SOVELLUKSEN KEHITTÄMINEN JA JULKAISEMINEN WINDOWS PHONELLE

Maze Challenge

Janne Uitto

Opinnäytetyö
Joulukuu 2013
Tietotekniikka
Ohjelmistotekniikka

TAMPEREEN AMMATTIKORKEAKOULU
Tampere University of Applied Sciences

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietotekniikka
Ohjelmistotekniikka

JANNE UITTO:
Sovelluksen kehittäminen ja julkaiseminen Windows Phonelle
Maze Challenge

Opinnäytetyö 29 sivua
Joulukuu 2013

Tässä dokumentissa kerrotaan kehittämästäni Maze Challenge –pelistä Windows Phonelle.

Dokumentin ensimmäisessä osassa käydään läpi sovellusten kehittämistä Windows Phonelle. Lisäksi kerrotaan Windows Phonen historiasta. Microsoftin mobiilikäyttöjärjestelmän uusimmasta versiosta esitellään uusia ominaisuuksia sekä tekniikoita. Myös sovellusten kehittämiseen tarkoitettuja työkaluja käydään läpi.

Toisessa osassa kerrotaan Maze Challengen kehittämisestä sekä siinä käytettävistä tekniikoista. Kappaleessa kerrotaan myös Microsoftin tarjoamista palveluista kuten beta-testaus tai in-app purchase.

Kolmannessa osiossa keskitytään sovelluksen julkaisuun Microsoftin tarjoamilla työkaluilla ja palveluilla. Myös sovelluksen julkaisuun liittyviä Microsoftin vaatimuksia kerrotaan tässä kappaleessa.

Lopuksi pohditaan sovelluksen menestystä julkaisun jälkeen sekä sovelluksen mahdollisia jatkokehityksiä.

Asiasanat: windows phone sovellus mobiili ohjelmointi

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree of Computer Science
Software Development

JANNE UITTO:
Developing and Publishing Application to Windows Phone
Maze Challenge

Bachelor's thesis 29 pages
December 2013

This document contains development of game called Maze Challenge for Windows Phone.

First part of the document goes through development for Windows Phone. Also history of Windows Phone is told in this part. Newest features and techniques of Microsoft's latest mobile operating system version are presented. Tools to develop applications are presented in this part.

The second part of this document contains development of Maze Challenge and techniques used during development. This part contains services like beta-test and in-app purchase provided by Microsoft.

The third part focuses to publishing application using Microsoft's tools and services. Also Microsoft's application requirements are presented in this chapter.

Finally document contains final thoughts of success of application after publishing and future development of the application.

SISÄLLYS

LYHENTEET JA TERMIT

GDR	General Distribution Release
IAP	In-app purchase, sovelluksen sisäinen osto
NFC	Near field connection, lähitunnistus
OS	Operating system, käyttöjärjestelmä
SDK	Software development kit, ohjelmointiympäristö
WP	Windows Phone
XAML	Extensible Application Markup Language

1 JOHDANTO

Tässä dokumentissa käsitellään opinnäytetyötäni, jonka tein Tampereen ammattikorkeakoulussa tietotekniikan koulutusohjelmassa ohjelmistotekniikka suuntautumisvaihtoehtona.

Mobiilisovellukset ovat tätä kirjoittaessa suuressa suosiossa, ja markkinoille tulee päivittäin useita uusia sovelluksia. Mobiilikäyttöjärjestelmistä Microsoftin Windows Phone on kovassa nousussa, jonka takia päätin tehdä sovelluksen käyttäen tätä mobiilikäyttöjärjestelmää. Työn tavoitteena oli saada sovellus julkaistuksi Microsoftin sovelluskaupassa.

Luvussa kaksi käydään läpi Windows Phonen historiaa sekä uusia ominaisuuksia WP 8 versiossa. Lisäksi luvussa kerrotaan työkaluista, joilla sovelluksia kehitetään ja sovelluksen suorituskykyä voidaan analysoida. Luvussa kolme käydään läpi tämän sovelluksen kehittämisen eri vaiheet aina labyrinttigeneraattorin luonnista sovelluksen viimeistelyyn. Neljännessä luvussa kerrotaan sovelluksen julkaisemisesta Microsoftin sovelluskauppaan sekä vaatimuksia, jotta sovellus tulee hyväksytyksi sovelluskauppaan. Viimeisessä luvussa kerrotaan sovelluksen julkaisun jälkeisistä sovellukseen liittyvistä tapahtumista sekä kerrotaan työssä tehdyn sovelluksen menestyksestä.

2 WINDOWS PHONE

2.1 Windows Phonen historia

Microsoft julkaisi vuonna 2003 Windows Mobile -mobiilikäyttöjärjestelmän, jonka Microsoft pyrki tekemään ulkonäöltään ja ohjelmointirajapinnoiltaan samanlaiseksi kuin yrityksen käyttöjärjestelmät, joita käytetään pöytäkoneissa. Windows Marketplace for Mobile –sivun kautta kolmannen osapuolen ohjelmistokehittäjät pystyivät jakamaan sovelluksia Windows Mobileen. Windows Mobile perustuu Windows CE –käyttöjärjestelmään. (Notebooks, 2013)

Vuonna 2010 Microsoftin julkaisema Windows Phone 7 –käyttöjärjestelmä korvasi Windows Mobilen, joka oli ehtinyt jo versioon 6.5. Uusi WP 7 perustuu myös Windows CE –käyttöjärjestelmään, mutta laitteistovaatimukset ja ohjelmointirajapinnat muuttuivat uuden käyttöjärjestelmän myötä, joten Windows Mobile -laitteita ei voinut päivittää. Windows Phone 8 julkaistiin vuonna 2012, joka perustuu Windows NT –käyttöjärjestelmään. Microsoftin Windows 8 ja Windows Phone 8 –käyttöjärjestelmissä on sama kernel, joten yhteisiä ohjelmointirajapintoja löytyy monia. Vanhat WP 7 -laittet ei ole päivitettävissä uuteen WP 8 -käyttöjärjestelmään uuden kernelin takia, mutta Microsoft on pitänyt WP 7:lle kehitettyjen sovellusten yhteensopivuuden uusissa WP 8 -laitteissa. (Blogging Hits, 2013)

2.2 Windows Phone 8

Windows Phone 7-laitteille tehdyt sovellukset toimivat myös WP 8 -laitteissa, mutta jos sovellus tehdään WP 8:lle käyttäen natiivia C++-koodia tai WP 8:ssa tulleita uusia ohjelmointirajapintoja, ei sovellus ole yhteensopiva vanhojen WP 7 -laitteiden kanssa. WP 8 tukee kolmea eri resoluutiota (WVGA 800x480, WXGA 1280x768, 720p 1280x720), jotka käyttöjärjestelmä hoitaa automaattisesti ilman että kehittäjän tarvitsee asiaa erikoisemmin miettiä. Windows Phone SDK:ssa kehitys tapahtuu WVGA resoluutiota ajatellen ja käyttöjärjestelmä skaalaa suuremmille resoluutioille näytettävät asiat. Mikäli kehittäjä haluaa, suuremmille resoluutioille saa esimerkiksi kuvat näkymään natiivilla resoluutiolla menettämättä tarkkuutta. WP 8 tukee NFC:tä, kolmannen osapuolen kamerasovelluksia sekä kernel tukee jopa 64 ytimisiä

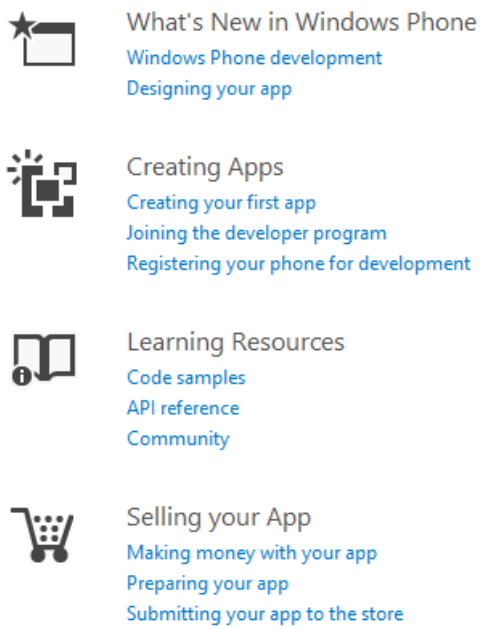
prosessoreita. Microsoft on luvannut päivityksiä tiheämmin WP 8:iin, joka on näkynyt jo kolmena suurempana päivityksenä vuonna 2013. Alkuvuodesta saapunut ”Portico”-päivitys (GDR1) sai jatkoa toukokuussa, jolloin esiteltiin GDR2. Lokakuussa 2013 Windows Phone 8 Update 3:n (GDR3) myötä lisättiin tuki myös 1080p-resoluution näytölle ja uusille prosessoreille. (MobiiliBlogi, 2013)

Windows Phone 8:n tuki natiiville C++-koodille helpottaa saman koodin käyttämisen kaikkien Windows ekosysteemin laitteiden kesken. Lisäksi C++-koodilla on mahdollista luoda tehokkaampaa koodia kuin C#-koodilla. WP 8:lle on mahdollista kehittää Direct3D:llä tehokkaita ja näyttäviä pelejä, jolloin täytyy käyttää C++-koodia.

Tässä työssä päädyin käyttämään C#-koodia, koska grafiikan puolesta sovelluksella ei ole niin suuria vaatimuksia, että tarvitsisin Direct3D:tä. Lisäksi halusin jakaa sovelluksen myös WP 7 käyttäjille.

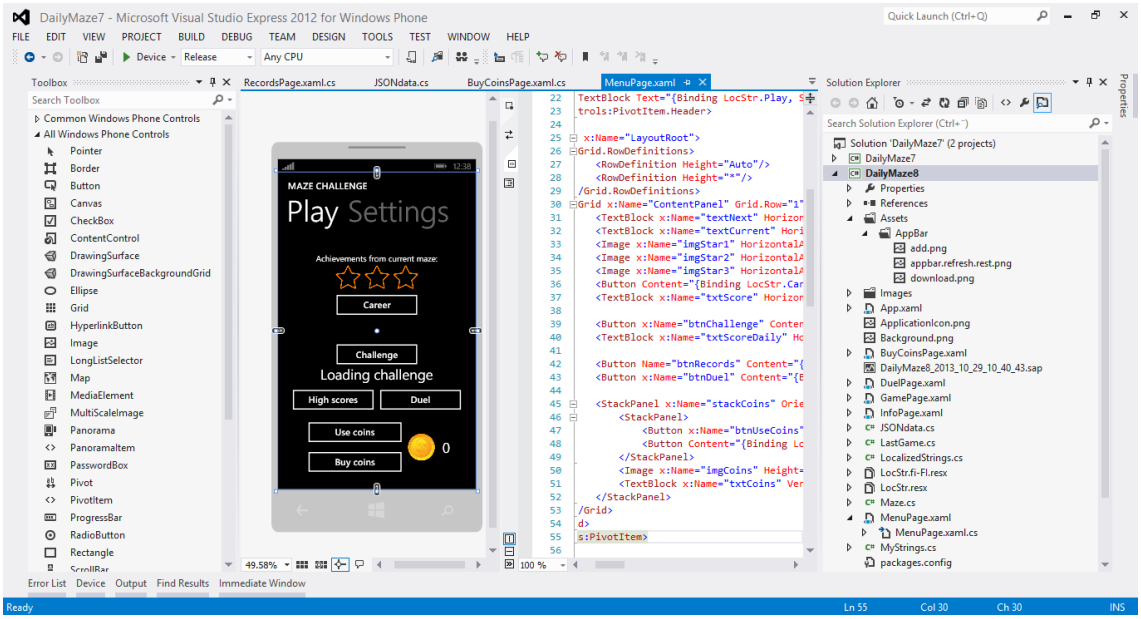
2.3 Windows Phone SDK

Windows Phonelle kehittämistä varten tarvitaan Microsoftin Visual Studio Express for Windows Phone ja WP 8:lle kehittämiseen tarvitaan ohjelmasta vähintään vuoden 2012 versio. WP SDK sisältää Visual Studion Express for Windows Phone:n ja SDK:n voi ladata ilmaiseksi Microsoftin sivuilta. WP 8 kehittämistä varten täytyy SDK asentaa Windows 8 tai uudemmalle käyttöjärjestelmälle. Visual Studion aukaisun jälkeen aukeaa aloitusopas (KUVA 1), jossa on todella paljon hyvää materiaalia aloittelevalle kehittäjälle, mutta myös kokeneet kehittäjät voivat hyötyä oppaista.



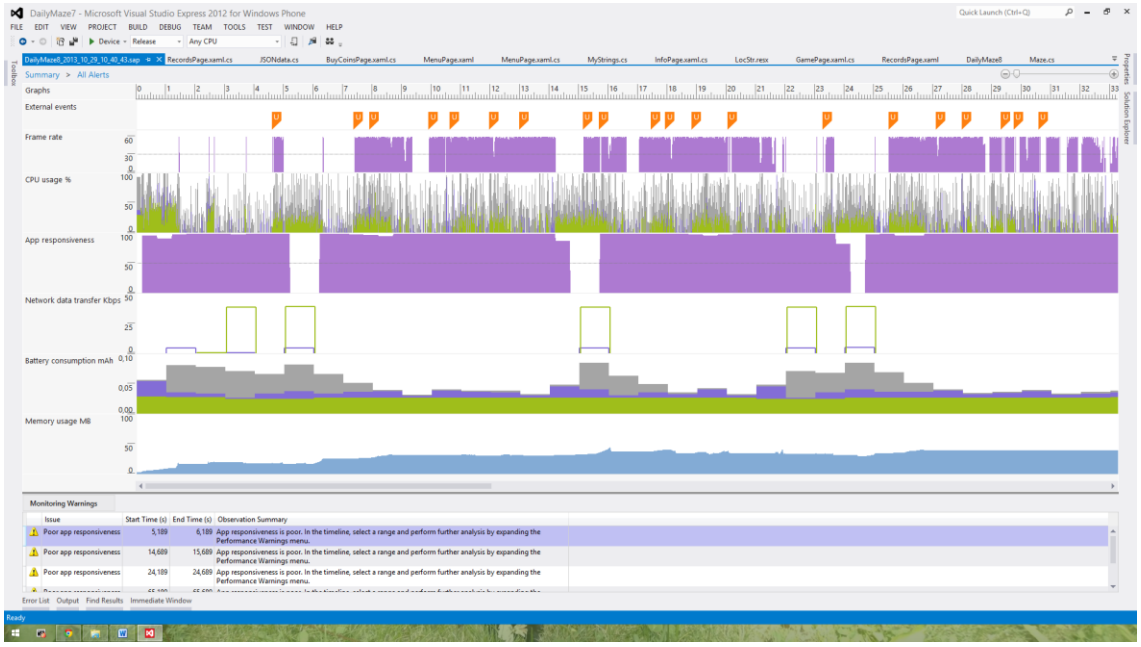
KUVA 1. Visual Studion aloitusopas

Uuden projektin luominen on helppoa. Valittavana on muutamia runkoja sovellukselle kuten tavallinen sivu, panorama-sivu sekä pivot-sivu, jota tässäkin työssä on käytetty. Projektin luomisen jälkeen aukeaa Visual Studio, jossa sovelluksen kehittäminen tapahtuu (KUVA 2). Ulkoasua voi luoda vetämällä vasemmalla olevasta valikosta elementtejä keskellä olevan puhelimen näytön päälle. Tämä tapa on varsinkin aloittelijalle huomattavasti helpompi, mutta elementtejä voi luoda myös XAML-koodilla ikkunassa, joka näkyy kuvassa puhelimen oikealla puolella. Valikosta vetäminen luo XAML-koodia, jota voi myös käsin muokata ja lisätä ominaisuuksia elementeille. Kuvassa oikeanpuoleisin valikko sisältää projektiin kuuluvat tiedostot.



KUVA 2. Visual Studiossa avattu projekti

Visual Studiolla voi mitata ja analysoida sovelluksen suorituskykyä valitsemalla Debug-valikosta ”Start Performance Analysis”, jolloin sovellus käynnistyy laitteessa. Tämän jälkeen kehittäjä käyttää sovellusta niiltä osin kun haluaa mitata suorituskykyä. Sovelluksen käyttämisen jälkeen Visual Studio lataa laitteen kirjoittaman lokin ja analysoi sen. Visual Studio kertoo sovelluksen käynnistämiseen kuluneen ajan, kokonaismäärän verkkoon ladatusta sekä lähetetystä datasta, akun ja muistin kulutuksen keskiarvon. Lisäksi kehittäjä voi tarkastella tarkemmin sovelluksen käyttäytymistä sovelluksen eri vaiheissa. Kuvan 3 yläreunassa on tapahtumat, esimerkiksi sovelluksessa olevien painikkeiden painaminen, joista kehittäjä tietää missä vaiheessa analyysiä mennään. Analyysistä näkee sovelluksen eri vaiheiden ruudunpäivitysnopeuden, prosessorin käytön dataliikenteen käytön, virrankulutuksen sekä muistin käytön.



KUVA 3. Visual Studion Performance Analysis

2.4 Microsoft Blend

Microsoftin WP SDK sisältää myös yhtiön Blend for Visual Studion. Blendillä kehittäjä voi helposti tehdä näyttäviä ulkoasuja sovellukseensa. Blend muokkaa samoja XAML-tiedostoja, joita Visual Studio käyttää. Ulkoasun muokkaamisen jälkeen, kun kehittäjä haluaa palata takaisin Visual Studioon jatkamaan sovelluksen kehittämistä, lataa Visual Studio muuttuneet tiedostot uudelleen ja muutokset sovelluksen ulkoasussa näkyvät myös Visual Studiossa. (Developer Network, 2013b)

Tässä työssä ei käytetty Blendiä ulkoasun tekemisessä, vaan kehittäminen onnistui Visual Studion puolella.

3 KEHITTÄMINEN

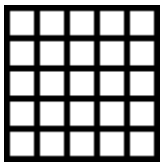
3.1 Suunnittelu

Olen aina pitänyt loogisista peleistä, joten päätin kehittää generaattorin, joka luo labyrinttejä. Selasin Windows Phonen sovelluskaupasta erilaisia labyrinttipelejä, jotta en kehittäisi samanlaista peliä, joka on jo olemassa. Vanhasta labyrintti-lautapelistä, jossa lautaa kallistaen vieritetään palloa välttäen sen putoamista reikiin, löytyy paljon modernisoituja versioita sovelluskaupasta, joten sellaisen tekeminen ei kiinnostanut.

Halusin pelin olevan pelattavissa ilmaiseksi, mutta käyttäjä saisi etuja maksamalla sovelluksesta. Tarkempien etujen muoto ja ilmaisuuden rajoittaminen jäi juuri sovelluksen julkaisun alle, jotta saisin mahdollisimman paljon palautetta testaajiltani sovelluksen koukuttavuudesta ja käytöstä. Projekti sai nimekseen DailyMaze, joka viittaa päivittäin vaihtuvaan labyrinttiin.

3.2 Generaattorin luominen

Erilaisia labyrintti tyyppejä löytyy monia, niistä kaikista kiinnostavin kehittää on ns. täydellinen labyrintti. Täydellinen labyrintti tarkoittaa sitä, että minkä tahansa kahden pisteen välillä on vain yksi ratkaisu. Internetistä löysin sopivan ohjeen generaattorin luomiseen , knock down -algoritmi (Gamedev, 2013). Alussa kaikkien ruutujen väliset seinät ovat ylhäällä kuten kuvassa 4, jossa on 5 x 5 labyrintti.



KUVA 4. 5 x 5 labyrintti, jossa kaikki seinät ylhäällä

Tämän jälkeen valitaan satunnainen seinä kahden ruudun välistä ja yhdistetään ruudut kaatamalla seinä niiden välistä (knock down), jos nämä kaksi ruutua eivät jo ole toista kautta yhteydessä. Tätä toistetaan kunnes kaikki ruudut ovat toisiinsa yhteydessä. Seiniä

poistettaessa täytyy pitää kirjaa siitä, mitkä ruudut ovat toisiinsa yhdistettynä. Algoritmin jälkeen valmis labyrintti voisi näyttä kuvan 5 mukaiselta.

KUVA 5. Täydellinen 5 x 5 labyrintti

Generaattoria luodessa labyrintin täydellisyyden varmistaminen SDK:n debuggerilla on hankalaa, mutta debuggerin avulla paperille piirrettynä 5 x 5 labyrintti näytti onnistuneelta. Suuremman labyrintin oikeellisuuden tarkistaminen oli työlästä, joten täytyi luoda koodia, joka piirtää puhelimen näytölle generoidun labyrintin. Labyrintti piirretään Silverlightin canvakselle. Tämän jälkeen pystyin antamaan generaattorille suurempia labyrinttejä generoitavaksi. Generaattorille annetaan luku, joka alustaa satunnaislukugeneraattorin. Labyrintti-generaattori luo siis identtisen labyrintin, jos sen alustaa samalla luvulla. Tämän avulla myöhemmin kerrotussa Haaste-pelimuodossa kaikille käyttäjille saadaan luotua identtinen labyrintti. Lisäksi generaattorille annetaan labyrintin leveys ja korkeus.

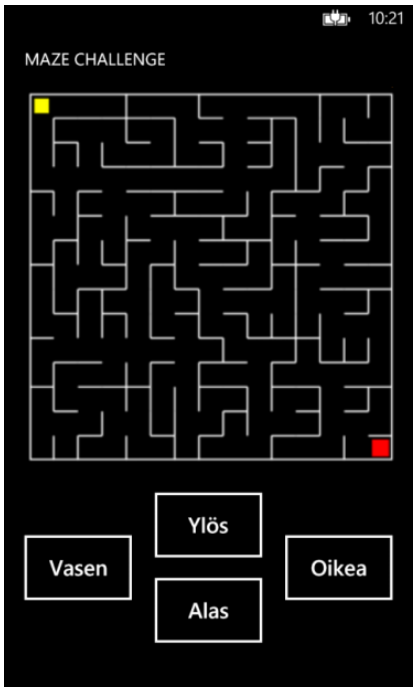
Tässä vaiheessa generaattori tietää, että labyrintti on ratkaistavissa, mutta oikea reitti ja sen pituus ei ole vielä tiedossa. Oikeaa ratkaisua varten käytin leveyshakua, joka aloituspisteestä lähtien erkanee jokaisessa labyrintin risteyksessä kaikkiin risteyksen suuntiin. Erkaantuneita haaroja ajetaan eteenpäin rinnakkain, kunnes jokin haaroista pääsee maaliin. Jos haara päätyy umpikujaan, voidaan kyseinen haara poistaa, koska maaliin tämä haara ei pääse. (Ohjelmointiputka, 2013)

3.3 Perustoiminnot

Generaattorin valmistuttua huomasin, että puhelimen näytölle mahtuu suurempikin kuin aluksi kuvittelemani 10 x 10 labyrintti. Kokeilin 15 x 15 kokoista labyrinttiä, joka omasta mielestäni oli sopivan kokoinen. Seuraavaksi aloin toteuttaa pelihahmon piirtämistä labyrinttiin. Päätin piirtää vain keltaisen neliön, joka olisi mahdollista korvata myöhemmin vaikka kuvalla, jos löytäisin siihen sopivan kuvan. Myös maalin

päätin toteuttaa tässä vaiheessa vain piirtämällä punaisen neliön. Hahmo lähtee vasemmasta yläkulmasta ja maali on oikeassa alakulmassa.

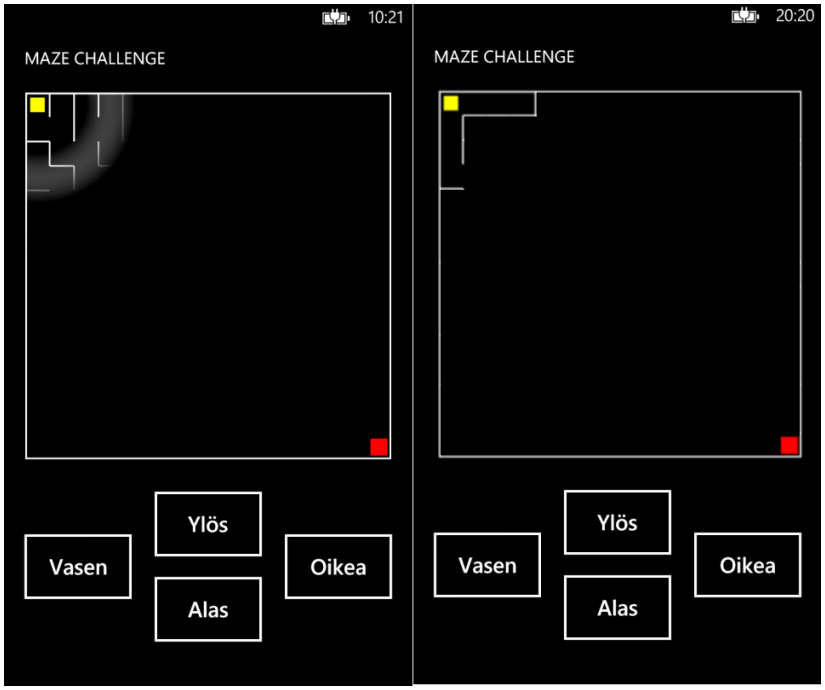
Hahmoa liikutetaan napeilla, jotka on näytön alalaidassa. Käyttäjän liikuttaessa hahmoa, täytyy tarkistaa, että suunnassa johon halutaan liikkua ei ole seinää. Sallitun liikkumisen jälkeen piirretään labyrintti uudelleen. Tämän jälkeen peliä pääsi pelaamaan ja huomasin, että hahmon liikkuessa kuluu häiritsevän paljon aikaa. Arvelin tämän johtuvan siitä, että kaikki näkyvät elementit piirretään canvakselle aina hahmon liikkuessa. Muokkasin piirtämistä niin, että vain muuttuneet elementit piirretään hahmon liikkuessa. Liikkuessa poistetaan hahmo vanhasta paikastaan piirtämällä musta neliö vanhan paikan päälle ja piirretään hahmo uuteen paikkaan. Itse labyrintti piirretään vain alussa, labyrintin generoimisen jälkeen. Tämän muutoksen jälkeen liikkuminen oli paljon nopeampaa, eikä enää häirinnyt pelaamista. Peli oli tässä vaiheessa jo pelattavissa. (KUVA 6)



KUVA 6. Ensimmäinen pelattavissa oleva versio

Päätin näyttää peliä tutuille. Lähes kaikki sanoivat, että peli on liian helppo. Pelin alussa, kun käyttäjä katsoo muutaman sekunnin labyrinttiä, oikean reitin löytää helposti, jonka jälkeen labyrintin pääsee läpi nopeasti. Tämän toki tiesin ilman näitä kommenttejakin, ja alkuperäisessä suunnitelmassa oli muutamia vaihtoehtoja, joilla estää reitin näkeminen heti alussa. Kaverin kanssa pidetyn aivoriihen jälkeen päätin

toteuttaa pyöreän varjon, muiden vaihtoehtojen ollessa ’mitä näkee’ tai ’missä ollut’ (KUVA 7).



KUVA 7. Labyrintin piilottamisen eri vaihtoehtoja

Varjon piirtämisen helpottamiseksi lisäsin hahmoa varten toisen canvaksen edellisen päälle (KUVA 8). Alemmalle canvakselle piirretään labyrintin seinät sekä kehitysvaiheessa käytössä ollut ”Näytä ratkaisu” –toiminto, joka piirsi oikean reitin. Ylemmälle canvakselle piirretään vain asiat jotka muuttuvat liikkuessa, hahmo ja varjo. Ylempi canvas tyhjennetään aina liikkuessa ja piirretään hahmo uuteen paikkaan sekä varjo (KUVA 9), joka vastaa uutta hahmon paikkaa.

```
<Canvas Name="baseCanvas" HorizontalAlignment="Center" Height="430" VerticalAlignment="Top" Width="430"/>
<Canvas Name="canvas" HorizontalAlignment="Center" Height="430" VerticalAlignment="Top" Width="430"/>
```

KUVA 8. Kaksi canvasta, joille piirretään labyrintti

```
// Luodaan ympyrä väri
RadialGradientBrush rgb = new RadialGradientBrush();
rgb.Center = new Point((float)(x + 0.5) / width, (float)(y + 0.5) / height);
rgb.GradientOrigin = new Point((float)(x + 0.5) / width, (float)(y + 0.5) / height);
rgb.RadiusX = (float)shadowSize / width;
rgb.RadiusY = (float)shadowSize / height;

// läpinäkyvää puolet
GradientStop gs = new GradientStop();
gs.Color = Colors.Transparent;
gs.Offset = 0.5;
rgb.GradientStops.Add(gs);

// mustaa loput
GradientStop gs2 = new GradientStop();
gs2.Color = Colors.Black;
gs2.Offset = 1;
rgb.GradientStops.Add(gs2);

// lisätään väri
Rectangle rect = new Rectangle();
rect.Width = canvas.Width;
rect.Height = canvas.Height;
rect.Fill = rgb;

// lisätään varjo canvakselle
canvas.Children.Add(rect);
```

KUVA 9. Varjon piirtäminen

Nyt kun oikeaa reittiä ei näe heti, toteutin pisteidenlaskujärjestelmän. Oikean reitin etsimiseen todennäköisesti menee muutama harhailu väärillä reiteillä, joten lyhimmän reitin etsiminen on tavoittelemisen arvoinen. Pisteiden visuaalistamiseksi tein tähdet. Yhden tähden saa, jos löytää maaliin. Toisen tähden saa, jos löytää maaliin käyttämällä siirtoja mahdollisimman vähän. Samaa labyrinttiä voi yrittää 10 minuutin ajan niin monta kertaa kun ehtii. Tämän jälkeen generaattori alustetaan uudella luvulla, jolloin luodaan uusi labyrintti, josta käyttäjä voi ansaita lisää tähtiä.

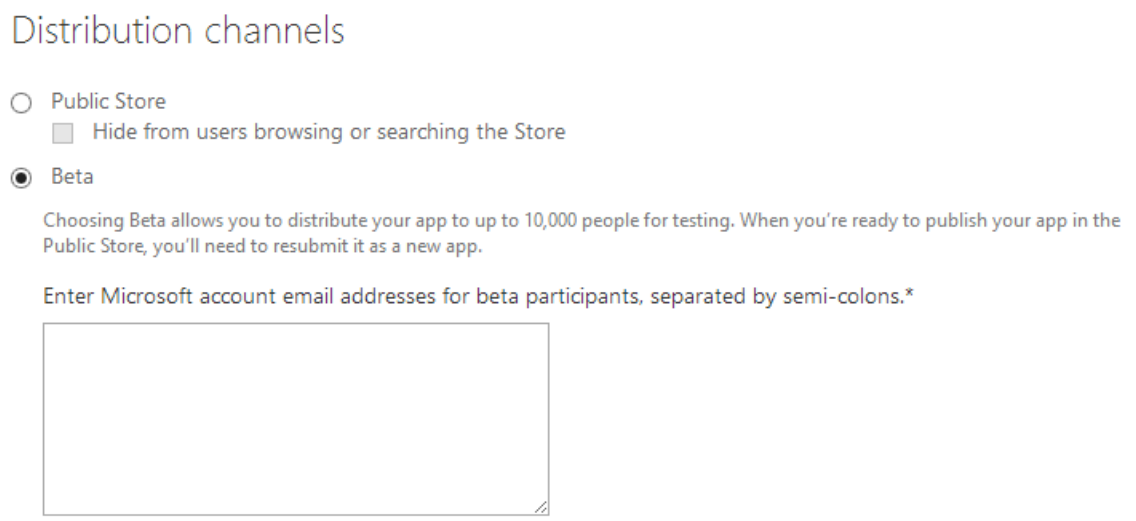
Sovellus on lokalisoitu niin, että kieli vaihtuu puhelimen kielen mukaan. Jos käyttäjän puhelin on suomen kielellä, myös sovellus on suomeksi. Puhelimen kielen ollessa jokin muu, sovellus on englannin kielellä. Kun lokalisoinnin toteuttaa heti sovelluksen teon alusta asti, eri kielien käännöksiä on helppo lisätä jatkossa ja koodiin ei tarvitse koskea ollenkaan. Eri kielien käännökset tehdään erilliseen resurssi-tiedostoon, johon viitataan C#-koodista. (KUVA 10)

```
txtScoreDaily.Text = LocStr.NewChallenge;
```

KUVA 10. Lokalisointi C#-koodissa

3.4 Beta-store

Microsoft tarjoaa ohjelmistokehittäjille sovelluskaupan kautta jaettavaksi beta-versiota sovelluksista (KUVA 11). Betana julkaistu sovellus ei mene Microsoftin normaalin sovelluksen hyväksymisprosessin läpi. Microsoft vain kirjoittaa asennustiedostoon sertifikaatin, jotta puhelin luottaa siihen ja sallii asentaa sovelluksen. Oikean sovelluksen hyväksymisprosessista kerrotaan myöhemmin. Betana julkaistu sovellus ei näy kenenkään käyttäjän tekemässä haussa sovelluskaupassa, kehittäjän tarvitsee jakaa beta-testaajilleen suora linkki, jonka kautta pääsee sovelluskaupasta lataamaan sovelluksen. Vaikka linkin jakaa ulkopuolisille, pääsee sen vain kehittäjän sallimat Microsoft-tilit lataamaan. Kehittäjän tarvitsee lisätä tilit sovelluksen tietoihin Dev Centerin kautta, josta löytyy myös linkki sovellukseen. Testaajat voivat antaa palautetta kehittäjälle sovelluskaupan kautta arvostelemalla sovelluksen, kuten minkä tahansa muunkin sovelluksen.



KUVA 11. Sovelluksen lisääminen betana sovelluskauppaan

Jaoin sovelluksen betana lähipiirilleni sekä koulukavereilleni, joilla on Windows Phone. Yhteensä testaajia oli noin 20 henkilöä. Erittäin moni antoi palautetta, jota itse en tullut ajatelleeksi, koska pidin asioita selvänä. Tein muutoksia sovellukseen lähes jokaisesta palautteesta. Beta-testin aikana sovelluksen nimi vaihtui Haaste-pelimuodon kehittämisen takia, uuden nimen ollessa Maze Challenge.

3.5 Web-liikenne

Palvelimella olevassa MySQL-tietokannassa on tallessa käyttäjien tilastot. Sovellus lähettää HTTP:n POST:ina tiedot palvelimella olevalle PHP-tiedostolle, joka käsittelee ja tallentaa tiedot tietokantaan. Uusien tietojen lisäämisen jälkeen PHP palauttaa JSON-muodossa tietoja, jotka sovellus tarvitsee. JSON:in purkamista varten asensin projektiin Visual Studion pakettienhallintatyökalulla JSON.NET-kirjaston. Luokat, johon JSON:ina tuleva data puretaan, on helppo luoda json2csharp.com-sivuston avulla (KUVA 12).

```
public class Score
{
    public string name { get; set; }
    public int score { get; set; }
}

public class JSONdata
{
    public int score { get; set; }
    public int dailyseed { get; set; }
    public int dailyWins { get; set; }
    public int wp8version { get; set; }
    public int wp7version { get; set; }
    public List<Score> careerScores { get; set; }
    public List<Score> dailyScores { get; set; }
    public List<Score> winsScores { get; set; }
    public string dbSuccess { get; set; }
    public string storeLink { get; set; }
}
```

KUVA 12. JSON:in purkamiseen tarvittavat luokat

3.6 Toimintojen lisääminen

Sovelluksen lähettäessä käyttäjän keräämien tähtien määrän palvelimelle oli mahdollista tehdä ennätyslistat, josta näkee toisten käyttäjien edistymisen. Aloitussivulle lisäsin painikkeen, josta aukeaa ennätyssivu. Sivulla on listattuna suuruusjärjestyksessä muiden käyttäjien ansaitut tähdet, sekä nimimerkki. Listan tiedot on kytketty luokkaan niin, että listan täyttäminen oikealla datalla on helppoa. XMAL:ssa määritellään listan ulkoasu niin, että nimi ja tulos kytketään luokan muuttujiin, tässä tapauksessa Score-luokka sopii mainiosti tähän tarkoitukseen (KUVA 13). Tämän jälkeen storage-muuttujasta haetaan lista, jossa on Uran ennätykset. Ennätykset tallennetaan laskevassa järjestyksessä muuttujaan, joka asetetaan listalle (KUVA 14).

```
<ListBox Name="listScore" HorizontalAlignment="Stretch" Margir
    <ListBox.ItemTemplate>
        <DataTemplate>
            <StackPanel Orientation="Horizontal" >
                <TextBlock Text="{Binding name}"/>
                <TextBlock Text=": "/>
                <TextBlock Text="{Binding score}"/>
            </StackPanel>
        </DataTemplate>
    </ListBox.ItemTemplate>
</ListBox>
```

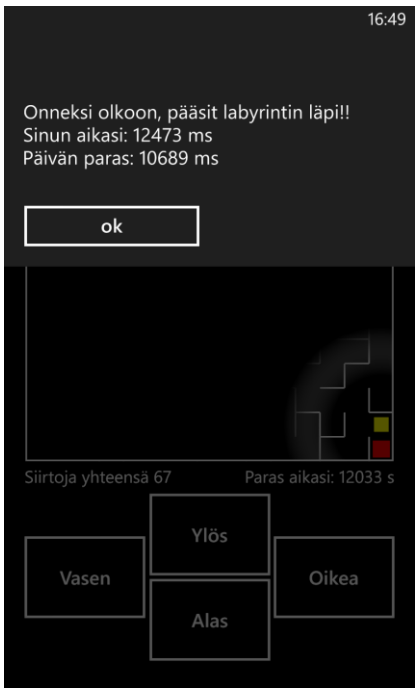
KUVA 13. XAML-koodi ennätyslistalle

```
List<Score> career = ((List<Score>)storage[MyStrings.careerScores]).
                    OrderByDescending(a => a.score).ToList();
listScore.ItemsSource = career;
```

KUVA 14. C#-koodi, jolla näytetään ennätykset

Palautteena testaajiltani sain, että peli on edelleen liian helppo ja kaksi tähteä saa lähes aina toisella yrittämällä, oikean reitin löytää ensimmäisellä yrityksellä. Palautteen jälkeen suunnittelin ja kehitin kolmannen tähden, jonka voi ansaita kahden tähden jälkeen, jos suorittaa labyrintin nopeasti. Tämä lisäsi saman labyrintin yrityksiä, jolloin pelaamiseen käytetty aika kasvaa, koska samaa labyrinttiä täytyy yrittää useamman kerran. Aika, joka pitää alittaa ansaitakseen kolmannen tähden, riippuu lyhyimmän reitin pituudesta. Muutaman eri kertoimen jälkeen päädyin kertoimeen 0,22. Eli jos oikean reitin pituus on 50, alitettava aika on 11 sekuntia.

Kolmannen tähden innoittamana toteutin toisen pelimoodin, jonka nimesin Haasteeksi. Tähtien kerääminen sai nimekseen Ura. Haasteessa yritetään vain mahdollisimman nopeaa aikaa labyrintissä, joka vaihtuu kerran päivässä. Jotta Haasteessa kaikilla käyttäjillä olisi sama labyrintti, niin sovellus saa palvelimelta ladatussa JSON datassa generaattorin alustusluvun, jolla sovellus generoi saman labyrintin kaikille käyttäjille. Myös sen hetken nopein aika saadaan JSONista, jotta siihen voidaan verrata käyttäjän aikaa suoritettuaan labyrintin (KUVA 15).



KUVA 15. Pelaajan aikaa verrataan päivän parhaaseen aikaan Haasteessa

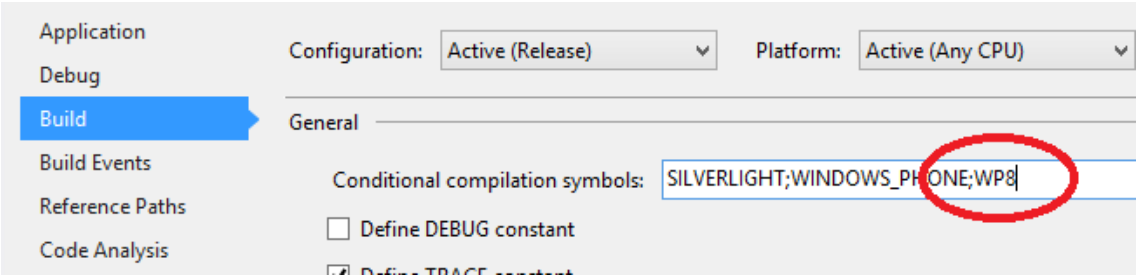
Ennätyssivun muokkasin Windows Phonen pivot-tyyppiseksi, jossa ensimmäisellä välilehdellä on Uran ennätykset ja toisella välilehdellä on nykyisen päivän Haasteen nopeimmat ajat. Haasteen voittaa päivän nopein aika, jonka tarkistaa ja tallentaa tietokantaan palvelimella oleva PHP-tiedosto. Kolmannella ennätyssivun välilehdellä näytetään Haasteen voittojen määrä. (KUVA 16)

KUVA 16. Pivot-tyylinen ennätyssivu

3.7 Kolikot

Alusta alkaen halusin kerätä maksua tästä sovelluksesta, koska aikaisemmat sovellukseni olen jakanut ilmaiseksi. Kehitin kolikot, joilla käyttäjä voi ostaa etuja sovelluksessa. Microsoftin sovelluskaupan kautta tapahtuvan maksun (in-app purchase, IAP) jälkeen, käyttäjä saa kolikoita käyttöönsä. Pelissä olevia molempia pelimuotoja täytyi rajoittaa, jotta käyttäjillä olisi jotain ostettavaa kolikoillaan. Urassa 10 minuutin sijaan uusi labyrintti luodaan 60 minuutin välein, ja yhdellä kolikolla käyttäjä voi ostaa uuden labyrintin heti, jotta pystyisi etenemään urallaan nopeammin. Haasteeseen kehitin päivittäisen 15 yrityksen rajan, jonka voi päivittäin poistaa kahdella kolikolla. Koska WP 7 ei tue in-app purchasea, niin WP 7 käyttäjät eivät voi ostaa kolikoita.

Koska WP 7 ja WP 8 versioissa on eri ominaisuudet tarkoittaa tämä, että koodissa täytyy ottaa huomioon kummasta käyttöjärjestelmästä on kyse. Microsoftin sovelluskauppaan täytyy lähettää omat xap-asennustiedostot molemmille käyttöjärjestelmille ja sovelluskauppa valitsee käyttäjän laitteen mukaan mikä asennustiedosto käyttäjälle asennetaan. Asennustiedostoja voi lähettää useita, esimerkiksi jos haluaa eri näytön resoluutioilla tukea eri ominaisuuksia. Käyttöjärjestelmän eri versioille kannattaa luoda omat projektit, mutta itse kooditiedostot voidaan linkittää projektista toiseen. Tällöin koodin ylläpitäminen helpottuu. Lisäksi kannattaa määrittää esimerkiksi WP 8 projektin tiedoista vakio, jolla saadaan eroteltua koodissa toimintoja käyttöjärjestelmien välillä (KUVA 17). Koska tätä vakiota ei ole WP 7 projektissa määritelty niin toiminnot on helppo erotella koodissa. (Windows Phone Dev Center, 2013)



KUVA 17. Projektien välinen eroittelu

Koska WP 7 laitteissa ei ole mahdollista ostaa kolikoita, käyttäjältä piilotetaan elementti, jossa on kolikoiden ostamiseen ja käyttämiseen tarkoitetut napit sekä kolikoiden määrä. WP 8 laitteissa tämä elementti näytetään käyttäjälle (KUVA 18).

```
#if WP8
            stackCoins.Visibility = System.Windows.Visibility.Visible;
#else
            stackCoins.Visibility = System.Windows.Visibility.Collapsed;
#endif
```

KUVA 18. Toiminnallisuuksien erottelu C#-koodissa

4 JULKAISEMINEN

4.1 Windows Phone Store

Windows Phone laitteisiin voi ladata sovelluksia vain Microsoftin Windows Phone Storen kautta. Microsoft tarkistaa kaikki sovellukset ennen kuin sallii niiden olevan ladattavissa sovelluskauppansa kautta.

4.2 Julkaiseminen

Kun sovelluskehittäjä on saanut sovelluksensa valmiiksi ja haluaa julkaista sen Windows Phone Storessa, täytyy sovellus lähettää Microsoftille tarkastettavaksi. Sovelluksen lähettäminen tapahtuu Windows Phone Dev Centerin lomakkeen kautta. Lomakkeeseen täytetään sovelluksen nimi, jolla sen halutaan näkyvän käyttäjille. Lisäksi valitaan mihin kategoriaan sovellus kuuluu ja paljonko sovellus maksaa ja onko sovelluksesta ladattavissa kokeiluversio. Windows Phone SDK luo xap-tiedoston, kun kääntää sovelluksen projektin. Tämä on asennustiedosto, joka liitetään lomakkeeseen. Lomakkeen kautta liitetään sovellukselle 300x300 pikselin kokoinen kuvan, joka näytetään julkaisun jälkeen sovelluskaupassa. Lisäksi sovelluksesta täytyy liittää vähintään yksi kuvakaappaus jokaista tuettua kieltä kohden. Myös sovelluksen kuvaus täytyy kirjoittaa jokaiselle tuetulle kielelle. Tietojen täyttämisen jälkeen lomake kertoo, jos jotain tietoja puuttuu. Jos kaikki tiedot ovat täytetty oikein voi sovelluksen lähettää Microsoftille hyväksyttäväksi. Lomakkeella voi lähettää sovelluksen myös betana, jolloin Microsoftin hyväksymisprosessia ei käydä lävitse. (Windows Phone Dev Center, 2013b)

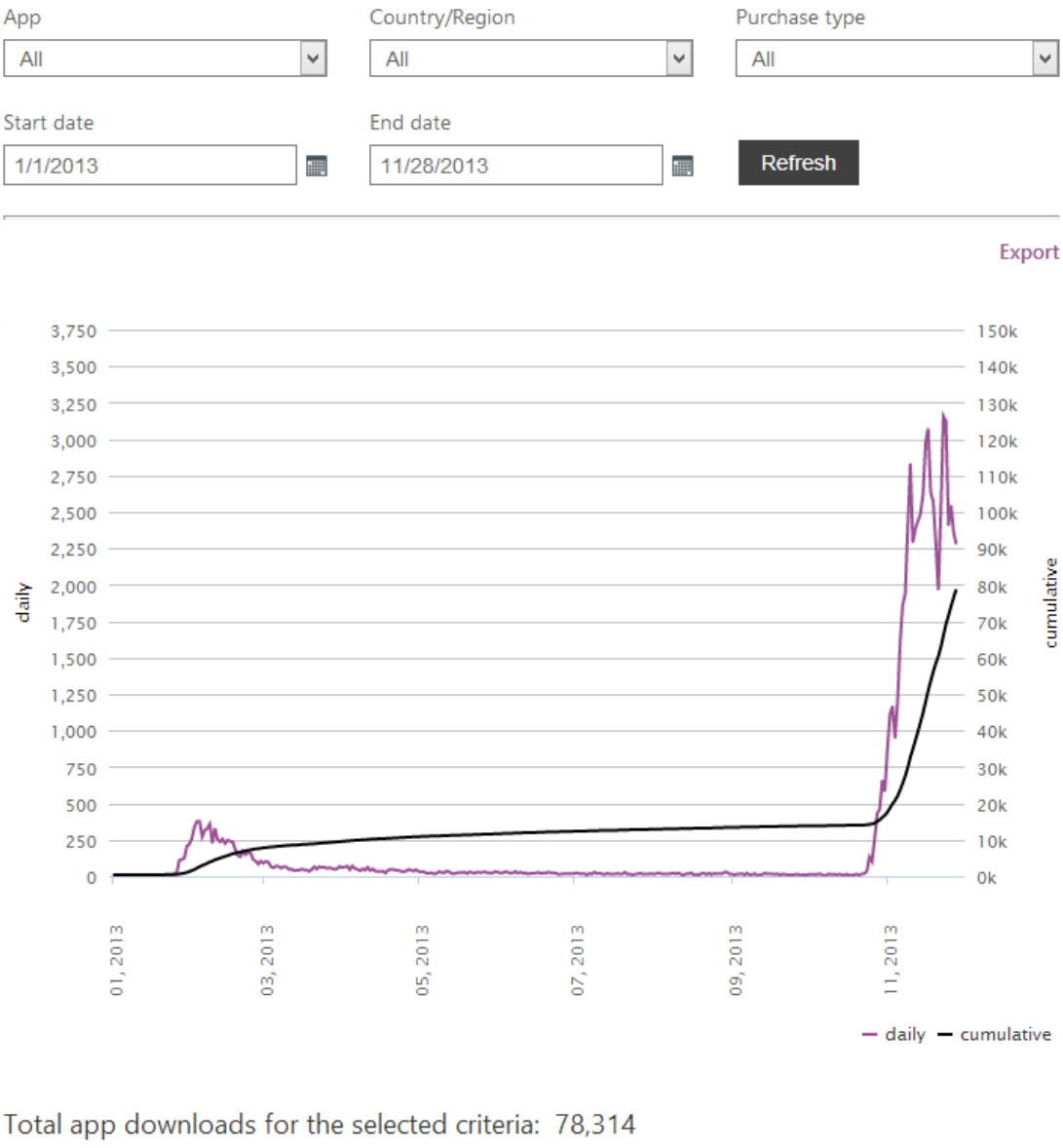
4.3 Sovelluksen tarkistusprosessi

Sovelluksen lähettämisen jälkeen Microsoft tarkistaa, että sovellus on Windows Phonen käytäntöjen ja käyttökokemuksen mukainen. Microsoftilla tarkastetaan sovelluksen toiminta kaikilla sovelluksen tukemilla näyttöresoluutioilla sekä eri muistisilla laitteilla. Testattavina asioina on sovelluksen luotettavuus, kun tapahtuu jokin käyttöjärjestelmä tason tapahtuma, esimerkiksi puhelu, herätyskello tai muu vastaava tapahtuma. Tällöin sovelluksen tulee jäädä taustalle ja toimia virheettömästi myös tapahtuman jälkeen.

Sovelluksen tulee olla testattavissa, jos sovellus vaatii sisäänkirjautumisen, täytyy toimivat tunnukset tarjota Microsoftille, jotta sovelluksen voi testata. Microsoft testaa myös sovelluksen suorituskykyä. Sovelluksen täytyy näyttää jotain (esimerkiksi latausikkuna) 5 sekunnin kuluttua sovelluksen aukaisemisesta ja 20 sekunnin kuluttua sovelluksen täytyy olla käytettävissä. Jos sovellus käyttää paikkatietoa tai push-viestiä täytyy käyttäjällä olla mahdollisuus poistaa ne käytöstä sovelluksen käyttöliittymästä. Käyttäjän kuunnellessa musiikkia, ei käynnistettävä sovellus saa pysäyttää musiikkia automaattisesti, vaan musiikin vaihtaminen sovelluksen omaan musiikkiin täytyy varmistaa käyttäjältä. Sovellusten musiikkien voimakkuus täytyy olla säädettävissä sovelluksen käyttöliittymästä. Hyväksymisprosessin jälkeen asiasta ilmoitetaan kehittäjälle sähköpostilla. Mikäli sovelluksessa on jotain korjattavaa, tarjoaa Microsoft hyvät ohjeet millaisen virheen he löysivät, sekä miten sovelluksen tulisi toimia, jotta se menisi hyväksymisprosessin läpi. Kehittäjän korjattua sovelluksensa, täytyy xap-tiedosto lähettää uudestaan hyväksyttäväksi. Hyväksytty sovellus julkaistaan kaupassa automaattisesti, mikäli kehittäjä on tämän sallinut lomakkeesta, jolla sovellus lähetettiin tarkastettavaksi. Mikäli kehittäjä ei halunnut automaattista julkaisua hyväksymisprosessin jälkeen, hän voi julkaista sovelluksen haluamanaan aikana Dev Centerin kautta. Dev Centerin kautta voi myös seurata sovelluksen hyväksymisprosessin vaiheita. (Developer Network, 2013a)

4.4 Julkaisun jälkeen

Kun sovellus on julkaistu kaupassa, kehittäjä voi seurata latauksien määrää Dev Centerin -sivuston kautta. Sovelluksen päivitys tehdään tämän sivuston kautta samanlaisella lomakkeella kuin sovelluksen julkaisu. Sovelluskaupan kautta näkee vain oman alueen käyttäjien antamat arviot sovelluksesta, mutta Dev Centerin kautta kehittäjä näkee kaikkien alueiden arviot sovelluskohtaisesti. Sovelluksien latauksia voi seurata päiväkohtaisesti sekä kumulatiivisesti. Valikosta voi valita kaikkien sovellusten latauksien sijasta myös tietyn sovelluksen lataukset. Lisäksi maan tai alueen mukaan sekä ostotyypin (kaikki, ilmainen, kokeilu, ostettu, beta) mukaan voi rajata näytettävien latauksien määrää. Latauksien määrän voi näyttää haluamana aikavälillä. (KUVA 19)



KUVA 19. Sovellusteni latauksien määrä vuonna 2013

Dev Centeristä näkee myös sovelluksen kaatumisien lukumäärän. Tieto sovelluksen kaatumisesta lähetetään Microsoftille, mikäli käyttäjä on laitteen asetuksista sallinut laitteen lähettää palautetta. Dev Centerin kautta kehittäjä saa kaatumisista Excel-taulukon, jossa on kerrottu tarkemmin mitä sovelluksen kaatuessa on tapahtunut. Tämän kautta saaduilla tiedoilla kehittäjän on mahdollista korjata sovellustaan ja tarjota käyttäjille parempaa käyttökokemusta. Myös In-app purchase –tilastoja voi seurata Dev Centerin kautta.

Dev Centeristä on saatavilla myös sovellus WP:lle. Kehittäjän tulee kirjautua sovellukseen tunnuksilla, joita ilman sovellusta ei voi käyttää. Valittavana on viiden päivän, yhden kuukauden, kuuden kuukauden ja vuoden graafit latauksista sekä

kaatumisista. Myös kaikkien alueiden arvostelut voi lukea sovelluksen kautta. Sovelluksen saa päivittämään aloitusnäytön tiiltään, jossa näkyy latausten ja kaatumisien määrä. Sovelluksen kautta kehittäjä voi jakaa suoran linkin omaan sovellukseensa. (KUVA 20)

KUVA 20. Dev Center –sovellus (Windows Phone Store, 2013)

5 POHDINTA

Opinnäytetyöni aiheen valintaa pidän onnistuneena, koska mobiilisovellukset ovat nousevassa trendissä tällä hetkellä ja Windows Phone on nopeimmin kasvava mobiilikäyttöjärjestelmä.

5.1 Sovelluksen menestys

Sovelluksesta on eri versiot WP 7 ja WP 8 laitteille, joten tiedossani on käyttäjien jakauma näiden käyttöjärjestelmien välillä. Sovelluksen käyttäjistä käyttää WP 7 käyttöjärjestelmää 48 % ja WP 8 käyttäjiä on 52 %. Vaikka WP 7 käyttäjät eivät voi ostaa sovellusta, niin olen tyytyväinen, että tuen myös vanhaa versiota.

Ensimmäisen viikon aikana sovelluksen oli ladannut 1287 eri pelaajaa. Kahdessa viikossa sovellusta ladattiin 7608 kertaa. Neljän viikon aikana sovellusta ladattiin 41 111 kertaa. Dev Centerin mukaan sovellustani on ladattu eniten Brasiliassa, jossa sovellusta on ladattu yli kaksi kertaa enemmän kuin seuraavaksi suosituimmassa maassa, Meksikossa. Sovelluksessa on painike, josta pääsee sovelluskauppaan arvostelemaan sovelluksen. Kuukauden aikana sovelluksen on arvostellut 560 käyttäjää, joista suurin osa on positiivisia. Dev Centeristä ei näe suoraan keskiarvoa kaikille arvosteluille, sovelluskaupassa näytetään keskiarvo vain aluekohtaisesti. Sovelluksessa on myös painike, josta voi lähettää minulle palautetta. Tämän painikkeen kautta olen saanut useita sähköposteja, joista suurin osa sisältää vain Windows Phonen allekirjoituksen, mutta myös positiivista palautetta olen saanut tämän kautta.

Sovelluksen voi laittaa sovelluskauppaan maksullisena, joka sisältää ilmaisen kokeilu version. Sovellus kuitenkin näkyy kaupassa maksullisena, jolloin osa käyttäjistä ei edes lue sovelluksen kuvausta tarkemmin. Kuvauksesta voisi selvitä, että sovellus on ilmainen käyttää, mutta maksamalla saa esimerkiksi mainokset pois. Päätin jakaa sovelluksen ilmaisena ja sovelluksen sisällä on mahdollisuus ostaa kolikoita, joilla saa etuja. Ilmaisuuden johdosta sovellus pääsi sovelluskaupassa ”uudet + suosiota kasvattaneet” –listalla parhaimmillaan sijalle 8. Koska sovelluksessa olevasta painikkeesta pääsee helposti kauppaan arvostelemaan sovelluksen, on se saanut monta positiivista arvostelua ja sovellus on ollut ”parhaat ilmaiset”-listalla sijalla 40.

5.2 Sovelluksen jatkokehitys

Seuraavaksi sovellukseen on kehitteillä kamppailu-pelimuoto, jossa pelaaja voi haastaa toisen pelaajan. Molemmat pelaajat laittavat kolikoita panokseksi ja voittaja saa kaikki kolikot. Kamppailu-pelimuodon haluan olevan hieman erilainen kuin muut pelimuodot, säilyttäen kuitenkin pelin hengen. Uusi pelimuoto voisi sisältää koko labyrintin täydeltä kerättäviä esineitä, ja enemmän esineitä 10 sekunnin aikana kerännyt pelaaja voittaa. Toisen pelaajan haastaminen kamppailuun voisi tapahtua NFC:n avulla, jolloin pelaaja voisi haastaa vain tuttujansa. Tässä tapauksessa pitäisi WP 7 käyttäjät jättää kamppailun ulkopuolelle, koska WP 7 laitteissa ei ole NFC tukea. Jos myös WP 7 käyttäjille haluan jakaa uuden pelimuodon, täytyy kamppailuun haastaminen tapahtua esimerkiksi nimimerkin kautta. Lisäksi WP 7 käyttäjillä ei tällä hetkellä ole kolikoita käytössä, joilla he voisivat osallistua kamppailuun. Kolikoita voisi kerätä esimerkiksi ura-pelimuotoa pelaamalla. Tässä tapauksessa pitäisi muuttaa kolikoiden arvoa, jolloin uuden labyrintin saa yhden kolikon sijasta kymmenellä kolikolla. Kolikoita kaupasta ostaessa kolikoita saisi kymmenkertaisen määrä nykyiseen verrattuna.

Myös Android-version tekeminen sovelluksesta on ollut mielessä koko kehittämisen ajan. Tämä varmaankin tapahtuu sen jälkeen, kun saan edellä mainitut jatkokehitysideat toteutettua.

LÄHTEET

Blogging Hits, The Brief History of Windows Phone. Luettu 20.11.2013. http://www.bloggінghits.com/2013/07/04/the-brief-history-of-windows-phone/

Developer Network, App certification requirements for Windows Phone, Luettu 10.10.2013a. http://msdn.microsoft.com/en-us/hh184843

Developer Network, Designing for Windows Phone. Luettu 20.11.2013b. http://msdn.microsoft.com/en-us/library/ff979338(v=expression.40).aspx

Gamedev, C# Workshop - Project 1: Maze Generator. Luettu 26.8.2013. http://www.gamedev.net/blog/1565/entry-2255278-c-workshop-project-1-maze-generator/

MobiiliBlogi, Windows Phone 8:n uudistukset koottuna. Luettu 13.11.2013. http://www.mobiiliblogi.com/2012/06/22/windows-phone-8n-uudistukset-koottuna/

Notebooks, A brief history of Windows Mobile. Luettu 4.11.2013. http://notebooks.com/2010/04/12/a-brief-history-of-windows-mobile/

Ohjelmointiputka, Koodivinkit: QB: Leveyshaku ja syvyyshaku. Luettu 8.9.2013. http://www.ohjelmointiputka.net/koodivinkit/25043-qb-leveyshaku-ja-syvyyshaku

Windows Phone Dev Center, How to target multiple versions with your app for Windows Phone. Luettu 9.10.2013a. http://msdn.microsoft.com/jj206997

Windows Phone Dev Center, Submit your app. Luettu 20.10.2013b. http://msdn.microsoft.com/jj206724

Windows Phone Store, Dev Center. Luettu 20.11.2013 http://www.windowsphone.com/s?appid=2d3063c2-4b29-4e69-9c03-50b67b0e6aec