missing a using directive or an assembly reference?	Asennettiin EasySave2 uudestaan ja asetettiin paikanpitäjät
Reference rewriter: Error: field `System.Type[] System.Type::EmptyTypes` doesn't exist in target framework. It is referenced from P31RestKit.dll at System.Object Prime31.Reflection. CacheResolver::getNewInstance(System.Type). UnityEngine.Debug:LogError(Object) PostProcessWinRT:RunReferenceRewriter()	Poistettiin Prime31 sekä kaikki sen mukana tulleet tiedostot ja asennettiin se uudestaan

Siirtäessä projektia Visual Studiolle tuli vastaan ongelma C++ Runtime Debuggerin kanssa. Projekti ei löytänyt koko SDK:ta ja peli ei suostunut käynnistymään. Ensin ratkaisuna toimi pelkkä tiedoston poistaminen, koska pelissä ei käytetä C++-ohjelmointikieltä. Lopulta kuitenkin löytyi toinenkin ratkaisu (taulukko 4).

Taulukko 4. C++ Runtime Debugger -virhe ja lopullinen ratkaisu

Virhe	Ratkaisu
Could not find SDK "Microsoft.VCLibs, Version=12.0 Exception thrown: 'System.IO.FileNotFoundException' in Tiltagon.exe	Asennettiin vanhempi Visual Studio, jonka kautta projektiin sai asennettua vanhempi SDK

Alkuperäisen buildaamisen esti kaksi asiaa. WinRTBridge ei toiminut kunnolla joten Unity ja Visual Studio ei pystyneet kommunikoimaan. Tämän lisäksi appmanifestissa määritetyt arvot eivät olleet samoja. Jos appmanifestissa on määritetty eriäviä arvoja samoissa kohdissa, ei ohjelma pyöri oikein tai välttämättä edes käynnisty (taulukko 5).

Taulukko 5. Visual Studion buildin estävät virheet

Virhe	Ratkaisu
WinRT information: System.TypeInitializationException: The type initializer for 'WinRTBridge.WinRTBridge' threw an exception.	Mentiin Unity projektiin, jossa laitettiin rasti ruutuun virheen sisältävän tiedoston asetuksien kohdasta Don’t process
System.TypeInitializationException: The type initializer for 'WinRTBridge.TypeInformation' threw an exception.	
System.IO.FileLoadException: Could not load file or assembly 'ES2, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null' or one of its dependencies. The located assembly's manifest definition does not match the assembly reference. (Exception from HRESULT: 0x80131040)	Laitettiin kaikki appmanifestin arvot samoiksi, kun vanhassa projektissa oli

Mainosten toimimaan saaminessa tuli virheitä vastaan usein koska Windows Phone -alustalle on hyvin harvassa sitä suosivia liitännäisiä ja niitäkin päivitetään hyvin harvoin. Mainosverkot eivät myös aina sisällä mainoksia, joten oli vaikea tietää toimiko tietyt verkot, kun virheilmoituksia ei aina tullut (taulukko 6).

Taulukko 6. Mainosverkkojen virheilmoitukset ja ratkaisut

Virhe	Ratkaisu
VMAXSDKWP81\packages\Mediation Partners\Microsoft\MicrosoftAdMediator\AdMediator.config' does not exist.	Liitännäisen .dll-tiedostot oli piilotettu. Ne täytyi laittaa näkyviin
Exception thrown: 'System.Exception' in AdDuplex.Universal.Controls.WinPhone.XAML.winmd	Ratkaisua näihin ei löytynyt, jonka takia täytyi vaihtaa toiseen mainosverkontarjoajaan
Exception thrown: 'System.Exception' in mscorlib.ni.dll	
Exception thrown: 'System.TypeLoadException' in MicrosoftAdMediator.DLL	

Pelin sisäisiin ostoihin käytettiin aluksi Prime31:n liitännäistä. Kun projekti täytyi vaihtaa Windows Phone 8.1:stä 8:aan, huomattiin, että Prime31:n lisenssi Windows Phone 8:aan on vanhentunut, joten koitimme eri liitännäisiä. Lopulta palattiin kuitenkin Prime31:een, jonka aikaisemmat virheet saatiin korjattua (taulukko 7).

Taulukko 7. Prime31-liitännäisen ongelmat ja ratkaisut

Virhe	Ratkaisu
Unhandled Exception: System.Reflection.ReflectionTypeLoadException: The classes in the module cannot be loaded.	Vaihdettiin buildin kohdetta Windows Phone 8:aan ja poistettiin kaikki Prime31-tiedostot uudestaan. Uudelleenasentaessa laitettiin kaikki asetukset kohdistamaan Windows Phone 8:aa. Myös koodia piti uudelleenkirjoittaa, koska komennot olivat erilaisia 8.1:n ja 8:n välillä, sekä asettaa kaikille tiedostoille tiedostopolku kohtaan placeholder.
The following assembly referenced from C:\tiltagon\Unity\Assets\Plugins\WP8\P31RestKit.dll could not be loaded: Assembly: System.Runtime (assemblyref_index=0)	
Missing method .ctor in assembly C:\tiltagon\Unity\Assets\Plugins\WP8\P31RestKit.dll, type System.Reflection.AssemblyTrademarkAttribute	
Missing method .ctor in assembly C:\tiltagon\Unity\Assets\Plugins\WSA\WindowsPhone81\P31WinPhoneSocialNetworking.dll, type System.Runtime.InteropServices.ComVisibleAttribute	

6 VIIMEISTELY

Projektin alussa annettujen tehtävien valmistuttua, alkoi pelin viimeistelyvaihe. Aluksi karsittiin koodia, jotta opinnäytetyön tilaajan olisi helpompi lukea sitä. Koodista otettiin pois turhia kommentointeja, sekä kaikki jo poistettujen liitännäisten ohjelmointikoodi ja funktiot, jotka kaikki pidettiin koodissa varmuuden vuoksi vielä siihen asti. Koodin osia poistettaessa oli jatkuvasti buildattava projekti uudestaan, jottei mitään ominaisuuksia ollut hävinnyt samalla.

6.1 Testaaminen

Testaaminen on tärkeä osa pelin kehittämisessä. Peliä oli testattu koko projektin ajan tasaisin väliajoin, mutta nyt koko projektin loppuaika keskitettiin ainoastaan testaamiselle ja bugien eli sovellusten virheiden löytämiselle ja paikallistamiselle.

Testauksessa kokeiltiin kaikenlaisia eri kombinaatioita, joita ei välttämättä normaalilla pelikerralla itse kokeile, mutta joku muu voi. Ensimmäinen virhe löytyi, kun yritti mennä inventaarioon eli myytävien tavaroiden tilaan, missä pystyi myös valitsemaan mieleisensä pallon. Jos koko ruudun mainos oli tullut ruudulle ennen inventaarion valitsemista, ei sitä pystynyt valitsemaan. Tämä johtui edellisen version jäänteestä. Pohjaprojektissa oli laitettu funktio, joka mainoksen tullessa estää inventoryyn pääsyn ja ei enää palauta sitä.

Twitterin yhdistämiseen liittyvä ongelma oli jo tiedossa ennen varsinaista testaamista, mutta ongelman laatua yritettiin minimoida piilottamalla näytölle jäävä teksti tietyn ajan jälkeen, jolloin se olisi jo yhdistänyt Twitteriin sekä ehtinyt tehdä päivityksen. Laitettiin kuuden sekunnin aika, jonka jälkeen teksti ruudulta katoaa ja pelaamista pystyy jatkamaan käynnistämättä sovellusta uudestaan. Tämä toimi komennolla Manager.instance.uiManager.HideMessage(6f).

Mainoksia integroidessa arvo, joka määritti mainosten ilmestymisen tiheyden, oli laitettu maksimiin eli mainoksia tuli joka pelikerran jälkeen. Tämä nopeutti mainosten testaamista kehitysvaiheessa. Testausvaiheessa kuitenkin laitettiin arvo normaaliksi eli mainosten välinen aika asetettiin kolmeen minuuttiin. Tästä kuitenkin löytyi virhe ja mainoksia tuli edelleen liian tiheään tahtiin.

Testausvaiheen loppuosiossa mainosten saatavuus heikentyi huomattavasti. Integrointivaiheessa tuli aina näytölle oikea mainos, mutta lopuksi siihen tuli vain valkoisia testikäyttöön tarkoitettuja mainoksia.

6.2 Julkaisu

Tiltagonin päivityksen julkaisupäivä ei ollut projektin loppumisen jälkeen tiedossa. Opinnäytetyön tilaajalla oli muita projekteja kesken ja vasta niiden jälkeen on mahdollisuus keskittyä päivityksen julkaisuun. Tiltagonin päivitys 2.0 oli jo ennen projektia julkaistu muilla mobiilialustoilla.

7 JOHTOPÄÄTÖKSET

Tiltagon-projektin parissa työskenteleminen alkoi kun tarvitsin työharjoittelupaikan, ja koululta minut opastettiin ottamaan yhteyttä Kiemuraan. Ennen opinnäytetyötä olin tehnyt Unityn kanssa kouluprojekteja sekä pelejä game jameissa, eli tapahtumissa, missä tehdään pelejä tietyn aikarajan sisällä. Tämä takasi minulle jonkinlaisen pohjan pelimoottorin toimintaan.

Vaikka pohja Unityn käyttöön suhteellisen vakaa olikin, sain nopeasti huomata, että projektissa mentiin epämukavuusalueelle, missä piti kysellä, että mitä minkäkin kooditiedoston on tarkoitus tehdä projektissa kaiken ollessa aluksi hyvin sekavaa. Vaikka projekteja olikin tullut tehtyä, täysin uusina asioina tuli varsinkin Unityn liitännäisten käyttö sekä Windows Phone -alustan kanssa toimiminen. Liitännäisten asettaminen on aivan oma maailmansa. Osassa liitännäisistä on todella hyvät demoprojektit sekä askel askeleelta opastavat ohjeet, toisissa taas on epäkunnossa olevat demoprojektit ja ohjeet ovat puuttelisia tai toimivat vain tietyissä tilanteissa. Varsinkin laadukkaat liitännäiset juuri Windows Phonelle ovat katoava luonnonvara markkinoiden vähäisyyden vuoksi. Tuntui, että Windows Phonessa muutenkin kaikki piti moninkertaisella vaivalla Androidiin verrattuna.

Työ oli haastava ja sitä vaikeutti netistä löytyvien ohjeiden vähyys. Harva kehittää enää mitään pelejä Windows Phonelle, varsinkaan Windows Phone 8:lle. Ongelmien tullessa oli vaihtoehtona lähettää viestejä eri liitännäisten asiakaspalveluihin, tai jos sekään ei auttanut, piti koittaa ratkaista ongelmat itse. Tämä tuotti pitkiä jaksoja ilman mitään etenemistä, mutta asioihin itse perehtyminen puolestaan kehitti tuntuvasti pelien kehittämisen yleistaitoa.

Uskon, että tämä opinnäytetyö kehitti osaamistani enemmän kuin yksikään kurssi tai projekti koko koulun aikana.

Projekti antoi minulle lisää uskoa siitä, että pelin ohjelmointi voisi onnistua tulevaisuudessakin. Tekemällä kehittyy ja jos ei pidä taukoja ja antaa vain mennä, voi kehittyminenkin vielä jatkua. Sain myös paljon lisätietoja eri mobiilialustojen vahvuuksista sekä heikkouksista. Jos joskus tulevaisuudessa olen kehittämässä mobiilipelejä, on pohjatyöt tehty mille alustalle kannattaa tehdä, sekä minkä kehittäjän liitännäisiä kannattaa ostaa ja mitkä kannattaa kiertää kaukaa. Näin sanoisin, että opinnäytetyölläni voi olla itselleni pitkäkestoinenkin vaikutus.

LÄHTEET

Arvai, Z., Balãssy, G., Fulop, D. & Novák, I. 2012. Beginning Windows 8 Application Development. Indianapolis, IN: Wiley.

Boshell, B. 2016. Mobile Monetization: A Study of Popular Games. Gamasutra. Saatavissa: http://www.gamasutra.com/blogs/BrendonBoshell/20160713/276973/Mobile_Monetization_A_Study_of_Popular_Games.php [viitattu 25.7.2016]

Business Dictionary. 2016. SKU definition. Saatavissa: http://www.businessdictionary.com/definition/stock-keeping-unit-SKU.html [viitattu 6.6.2016]

deVilla, J. 2015. It's 2015, the year when Windows Phone is supposed to take over. Global Nerdy. Saatavissa: http://www.globalnerdy.com/2015/01/01/its-2015-the-year-when-windows-phone-is-supposed-to-take-over/ [viitattu 6.6.2016]

Dotnetperls. 2016. Enum. Saatavissa: http://www.dotnetperls.com/enum [viitattu 25.7.2016]

Järvinen, J. Windows Phone: sovelluskehitys. 2012. Jyväskylä : Docendo.

Kiemura. 2016. Teoksia uusilla piirteillä. Saatavissa: www.kiemura.com [viitattu 27.4.2016]

Microsoft Developer Network. 2016a. Adding References Using NuGet Versus an Extension SDK. Saatavilla: https://msdn.microsoft.com/en-us/library/jj161096.aspx [viitattu 1.6.2016]

Microsoft Developer Network. 2016b. Managing Exceptions with the Debugger. Saatavilla: https://msdn.microsoft.com/en-us/library/x85tt0dd.aspx [viitattu 1.6.2016]

Microsoft Developer Network. 2016c. Action <T> Delegate. Saatavilla: https://msdn.microsoft.com/en-us/library/018hxwa8(v=vs.110).aspx [viitattu 8.6.2016]

Prime31. 2016a. Windows Universal Store Plugin Documentation. Saatavissa: https://prime31.com/docs#winMetroStore [viitattu 27.4.2016]

Prime31. 2016b. Social Networking Plugin Support. Saatavissa: http://support.prime31.com/22293/error-when-building-winuniversalstore-in-unity-2f1-for-wp8 [viitattu 25.7.2016]

SourceTree. 2016. Free Mercurial and Git Client for Windows and Mac. Saatavissa: https://www.sourcetreeapp.com/ [viitattu 17.5.2016]

StatCounter. 2016. Mobile Operating Systems. Saatavilla: http://gs.statcounter.com/#mobile_os-wwmonthly-201601-201601-bar [viitattu 29.1.2016].

Techopedia. 2016. Advanced RISC Machine (ARM). Saatavilla: https://www.techopedia.com/definition/5900/advanced-risc-machine-arm [viitattu 25.7.2016]

Tiltagon. 2016. Tiltagon. Saatavilla: http://www.tiltagon.com/ [viitattu 27.4.2016]

Unity Asset Store. 2016a. Asset Store. Saatavilla: https://www.assetstore.unity3d.com/en/ [viitattu 18.5.2016]

Unity Asset Store. 2016b. LeanTween. Saatavilla: https://www.assetstore.unity3d.com/en/#!/content/3595 [viitattu 18.5.2016]

Unity Asset Store. 2016c. NGUI. Saatavilla: https://www.assetstore.unity3d.com/en/#!/content/2413 [viitattu 18.5.2016]

Unity Asset Store. 2016d. UniRate. Saatavilla: https://www.assetstore.unity3d.com/en/#!/content/10116 [viitattu 18.5.2016]

Unity Asset Store. 2016e. Easy Save – The Complete Save & Load Asset for Unity. Saatavilla: https://www.assetstore.unity3d.com/en/#!/content/768 [viitattu 18.5.2016]

Unity Technologies. 2016a. Windows Store project types. Saatavissa: https://docs.unity3d.com/Manual/windowsstore-projecttypes.html [viitattu 1.6.2016]

Unity Technologies. 2016b. Platform Depended compilation. Saatavissa: https://docs.unity3d.com/Manual/PlatformDependentCompilation.html [viitattu 25.7.2016]

Uniity Technologies. 2016c.Windows Store – AppCallBacks. Saatavilla: http://docs.unity3d.com/Manual/windowsstore-appcallbacks.html [viitattu 8.6.2016]

Unity Technologies. 2016d. InputField.placeholder. Saatavissa: https://docs.unity3d.com/ScriptReference/UI.InputField-placeholder.html [viitattu 16.6.2016]

Visual Studio. 2016. Visual Studio – Microsoft Developer Tools. Saatavilla: https://www.visualstudio.com/ [viitattu 1.6.2016]

Vmax. 2016. Mobile App Monetization. Saatavilla: http://www.vmax.com/ [viitattu 17.5.2016]

Vungle. 2016. Get Started with Vungle - Unity: https://support.vungle.com/hc/en-us/articles/204311244-Get-Started-with-Vungle-Unity [viitattu 1.6.2016]