Olli Lahtinen ja Joonas Alhonen

2D MOBIILIPELIN KEHITTÄMINEN
CASE: HOPPING PENGUIN

Opinnäytetyö
Kajaanin ammattikorkeakoulu
Luonnontieteiden ala
Tietojenkäsittely
Syksy 2014

KAJAANIN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ
TIIVISTELMÄ

Koulutusala Luonnontieteiden ala	Koulutusohjelma Tietojenkäsittelyn koulutusohjelma
Tekijä(t) Olli Lahtinen ja Joonas Alhonen	
Työn nimi 2D mobiilipelin kehittäminen, Case: Hopping Penguin	
Vaihtoehtoiset ammattiopinnot	Toimeksiantaja
Aika Syksy 2014	Sivumäärä ja liitteet 52

Opinnäytetyössä kuvataan Hopping Penguin -mobiilipeliprojekti hyvin käytännönläheisesti. Työssä paneudutaan ensin pelinkehityksen teoriaan ja työkaluihin, jonka jälkeen tarkastellaan tarkemmin Hopping Penguin projektia ja siinä käytettyjä ratkaisuja teknisestä näkökulmasta.

Peliprojektin tarkoituksena oli kehittää julkaistava sovellus, jolla olisi kaupallista potentiaalia. Lisäksi tarkoituksena oli kehittää ryhmämme taitoja tulevaa yhteistä yritystämme varten. Peliprojekti saatiin valmiiksi ja se julkaistiin yksinoikeudella Windows Phone 8 – laitteille saamamme rahoituksen ehtojen vuoksi. Hopping Penguin -pelillä on kirjoitushetkellä yli 300 000 latausta Windows Phone 8 – alustalla.

Opinnäytetyössä käydään läpi mobiilipelin kehittäminen julkaistavaksi tuotteeksi cocos2d-x sekä Unity -pelinkehitysympäristöissä ja paneudutaan Hopping Penguinin palvelinratkaisuihin. Tarkastelun kohteena ovat myös pelianalytiikka sekä peliprojektin palvelinjärjestelmä.

Lopuksi käydään läpi projektin ongelmakohdat ja pohditaan mitä projektissa olisi voinut tehdä paremmin.

Kieli	Suomi
Asiasanat	Unity, pelinkehitys, mobiilipeli
Säilytyspaikka	☐ Verkkokirjasto Theseus ☐ Kajaanin ammattikorkeakoulun kirjasto

KAJAANIN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

THESIS ABSTRACT

School	Degree Programme
Business	Business Information Technology

Author(s)
Olli Lahtinen and Joonas Alhonen

Title
2D Mobile Game Development, Case: Hopping Penguin

Optional Professional Studies	Commissioned by

Date	Total Number of Pages and Appendices
Autumn 2014	52

This thesis describes a project of developing a mobile game called Hopping Penguin in a practical manner. First, we go through theory and tools in game development in general. Then we will take a closer look at the project of Hopping Penguin and the solutions used in it from a technical perspective.

The purpose of the game project was to develop a game that would have commercial potential. In addition, we wanted to improve our group's development skills for our future company. The project was finished successfully and Hopping Penguin was released exclusively for Windows Phone 8 –platform. Currently Hopping Penguin has been downloaded over 300 000 times on Windows Phone 8 –platform.

This thesis shows how to develop a commercial mobile game using cocos2d-x and Unity platforms and what kind of server side solutions were used in the project. Game analytics and server backend in Hopping Penguin is also covered.

Finally, we will revisit the problem areas of the project and ponder how certain things could have been done better.

Language of Thesis Finnish

Keywords Unity, game development, mobile game

Deposited at
☐ Electronic library Theseus
☐ Library of Kajaani University of Applied Sciences

ALKUSANAT

Opinnäytetyön tekemisestä on ollut meille suuri hyöty. Työn suunnittelu ja tekeminen pakottivat meidät selvittämään käytäntöjämme, sekä perehtymään syvemmin pelinkehittämisen teoriaan.

Koko projekti on opettanut meille todella paljon. Hopping Penguin on ryhmämme ensimmäinen kaupallinen tuotos sekä portti yrittäjän arkeen. Projekti on antanut meille realistisen kuvauksen siitä, miten mobiilipelimarkkinat toimivat ja mitä tulevaisuudelta voi odottaa. Pelin tekemisen ohessa on oppinut paljon uutta asiaa, joista osa kantapään kautta.

Haluamme kiittää Kimmo Nikkasta sekä AppCampusta korvaamattomasta avusta pelin kehityksessä.

SISÄLLYS

SYMBOLILUETTELO

HTTP	Tiedonsiirtotekniikka, käytetään mm. verkkosivuissa
TCP	Tiedonsiirtoprotokolla
UDP	Tiedonsiirtoprotokolla
CloudSync	Ominaisuus Hopping Penguinissa, jonka avulla voidaan tallentaa pelin eteneminen palvelimelle
IDE	Ohjelmointiympäristö
SDK	Software Development Kit. Sarja työkaluja ohjelmiston kehittämiseen
API	Ohjelmointirajapinta
iOS	Applen käyttöjärjestelmä mobiililaitteille
OS X	Applen käyttöjärjestelmä Mac-tietokoneille
ppi	Pixels per inch, näytön erottelutarkkuus
GGJ	Global Game Jam -pelinkehitystapahtuma
XML	Extensible Markup Language. XML merkintäkieli.
KPI	Key Performance Indicator.
REST	Representational State Transfer. HTTP-protokollaan perustuva arkkitehtuurimalli.

1 JOHDANTO

Mobiililaitteet ja -sovellukset ovat kehittyneet hyvin nopeasti viime vuosina. Mobiilipelien markkinoiden odotetaan tuplaantuvan vuoteen 2016 mennessä. Nopea taloudellinen kehitys tuo myös haasteita kehittämiseen, sillä kilpailu mobiilipelien saralla on erittäin kovaa. Pelien kehityksessä käytettävät teknologiat ja kehitysympäristöt kehittyvät huimaa vauhtia, ja perässä pysyäkseen on oltava valmis nopeisiin muutoksiin. Sosiaalisen median käyttö peleissä ja pelaajien psykologian ymmärtäminen on nykyään tärkeämpää. Analytiikat ja verkkopalvelut ovat nykyisin arkipäivää mobiilisovelluksissa. (1)

Tämän opinnäytetyön aiheena on 2D mobiilipelin kehittäminen sekä Hopping Penguin -peliprojektin lähempi tarkastelu. Opinnäytteen tarkoituksena on kuvata peliprojektin vaiheita teknisestä näkökulmasta, sekä tarjota näkökulmaa mobiilipelin kehitykseen käytettävistä teknologioista ja haasteista.

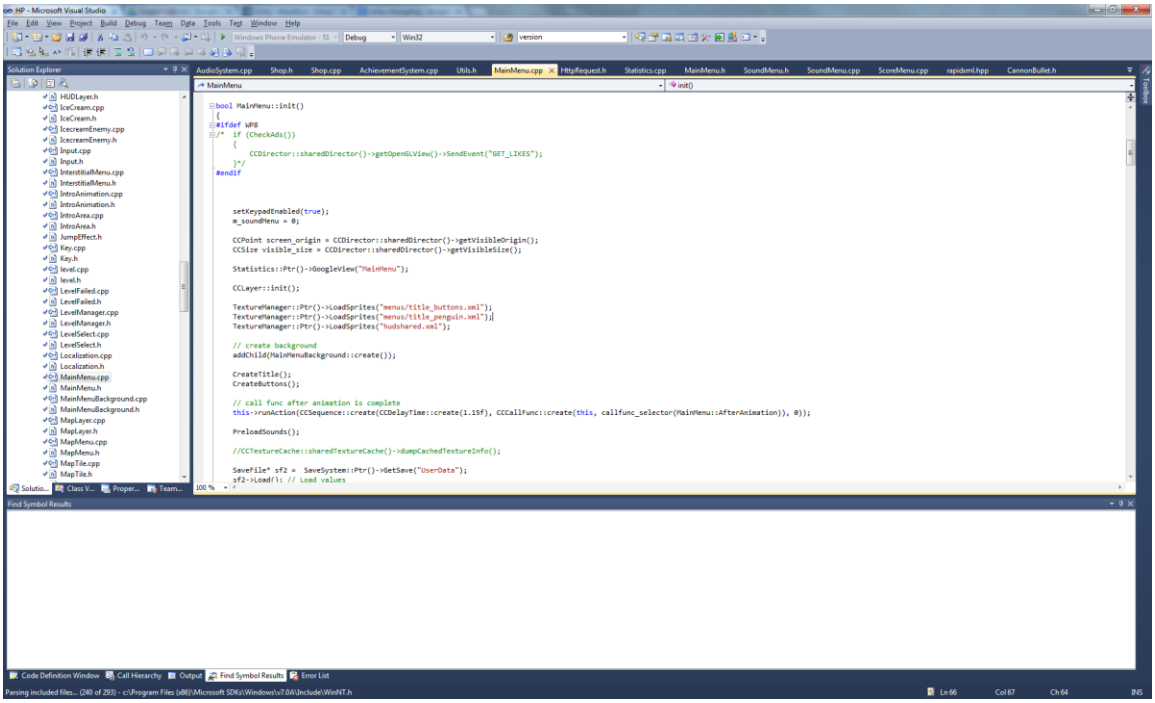
Aluksi perehdytään yleisesti pelien kehitysympäristöihin ja tutkiskellaan Unity-pelinkehitysympäristöä tarkemmin. Käymme yleisesti läpi projektin eri vaiheet ja sen rakenteen. Näiden lisäksi syvennymme pelien analytiikkoihin sekä pelimme palvelinjärjestelmään. Lopuksi työssä pohditaan projektia kokonaisuutena omin sanoin.

Tämä työ pyrkii antamaan tietoa mahdollisimman monesta pelinkehitykseen liittyvästä seikasta kehittäjän näkökulmasta. Aloittaville pelinkehittäjille työn sisältämä tieto voi olla hyödyllistä, sillä pelinkehitys on hyvin monisyinen prosessi, jota voi olla hankala hahmottaa sitä aloittaessa.

2 KEHITYSYMPÄRISTÖT

Kehitysympäristöt ovat ohjelmistokehitystä nopeuttavia ja helpottavia työkaluja. Ohjelmistokehityksessä käytettäviä työkaluja ja ympäristöjä on valtava määrä, ja yleensä tietylle järjestelmälle kehitettäessä vaaditaan tietyn kehitysympäristön käyttöä. Esimerkiksi iOS-laitteille kehittämiseen vaaditaan Applen Mac OS X -käyttöjärjestelmä, sekä XCode-kehitysympäristö. Kehitysympäristöistä puhuttaessa tulee vastaan useita lyhenteitä. (2)

IDE tarkoittaa ohjelmointiympäristöä ja se on ohjelmisto, joka tarjoaa kattavat työkalut sovelluskehitystä varten. Yleensä IDE sisältää tekstieditorin, jolla varsinainen ohjelmakoodi kirjoitetaan. Esimerkki IDE:stä on Kuviossa 1 esitetty Microsoftin Visual Studio. (3) (4)



Kuvio 1. Visual Studio 2010 ohjelmiston käyttöliittymä. (Kuvankaappaus ohjelmistosta)

SDK tarkoittaa sarjaa työkaluja, jotka mahdollistavat ohjelmistokehityksen tietylle järjestelmälle. Yleensä laitevalmistaja tarjoaa SDK:n ohjelmistokehittäjille. SDK eroaa IDE:stä siten, että yleisesti pelkkä SDK ei riitä sovelluskehitykseen, vaan sitä käytetään IDE:n kanssa. Esimerkki SDK:sta on Windows Phone 8 SDK, jonka avulla voidaan kehittää sovelluksia Windows Phone 8 -laitteille. (5)

API tarkoittaa ohjelmointirajapintaa. Se on määritelmä, jonka mukaan ohjelmoijat voivat käyttää esimerkiksi SDK:n tarjoamia toimintoja. Se voi sisältää protokollia ja työkaluja ohjelmistokehityksen helpottamiseksi. (6)

2.1 Alustariippumaton pelinkehitys

Jos peli on suunniteltu julkaistavan useammalle kuin yhdelle alustalle, tulee se ottaa huomioon heti projektin alkuvaiheessa. Yhdelle alustalle kehittäminen ja projektin kääntäminen myöhemmin toiselle alustalle on kallista ja aikaa vievää. Natiivikehitystyökalut mahdollistavat yleensä vain tietylle alustalle kehittämisen, joten kehittäjien on usein turvauduttava kolmannen osapuolen kehitystyökaluihin. Alustakohtaisia ongelmakohtia on useita, sillä esimerkiksi tietoturva-asetukset eri alustoilla vaikuttavat tiedosto-oikeuksiin, eli onko ohjelmalla oikeus lukea tai kirjoittaa tiettyihin tiedostoihin tai hakemistoihin. (7)

Valitut alustat vaikuttavat myös käytettävän ohjelmointikielen valintaan. Usein alustariippumattomassa kehityksessä käytetään joko C++:aa, skriptikieltä kuten Python tai sellaista ohjelmointikieltä, jota voi suorittaa virtuaalikoneessa alustasta riippumatta, usein Javaa tai C#:ia Mono-kehitysympäristön kanssa.

Alustariippumattomassa pelinkehityksessä tulee ottaa huomioon myös käytettävä laite, eikä pelkkää ohjelmistoalustaa. Esimerkiksi tietokoneella suunniteltu ja testattu peli ei välttämättä tuota vastaavaa käytettävyyttä kosketusnäyttölaitteilla. (8)

Nykyaikaisessa ohjelmistokehityksessä alustariippumattomuus on todella tärkeä asia, sillä yksittäiselle mobiilialustalle kehitettäessa sovelluksen potentiaalinen käyttäjäkunta rajoittuu merkittävästi. Pelkästään iOS laitteita käyttäviä on satoja miljoonia. Erilaisten laitteiden välillä on suuria eroja myös näytön resoluutioissa sekä pistetiheyksissä, joka vaikuttaa käyttöliittymäsuunnitteluun oleellisesti. Pelkästään Android-laitteita on tuhansia erilaisia. (9) (10)

2.2 Pelimoottori

Pelimoottori on kehitysympäristö, jonka tarkoituksena on helpottaa erityisesti pelien kehittämistä. Pelimoottori suorittaa perustehtäviä, joita voidaan käyttää kaikissa peleissä. Pelimoottori voi hallita esimerkiksi resurssien lataamisen, grafiikan piirtämisen tai fysiikan mallintamisen. Pelimoottori helpottaa ja nopeuttaa pelien kehittämistä, koska osa pelin teknologiasta on jo valmiina pelimoottorissa. Pelin kehittäjät voivat näin keskittyä itse pelin kehittämiseen ja se voi vähentää myös pelin kehitykseen käytettävää budjettia. (11)

Pelin kehittäminen ilman pelimoottoria on työlästä. Yksinkertaisetkin ominaisuudet vaativat paljon työtä ja pelin kehityksessä työtunnit ovat merkittävä kulu. Esimerkiksi yksittäisen kuvan piirtäminen näytölle saattaa vaikuttaa yksinkertaiselta, mutta todellisuudessa siihen liittyy useita eri toimenpiteitä, mitkä pelimoottori hoitaa puolestasi, mm.

- kuvatiedoston pakkauksen purkaminen ja lataaminen muistiin
- kuva-alueen jakaminen kolmioihin
- kolmion pisteiden sijainnin määrittely tekstuurissa (tekstuurikoordinaatit)
- datan lähetys näytönohjaimelle
- varjostimen ohjelmointi (kuinka data piirretään)

Esimerkkitapauksessa yllä listattujen asioiden lisäksi pelimoottori huolehtii myös optimoinnista. Varsinkin piirtämiseen liittyvät funktiot ovat usein raskaita suorittaa ja ne voivat aiheuttaa pullonkauloja etenkin mobiilipelien kehityksessä, sillä suorituskyky on rajallinen. (12)

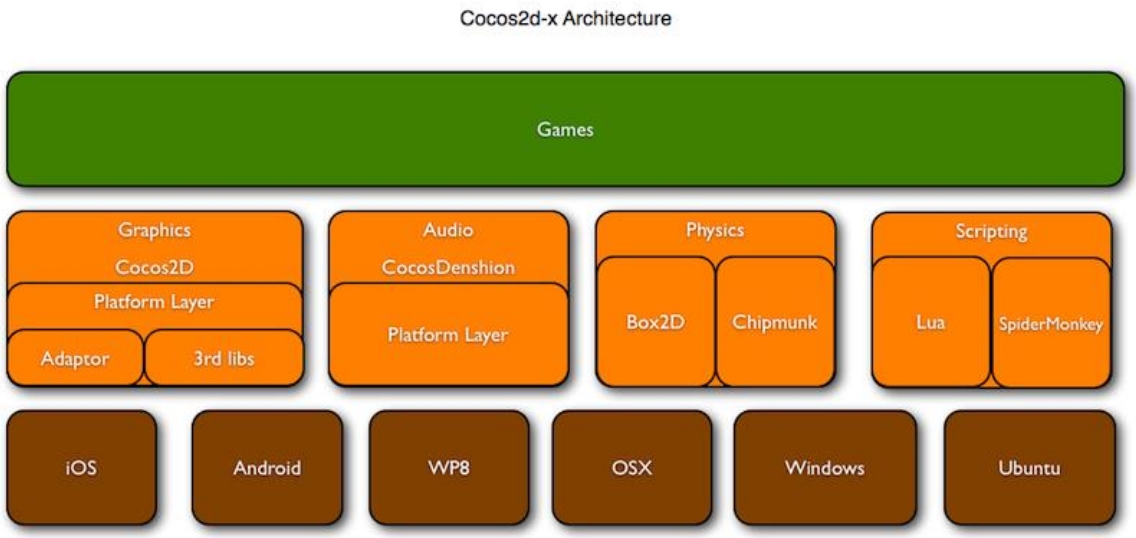
Pelimoottorit tukevat usein useampaa alustaa, joka nopeuttaa kehitystä entisestään. Eri alustat saattavat käyttää eri grafiikkakirjastoja ja niistä huolehtiminen jää pelimoottorin vastuulle.

Pelinkehityksessä käytetään usein apuna skriptikieliä. Ohjelman logiikkaa ei voi yleensä muokata ohjelman suorituksen aikana. Jos ohjelman toimintaa haluaa siis muuttaa, ohjelma täytyy kääntää uudelleen konekielelle. Suurissa projekteissa ohjelman kääntämiseen saattaa kulua jopa tunteja. Ratkaisu tähän ongelmaan on käyttää skriptejä ohjelman logiikan muuttamiseen ohjelman suorituksen aikana. Pelin suunnittelija voi skriptata esimerkiksi pomotaistelun kääntämättä ohjelmaa uudestaan. (13)

Tässä työssä esitellään kaksi suosittua pelinkehitysympäristöä, joita käytettiin Hopping Penguin –pelin kehityksen eri vaiheissa.

2.3 Cocos2d-x

Cocos2d on sovelluskehys, joka on tehty 2D pelien, demojen ja muiden graafisten sovellusten kehittämiseen. Alun perin Cocos2d on kehitetty Python-ohjelmointikielellä, mutta sitä alettiin käyttää yleisesti vasta iPhonelle tehdyn version jälkeen. Cocos2d-x on sen monialustakehitykseen suunnattu versio. Se mahdollistaa pelien kehittämisen usealle alustalle C++-kielellä. Siitä on julkaistu useita versioita ja sitä on kehitetty pitkään, joten sen toimivuus on todettu käytännössä. (14) (15)



Kuvio 2. Cocos2d-x:n arkkitehtuuri (16)

Cocos2d-x on erityisesti kehitetty pelejä varten ja sillä onkin tehty useita suosittuja mobiilipelejä kuten Woogan Jelly Splash ja Fingersoftin Hill Climb Racing. (17)

Cocos2d-x sisältää valtavan määrän pelinkehitystä helpottavia ominaisuuksia. Cocos2d-x sisältää esimerkiksi ohjelmoitavia toimintoja, joilla voidaan laittaa peliobjekti helposti vaikkapa liikkumaan paikasta paikkaan samalla pyörien tietyn akselin ympäri.

Esimerkkejä Cocos2d-x:n ominaisuuksista:

- näkymän hallintajärjestelmä
- peliobjektien hierarkia
- siirtymäefektejä
- edistynyt 2D grafiikka
- skriptaus
- tekstin piirtäminen
- fysiikkamallinnus
- äänet ja musiikki

2.4 Unity

Unity sisältää UnityEditor-pelinkehitysympäristön ja UnityEngine-pelimoottorin. UnityEditor toimii Windows ja OS X -alustoilla, mutta UnityEnginellä luotuja pelejä voi kuitenkin helposti ajaa lukuisilla alustoilla, mikä onkin yksi Unityn valttikorteista. Tuettuja alustoja ovat PC, Mac, Linux, Android, iOS, Windows Phone 8, Blackberry, PlayStation 3, PlayStation 4, PlayStation Vita, Xbox 360, Xbox One ja Wii U. Näiden lisäksi Unity mahdollistaa pelien pelaamisen web-selaimessa erikseen asennettavan Unity Web Player - liitännäisen avulla. Liitännäinen on saatavilla Internet Explorer, Safari, Mozilla Firefox ja Google Chrome selaimille. Unity pitää sisällään myös Asset Store -kaupan, josta voi ostaa muiden käyttäjien tekemiä lisäosia tai resursseja, kuten 3d-malleja. (18)

Unity mahdollistaa pelin kehittämisen usealle alustalle samaan aikaan, ilman että kehittäjä käyttäisi merkittävästi aikaa alustakohtaiseen koodiin. Parhaassa tapauksessa peli toimii kaikilla alustoilla täysin ilman alustakohtaista koodia. Unityn ensimmäiset versiot oli suunniteltu 3D-pelien kehittämiseen, mutta uusimmissa versioissa on täysi tuki myös 2D-pelikehitykseen. (19)

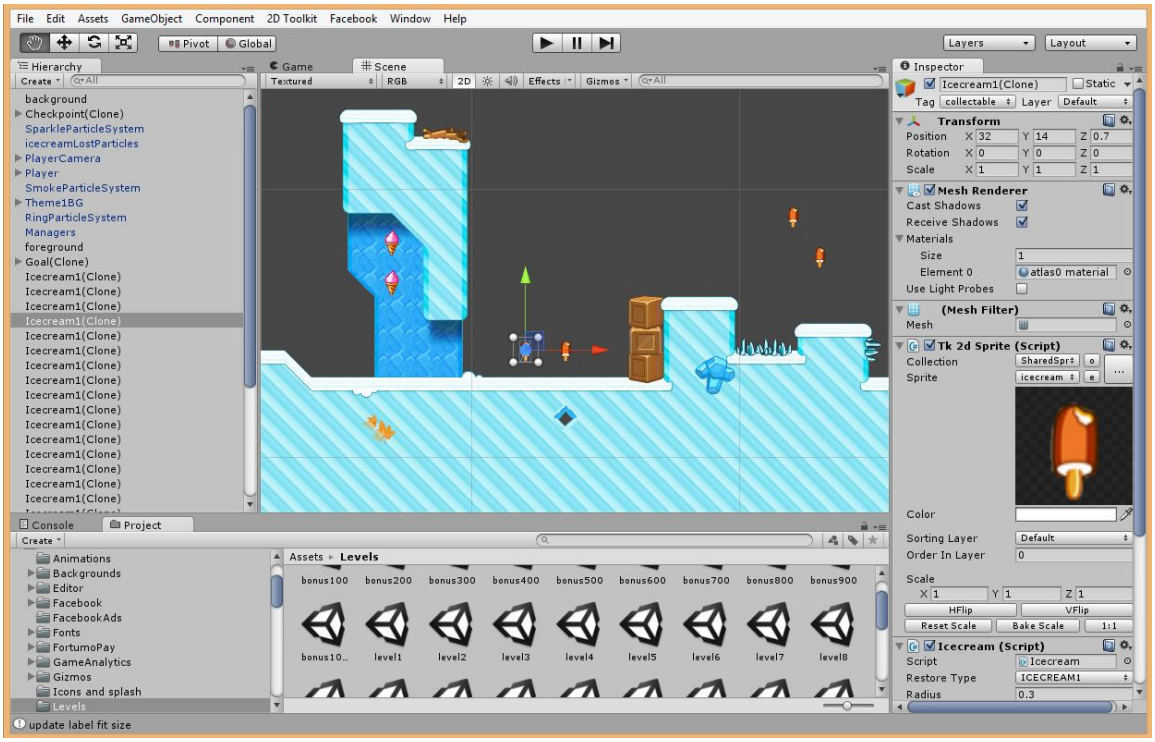
Unity on yleistynyt huomattavasti, ja se onkin nykyään käytetyin pelinkehitysympäristö mobiilipelien kehityksessä. (20)

Unitystä on saatavilla ilmainen Free-versio sekä maksullinen Pro-versio. Pro versioon voi ostaa käyttöoikeuden hintaan $75 kuukaudessa tai kertamaksuna $1,500. Kuukausimaksulliseen versioon sisältyy kaikki tulevat päivitykset, joten se on aina ajan tasalla. Kertaostoksella ostettava lisenssi on rajoitettu ostohetken versioon. Kyseisen version pienemmät päivitykset ovat saatavissa ilmaiseksi, mutta isommistä versiopäivityksistä vaaditaan päivityslisenssi. Ilmaisversiosta puuttuu joitakin ominaisuuksia sekä kehittäjä on pakotettu näyttämään Unity-aloitusruutu pelin käynnistyessä. Jos julkaisijan bruttotulot ylittävät $100,000 vuodessa, Pro-versio vaaditaan.

Unityyn on saatavilla myös Android Pro sekä iOS Pro -lisäosat, jotka tuovat lisää ominaisuuksia kyseisille alustoille, sekä Team License -lisäosa, jonka tarkoitus on helpottaa tiimityöskentelyä mm. Unityyn integroidulla Asset Server -versiohallintatyökalulla. Lisäosat ovat maksullisia. (21)

2.4.1 Käyttöliittymä

Unityn käyttöliittymä voidaan jakaa viiteen pääkomponenttiin: näkymän hierarkia, projektin resurssit, tarkastelu-ikkuna sekä peli- ja näkymä -ikkunat. Seuraavassa kuviossa esimerkki Unityn käyttöliittymässä.



Kuvio 3. Unityn käyttöliittymä Windowsilla. (Kuvankaappaus ohjelmistosta)

Hierarkia-ikkunassa on listattu kaikki peliobjektit, mitkä kuuluvat valittuun näkymään. Hierarkiaa voi järjestellä miten haluaa. Peliobjektien omistussuhteita voi muokata helposti hiirellä raahaamalla objekti paikasta toiseen. Unity käyttää Scene Graph -järjestelmää peliobjektien omistussuhteiden määrittelemiseen. Scene Graph -järjestelmässä peliobjektilla voi olla useita lapsia, mutta vain yksi vanhempi. Peliobjekti sijoitetaan pelimaailmaan vanhemman mukaan.

Resurssi-ikkunassa näkyy kaikki projektin käytettävissä olevat tiedostot. Valmiit peliobjektit voi raahata hiirellä suoraan näkymä-ikkunaan tai hierarkiaan, jolloin peliobjektista tehdään kopio valittuun näkymään.

Tarkastelu-ikkunassa näytetään valitun peliobjektin komponentit sekä niihin liittyvät säädöt ja valinnat.

Näkymä-ikkunassa näkyy valittu näkymä ilman lopullisia tehosteita. Näkymä-ikkunasta voi myös valita peliobjektin klikkaamalla sitä. Valitun objektin kohdalle tulee näkymässä mm. raahain, jolla voi siirtää objektia pelimaailmassa. Peliobjektin komponenteista riippuen, tulee näkymä-ikkunaan myös muita valintoja, esimerkiksi kuvaa voi skaalata erikokoiseksi käyttämällä siihen tarkoitettua työkalua.

Unity-kehitys pyörii käytännössä kokonaan näkymä-ikkunan ympärillä, sillä se on helppokäyttöinen graafinen näkymä, josta näkee välittömästi miltä peli tulee näyttämään, ilman että peliä tarvitsee edes käynnistää. Näkymä-ikkunaa voi käyttää myös pelin pyöriessä.

Peli-ikkunassa näkyy nimensä mukaisesti lopullinen peli, kuten se näkyy loppukäyttäjälle. Peli-ikkunan kokoa voi muuttaa lennosta ja ikkunaan voi tallentaa resoluutioita, jolloin resoluution saa halutun kokoiseksi yhtä painiketta painamalla, joka taas helpottaa testausta huomattavasti.

Unityn vakioasetuksilla peli- ja näkymä-ikkunat ovat käyttöliittymässä samassa kohtaa, joista näkymä-ikkuna on aluksi näkyvillä. Kun peli laitetaan pyörimään, vaihtuu näkymä-ikkunan tilalle peli-ikkuna ja pelin sulkeutuessa ikkunat vaihtavat taas paikkaa. Ikkunat saa halutessaan näkyviin samaan aikaan ja ne voi raahata näytöllä minne tahansa, kuten kaikki muutkin Unityn sisäiset ikkunat.

2.4.2 Komponentit

Unityssä peli rakennetaan peliobjekteista ja niiden välisestä vuorovaikutuksesta. Peliobjektissa on vähintään yksi komponentti, Transform, joka määrittelee objektin sijainnin, rotaation sekä skaalan.

Komponentti on ohjelmakoodia, joka liitetään peliobjektiin. Peliobjektilla voi olla useita eri komponentteja ja jokin komponentti voi olla useassa eri peliobjektissa samaan aikaan. Komponenttien idea on siinä, samaa koodia voidaan käyttää useissa eri peliobjekteissa. Komponentin (koodin) sisältämät muuttujat ja viittaukset voi rakentaa siten, että niitä voi muokata Unityn sisältä, koskematta koodiin. Komponentteja voi tehdä Unityn tukemilla skripti- ja ohjelmointikielillä, mitkä ovat C#, JavaScript ja Boo Script.

Kuvio 4. Peliobjekti valittuna tarkastelu-ikkunassa. Peliobjektissa on Camera Anchor -komponentti. (kuvankaappaus ohjelmistosta)

Esimerkkikuvassa peliobjektilla on Camera Anchor -niminen komponentti, jonka tarkoitus on kohdistaa peliobjekti jonkin tietyn kameran reunakohtaan. Komponentille voi antaa Unityn sisällä tiedon siitä, mihin kohtaan peliobjekti kameran suhteen sijoitetaan ja mihin kameraan viitataan. Näinollen samaa ohjelmakoodia voidaan suorittaa useissa eri peliobjekteissa, mutta eri parametreillä.

2.4.3 Erot perinteiseen pelinkehitykseen

Unityn käyttö eroaa perinteisestä pelinkehityksestä merkittävästi. Perinteisesti pelinkehitys on pyörinyt koodin ympärillä: graafisia käyttöliittymiä ei niinkään ole, vaan koodi vie suurimman osan näytön pinta-alasta. Jos tarvittiin esimerkiksi kenttäeditoria, se piti ohjelmoida itse. Työkalut olivat usein itse ohjelmoituja ja projektikohtaisia.

Unityn lähestymistapa pelinkehitykseen on nykyaikaisempi ja se on tarkoitettu kaikille pelinkehityksen osa-alueille. Unityä käyttää niin ohjelmoijat, pelisuunnittelijat kuin graafikotkin. Suuri osa työskentelystä tapahtuu graafisella käyttöliittymällä ja koodin kirjoitus jää minimiin. Unityn käyttö on hyvin yksinkertaista, eikä vaadi juurikaan tietämystä ohjelmoinnista.

Unityä kehitetään koko ajan kattavammaksi paketiksi, viime aikoina lisättyjä ominaisuuksia ovat mm. kattavat äänityöskentelyominaisuudet ja animointityökalut. Opinnäytetyön kirjoitushetkellä Unity sisältää seuraavia ominaisuuksia:

- MonoDevelop IDE
- animointi
- äänet ja musiikki
- fysiikka
- käyttöliittymä
- suorituskyvyn analysointi
- verkko-ohjelmointi

Perinteisesti edellä mainitut ominaisuudet on hoidettu lukuisia eri työkaluja käyttäen, mutta Unity tarjoaa kaikki ominaisuudet samassa paketissa.

Unity tukee lukuisten kolmannen osapuolten ohjelmien käyttämiä formaatteja. Esimerkiksi laajasti käytetyn Photoshopin PSD-formaatti on tuettuna, joka helpottaa työskentelyä Photoshopin ja Unityn kanssa.

3D Package Support

	Meshes	Textures	Anims	Bones
Maya .mb & .ma[1]	✓	✓	✓	✓
3D Studio Max .max[1]	✓	✓	✓	✓
Cheetah 3D .jas[1]	✓	✓	✓	✓
Cinema 4D .c4d[1 3]	✓	✓	✓	✓
Blender .blend[1]	✓	✓	✓	✓
modo .lxo[2]	✓	✓	✓	
Autodesk FBX	✓	✓	✓	✓
COLLADA	✓	✓	✓	✓
Carrara[1]	✓	✓	✓	✓
Lightwave[1]	✓	✓	✓	✓
XSI 5.x[1]	✓	✓	✓	✓
SketchUp Pro[1]	✓	✓		
Wings 3D[1]	✓	✓		
3D Studio .3ds	✓			
Wavefront .obj	✓			
Drawing Interchange Files .dxf	✓			

[1] Import uses the application's FBX exporter. Unity then reads the FBX file.
[2] Import uses the application's COLLADA exporter. Unity then reads the COLLADA file.
[3] Cinema4D 10 has a buggy FBX exporter. Please see here for workarounds.

Kuvio 5. Unity tukee laajasti eri 3d-mallien formaatteja. (22)

2.5 Versiohallinta

Versiohallinalla tarkoitetaan järjestelmää, jolla voi järjestelmällisesti ylläpitää ohjelman eri versioita. Versiot säilytetään tietovarastossa, joka sijaitsee tyypillisesti palvelimella. Kaikki muutokset (versiot) jäävät järjestelmän muistiin, joka helpottaa esimerkiksi ohjelmointivirheen löytämistä, kun tiedetään missä versiossa virheellinen toiminta alkoi. Jokaiseen versioon voidaan tarvittaessa palata milloin tahansa.

Ohjelmistoprojektin edetessä lähdekoodi kasvaa jatkuvasti. Mitä enemmän lähdekoodia on, sitä hankalampi kokonaisuutta on hallita, etenkin jos koodin muokkaajia on useita. Versiohallinta auttaa kokonaisuuden hallitsemisessa huomattavasti.

Versiohallintaa käytettäessä tulee automaattisesti hoidettua myös projektin versiohallinnassa olevien tiedostojen perustason varmuuskopiointi . Vaikka palvelimella oleva tallennus tuhoutuisi, löytyy jokaisen kehittäjän tietokoneelta joku versio projektista. Uusin versio on aina vähintään kahdella laitteella samanaikaisesti.

Versiohallintaan voi tallentaa kaiken tyyppisiä tiedostoja, mutta järjestelmät on suunniteltu tekstipohjaisten tiedostojen hallintaan. Jos esimerkiksi kaksi ohjelmoijaa muokkaa samaa tiedostoa, järjestelmä osaa yhdistää tiedostot järkevästi versioinnin yhteydessä. Tiedostojen yhdistely onnistuu vain tekstipohjaisissa tiedostoissa, esimerkiksi lähdekoodissa.

Versiohallinta mahdollistaa turvallisen ohjelmiston kehittämisen. Järjestelmästä voi seurata jokaista muutosta yksityiskohtaisesti. Kaikki ohjelmoijat näkevät muiden ohjelmoijien tekemät muutokset, joka puolestaan helpottaa muiden ohjelmoijien tekemien ohjelmointivirheiden selvittämistä.

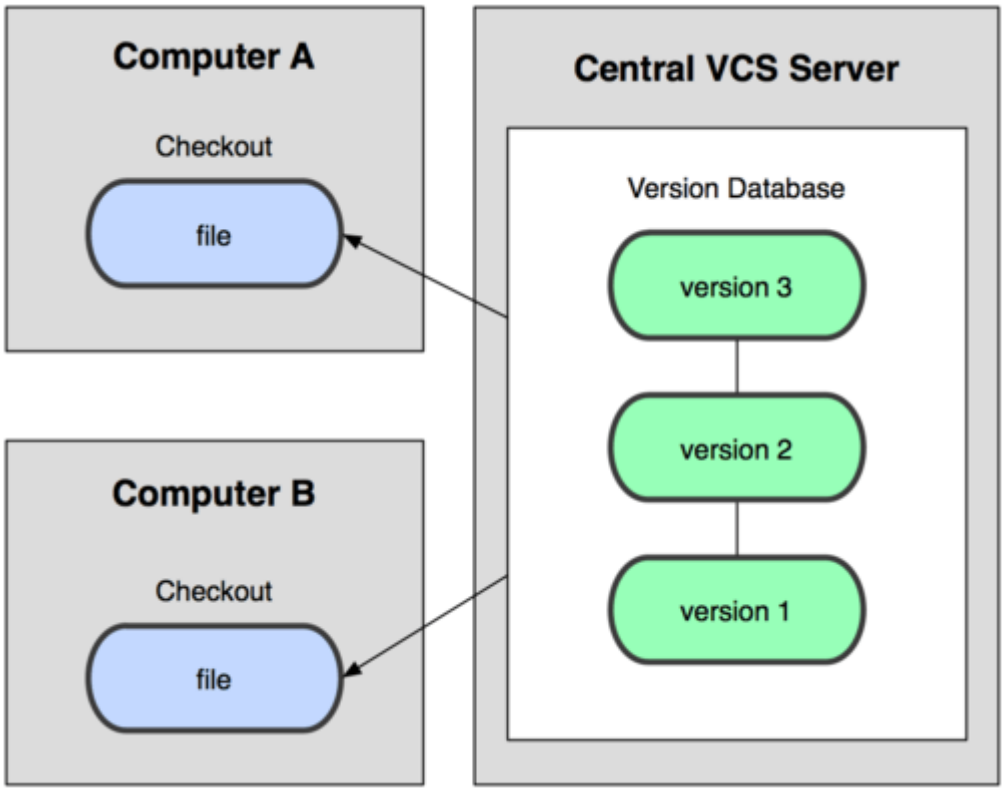
Yksin työskennellessä versiohallinnan käyttö on suotavaa, mutta tiimityöskentelyssä sen käyttö on välttämätöntä. Mitä enemmän henkilöitä on muokkaamassa projektia, sitä tärkeämmäksi versiohallinta muodostuu.

Järjestelmä perustuu siihen, että kehittäjä lataa kopion projektista omalle laitteelleen. Kehittäjä tekee muutokset ja lähettää muokatun version takaisin palvelimelle. Järjestelmä tunnistaa versioiden erot, tallentaa muokatun tiedon sekä tiedon muokkaajasta. Versiohallinnan avulla on siis helppo selvittää milloin tiettyä tiedostoa muokattu, kuka sitä on muokannut ja mitä muutoksia kyseinen versio sisältää verrattuna edelliseen versioon.

2.5.1 Versiohallintajärjestelmät

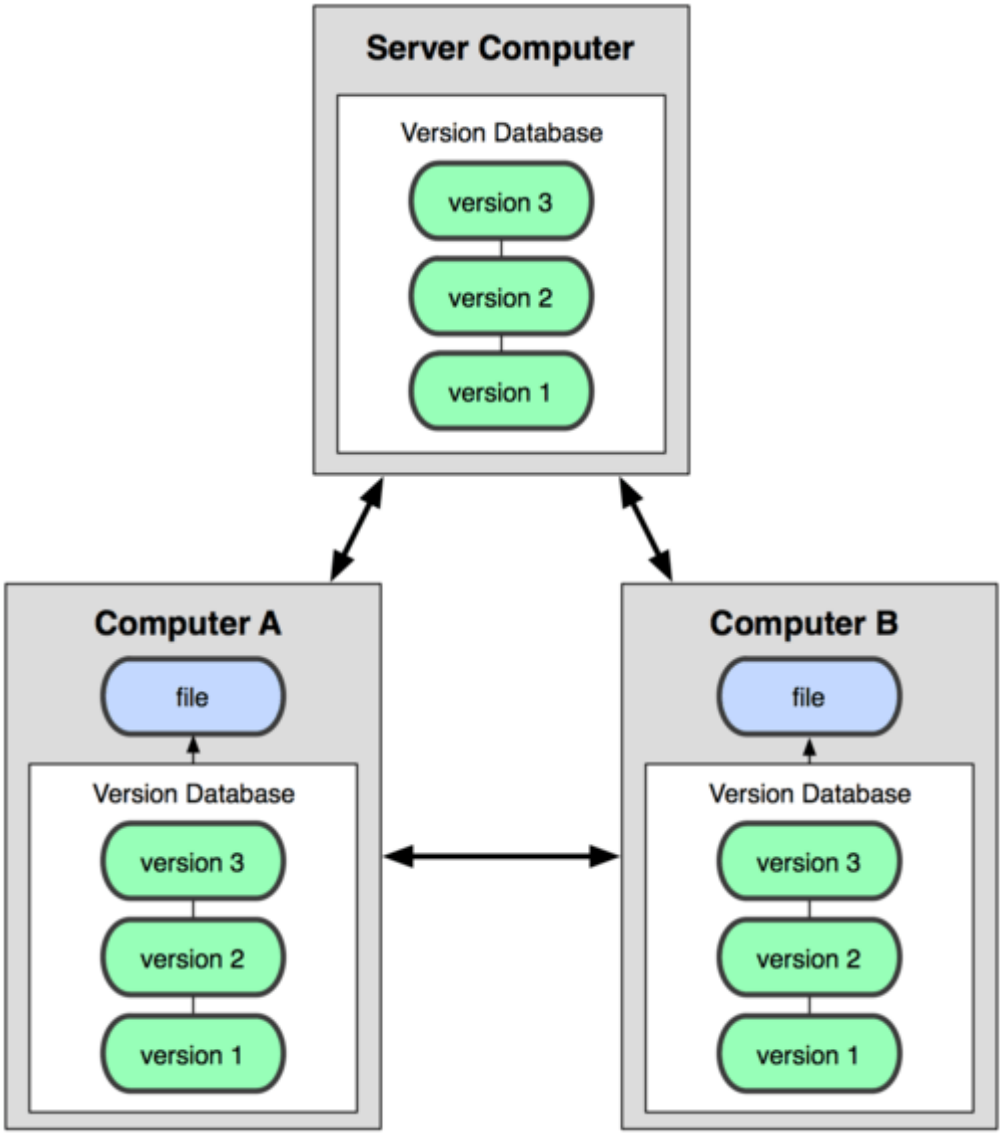
Järjestelmät voidaan jakaa tyypillisesti kahteen ryhmään: hajautetut sekä keskitetyt versiohallintajärjestelmät.

Keskitetyssä järjestelmässä on yksi keskitetty tietovarasto, joka sijaitsee tyypillisesti palvelimella, mutta voi sijaita myös paikallisesti käyttäjän omalla laitteella. Käyttäjän tekemät muutokset tallennetaan keskitettyyn tietovarastoon.



Kuvio 6. Kuvaus keskitetystä versiohallinnasta. (23)

Hajautetussa järjestelmässä käytetään keskitetyn tietovaraston sijaan vertaisverkkoa: jokaisella kehittäjällä on oma kopio koko tietovarastosta, varaston koko historia mukaanlukien. Hajautetusta luonteestaan huolimatta järjestelmä pyörii tyypillisesti palvelimella, jotta esimerkiksi käyttäjien hallinnoiminen on mahdollista. Käyttäjän tekemät muutokset tallennetaan paikalliseen versioon ja paikallinen versio lähetetään kokonaisuudessaan palvelimelle. Hajautettu järjestelmä mahdollistaa kehityksen ilman aktiivista yhteyttä tietovarastoon.



Kuvio 7. Kuvaus hajautetusta versiohallinnasta. (23)

Suosituimpia versiohallintajärjestelmiä ovat Git ja Apache Subversion (SVN). Subversion on keskitetty versiohallintajärjestelmä, joka perustuu avoimeen lähdekoodiin. Subversion on tarkoitettu kaikenlaisten tiedostokokonaisuuksien hallitsemiseen. Tietovarasto voi sisältää siis vaikkapa kuvakirjaston, ei pelkästään koodia. Git on hajautettu versiohallintajärjestelmä, joka perustuu avoimeen lähdekoodiin. Gitin suurimpia eroja Subversioniin ovat:

- Hajautettu järjestelmä mahdollistaa mm. käytön ilman verkkoyhteyttä

- Git on tehokkaampi kuin Subversion
- Git tallentaa tiedoston metadatan, Subversion tallentaa koko tiedoston
- Git on suunniteltu ohjelmistokehityksen tarpeisiin, Subversion on suunniteltu tiedostokokonaisuuksien hallintaan

(24)

2.5.2 Versiohallinta Hopping Penguinin kehityksessä

Hopping Penguinin kehityksessä käytettiin alunperin Subversionia, koska olimme opiskelleet koulussa sen käyttöä sekä ammattikorkeakoulu tarjosi meille ilmaisen Subversion-palvelimen. Myöhemmässä vaiheessa päätettiin kuitenkin siirtyä GitHubin käyttöön koska:

- ammattikorkeakoulun palvelin oli hidas
- ammattikorkeakoulun palvelinta ei voitu käyttää valmistumisen jälkeen
- kehittäjillä oli valmiiksi ostettuna henkilökohtainen GitHub-tili

Git mahdollistaa nopeamman työskentelyn, koska jokaista pientä muutosta ei tarvitse lähettää palvelimelle. Gitiin siirtyminen ei käytännössä vaikuttanut muuhun kuin asiakasohjelman vaihtamiseen TortoiseSVN:stä TortoiseGitiin.

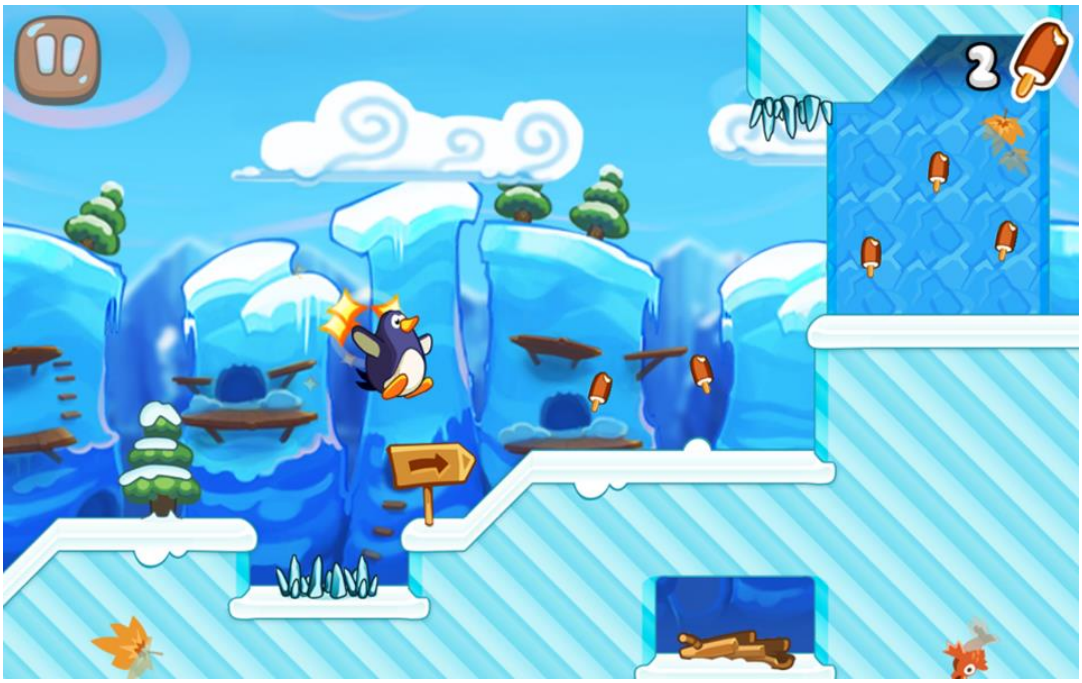
Tekninen versiohallinta Hopping Penguinissa on suhteellisen yksinkertaisella tasolla. Versiohallintaa käytetään vain koodin jakamiseen ohjelmoijien välillä, eikä muita ominaisuuksia käytetty. Kehittäjä ei kokenut tarpeelliseksi määritellä käytäntöjä versiohallinnan käyttämiseen, koska projektissa on vain kaksi ohjelmoijaa ja suurimman osan ajasta he istuvat vierekkäin toimistossa, joka teki kommunikaatiosta vaivatonta.

Jälkeenpäin ajateltunta joillekin käytännöille olisi ollut tarvetta, esimerkiksi kuvaavat viestit versioinnin yhteydessä (mitä muutettiin) sekä ylimääräisten väliversioiden karsiminen kokonaan pois.

Unityyn siirtymisen jälkeen versiohallinnassa oli ongelmia 2DToolkit-lisäosan kanssa. 2DToolkitin metadata ei ollut tarpeeksi yhteensopiva versionhallintaan, minkä seurauksena tekstuurit meni sekaisin ja niitä jouduttiin korjailemaan käsin.

3 HOPPING PENGUIN –PELIPROJEKTIN KULKU

Hopping Penguin on mobiililaitteille kehitetty tasohyppelypeli, jossa seikkailee iloinen pingviini. Pelissä on hyväntahtoinen tarina ja se sopii kaikenikäisille pelaajille sisältönsä ja helppokäyttöisen pelimekaniikkansa puolesta. Pelin kontrollit ovat hyvin yksinkertaiset ja kaikki pelin toiminnot ovat yhden sormikosketuksen takana. Peli on julkaistu Windows Phone Store -kauppapaikalla ja se on ladattavissa Windows Phone 8.0 ja 8.1 -puhelimiin. Kirjoittamishetkellä sitä on ladattu jo yli 300 000 kertaa ja se on saanut erittäin hyviä käyttäjäarvosteluja. Peliprojektin kulku ei ollut mutkatonta ja kokonaisuudessaan peliprojektin lähtötilanteesta lopulliseen julkaisuun kului aikaa yli kaksi vuotta. Pelin kehitys tapahtui pääasiassa ammattikorkeakouluopintojen ohessa.

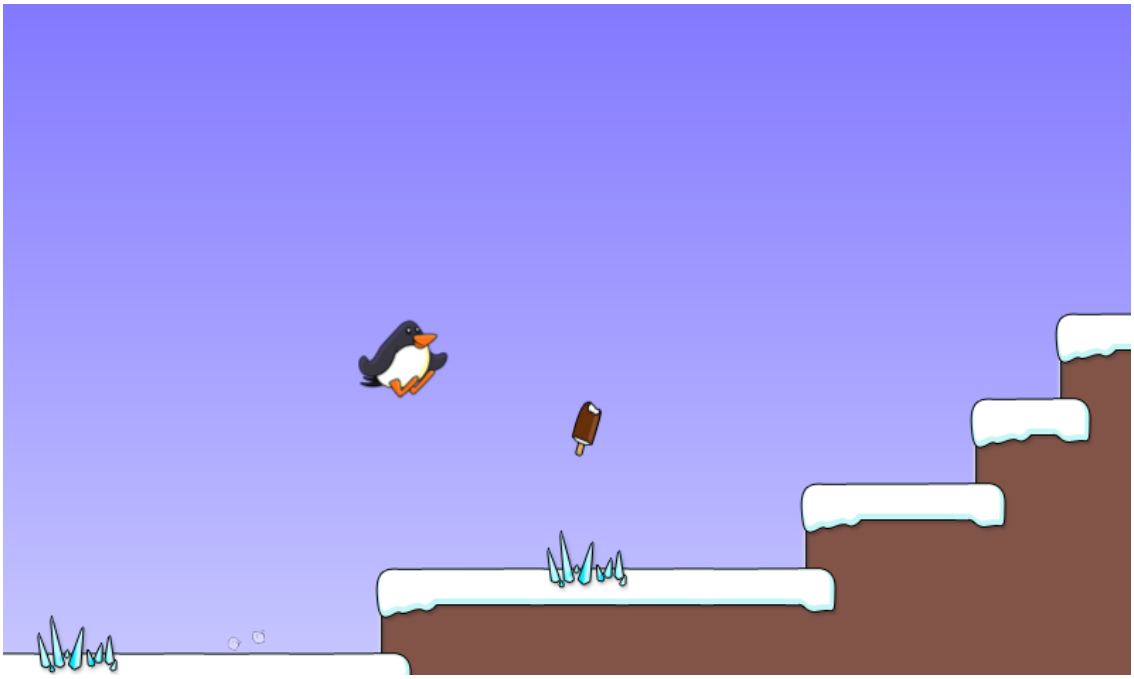


Kuvio 8. Kuvakaappaus Hopping Penguin –pelistä

3.1 Peliprojektin aloitus

Hopping Penguin- peliprojekti sai alkunsa Global Game Jam 2012- tapahtumassa. GGJ on vuosittainen maailmanlaajuinen pelinkehitystapahtuma, jossa pelinkehittäjät ja alan harrastajat

voivat haasta itseään ja pitää hauskaa. Tapahtuman tarkoitus on tehdä tapahtuman aikana julkistettuun teemaan sopiva peli 48 tunnin aikana. Päätimme jo tapahtuman aikana, että teemme pelistä potentiaalisesti kaupalliseksi peliksi sopivan, ja asetimme sen tärkeimmäksi tavoitteeksi. Toinen merkittävä päätös oli ohjausmekaniikan yksinkertaisuus; halusimme pelin olevan välttämättä vain yhdellä kosketuksella helposti pelattavissa, sillä monet menestyneet mobiilipelit ovat hyvin yksinkertaisia. Koska peli piti kehittää hyvin lyhyessä ajassa käytimme kehityksessä C#-ohjelmointikieltä ja XNA-kehitysympäristöä. Näistä meillä oli opintojen aikana tullut eniten kokemusta ja niitä käyttämällä pystyimme keskittymään vain pelin kehittämiseen. Saimme 48 tunnin aikana mielestämme toimivan ja hauskan pelimekaniikan aikaiseksi, ja lopullinen tuotos tuntui oikealta peliltä. Voitimme Kajaanin GGJ:n osakilpailun ja päätimme kehittää peliä myöhemmin lisää.



Kuvio 9 Kuvakaappaus alkuperäisestä GGJ:ssä kehitetystä Hopping Penguinista

Tiimillämme oli GGJ-tapahtuman jälkeen paljon muita opintoja, mutta jatkoimme silti projektin kehittämistä. Kehitimme peliä vajaa kaksi viikkoa ja osallistuimme Nokian kilpailuun, missä etsittiin hyviä suomalaisia? pelejä. Kilpailussa vaadittiin, että peli oli julkaistu, joten julkaisimme pelin Windows Phone 7:lle Kajak Games osuuskunnan kanssa. Kilpailua emme voittaneet, mutta peli sai yllätykseksemme odotettua enemmän latauksia (120 000), joten huomasimme pelissä olevan potentiaalia.

Kuvio 10. Kuvakaappaus julkaistusta Windows Phone 7 versiosta.

Jatkokehitimme vielä muutaman kuukauden lisäillen ominaisuuksia ja tehden peliin lisää sisältöä. Saimme pelin hyvälle mallille, mutta työtä oli kuitenkin vielä paljon, jotta pelin olisi voinut julkaista. Tässä vaiheessa päätimme, että opinnot saivat olla etusijalla ja projekti jäi tauolle.

3.2 Peliprojektin kaupallinen kehitys

Myöhemmin saimme opintomme vaiheeseen, jossa pystyimme paremmin keskittymään pelin kehittämiseksi oikeaksi kaupalliseksi sovellukseksi. Aloitimme kokonaan puhtaalta pöydältä, sillä aiempi versio oli kehitetty alkuperäisen prototyypin pohjalta, eikä lähdekoodi ollut kovinkaan käyttökelpoista jatkokehityksen kannalta. Tauon jälkeen näkökantamme oli muutenkin muuttunut ja halusimme kehittää pelistä selvästi laadukkaamman. Tässä vaiheessa vaihdoimme myös kehitysympäristön cocos2d-x:ään. Halusimme lähtökohtaisesti kehittää peliä kaikille mobiililaitteille, joten XNA:lla emme voineet jatkaa, sillä se on Microsoftin oma kehitysympäristö vain Windows-laitteille. Olimme käyneet paljon opintoja C++ kielestä ja sen käyttäminen houkutteli paljon. Cocos2d-x tarjosi ilmaisen ja hyvän vaihtoehdon mobiilipelien kehitykseen

C++ kielellä. Olimme kokeilleet cocos2d-x pelinkehitysympäristöä aiemmin ja tiesimme sen olevan toimiva ratkaisu. Siihen aikaan se oli yksi yleisimmistä mobiilipelien kehitysympärisöistä, joten se tuntui luonnolliselta vaihtoehdolta. (25)

Yksi tärkeä tavoite kehityksessä oli, että voisimme testailla ja kokeilla uusia asioita hyvin nopeasti ja helposti. Saimme ajatuksen kenttäeditorista, joka mahdollistaisi myös pelin pelaamisen samanaikaisesti kentän muokkauksen yhteydessä. Kenttäeditoreja on saatavilla ilmaiseksi verkosta, mutta ne ovat yleensä hyvin pelkistettyjä ja ne ovat huonosti muokattavissa. Kehitimme siis itse cocos2d-x:llä oman kenttäeditorin, jonka kanssa kehitimme pelimekaniikan alkuperäisen peliin kehitettyjen ideoiden pohjalta. Saimme nopeasti kasaan työkalun, jolla oli mahdollista luoda kenttiä ja pelata niitä samanaikaisesti. Lisäksi teimme kenttäeditoriin ominaisuuksia jotka vähensivät manuaalisen työn tarvetta. Esimerkiksi automaattinen kenttäelementin valinta ympäröivien elementtien perusteella. Käytännössä tämä mahdollisti tasohyppelypelin tasojen maalaamisen helposti hiirtä käyttäen. Nämä ominaisuudet nopeuttivat selvästi kenttien kehitystä, sillä kenttien prototypointi, ongelmien testaaminen ja uusien ominaisuuksien kokeileminen nopeutui huomattavasti.

Tässä vaiheessa saimme tiimiimme mukaan myös oikean graafikon, sillä aiemmin olimme koodauksen ohessa tehneet itse kaiken grafiikan. Graafikon mukaan tuleminen projektiin oli pelin kannalta erittäin merkittävää, sillä nykyään pelien grafiikka on erittäin tärkeää. Grafiikka on pelin näkyvä osa, minkä pelaaja näkee ensimmäisenä ja muodostaa ensivaikutelmansa sen perusteella. Jos ensivaikutelma on huono, voi koko peli jäädä pelaamatta.

3.3 AppCampus rahoitus

Saimme pelin kehityksen aikana tietoomme Microsoftin, Aalto-yliopiston ja Nokian AppCampus – yhteishankkeesta. Hankkeen tarkoitus on auttaa sovellusten kehittämisessä ja kääntämisessä Microsoftin Windows – järjestelmille. Hanke tarjoaa kehittäjille rahoitusta, koulutusta ja tukea kehityksen eri osa-alueille. Vastineeksi kehittäjät antavat kolmen kuukauden yksinoikeuden, jonka aikana sovellus voi olla saatavilla vain Windows – järjestelmille. Haimme mukaan ohjelmaan, koska rahoitus ilman merkittävää vastinetta olisi erittäin kannattavaa. Lisäksi tarkoituksenamme oli muutenkin testata peliä todellisessa tilanteessa ennen varsinaista Android ja iOS julkaisua. Ohjelmaan haku tapahtui Microsoftin järjestämissä paikallisissa tilaisuuksissa,

ja sellainen järjestettiin sopivasti Kajaanissa kehityksen aikana. Pääsimme mukaan hankkeeseen ja keskityimme täysin pelin Windows Phone kehittämiseen. Hankkeen aikana pelistä vaadittiin tarkat määritelmät ja tavoitteet mitkä toivat kehitykseemme lisää järjestelmällisyyttä.

AppCampus tarjosi lisäksi vielä AppCademy kurssin. Kurssi oli kahden viikon intensiivikurssi, joka sisälti mobiilisovellusten markkinointia, kehitysapua ja tukea ulkoiseen rahoituksen hakuun. Kurssilla oli mukana mobiilialan ammattilaisia ohjaamassa sekä luennoimassa mobiilikehittäjille tärkeistä seikoista. Ryhmästämme lähti kurssille Olli Lahtinen ja koimme sen koko projektimme kannalta hyödylliseksi. Vielä parempi olisi tietenkin ollut, mikäli olisimme päässeet kurssille ennen julkaisua, sillä silloin olisimme välttäneet monia virheitä joita teimme julkaisun yhteydessä.

3.4 Pelin julkaisu ja markkinointi

Hopping Penguin julkaistiin pitkän kehityksen jälkeen toukokuussa 2014. Pelin jatkokehitys kuitenkin vielä jatkui, sillä meillä oli visio kehittää pelistä vielä parempi ja lisätä siihen sisältöä.

Peliä julkaistaessa yksi tärkeimmistä asioista on markkinointi. Mobiilipelejä on kauppapaikoissa niin paljon, että ilman kunnollista markkinointia on vaikeaa saada pelilleen käyttäjiä. Windows Phone kaupassa on yli 300 000 sovellusta, joten on selvää että käyttäjien on vaikea löytää juuri sinun sovellustasi. (26) (27)

Androidin Google Playssa ja iOS:n AppStoressa on vielä moninkertaisesti sovelluksia.

Kun julkaisimme pelin, koitimme markkinoida peliä mahdollisimman paljon. Teimme pressmateriaaleja kuten kuvakaappauksia, pelivideota ja muuta grafiikkaa, joita artikkeleiden ja arvostelujen kirjoittamisessa voidaan käyttää.

Kirjoitimme foorumeille pelimme julkaisusta ja lähetimme lehdistötiedotteita useisiin kymmeniin Windows Phone-peleihin keskittyneisiin sivustoihin.

Lehdistötiedotteen lähettäminen osoittautui hyödylliseksi, sillä ensimmäiset merkittävät latausmäärät tulivat italialaisten sivustojen arvostelujen ansiosta.

Tärkeintä oli tässä vaiheessa kuitenkin se, että saisimme pelin suurimman Windows Phone peleihin keskittyneen sivuston WPCentralin arvosteltavaksi. Ensimmäisen lehdistötiedotteen

jälkeen ei asiasta kuulunut mitään, joten päätimme tehdä asialle jotakin. Olimme saaneet artikkeleita pienemmille sivustoille, kuten WMPowerUseriin ja Windowsgamersiin. WPCentraliin voi kuka tahansa lähettää juttuvinkkejä, joten keksimme lähettää salanimellä juttuvinkin, jossa ilmaistaan olevan WPCentralin sekä meidän pelimme fani, ja ihmetellään miksi uusi ja muualla huomioitu peli ei ole saanut näkyvyyttä WPCentralissa. Eipä aikaakaan, kun sivustolla julkaistiin positiivinen arvostelu ja arvostelun lopussa kiitettiin meidän juttuvinkin keksittyä italialaista pelifania Davide Bonaccorsoa.

Näiden lisäksi ostimme mobiilimainostusta useilta eri tarjoajilta kuten AdDuplexilta ja AdDealsilta. Lisäksi saimme AppCampukselta AdDuplexiin kupongin, joka oikeutti tuhannen dollarin edestä mainostukseen. Näiden avulla saimme paljon käyttäjiä lyhyessä ajassa, mikä on merkittävää kauppapaikkojen pelilistauksien sijoitusten kannalta. Listasijoitukset ovat merkittäviä siksi, että niiden avulla voi saada orgaanisia latauksia ilmaiseksi.

Julkaisuvaiheessa pelimme rahoitusmalli oli hyvin yksinkertainen. Pelissämme oli bannerimainokset, jotka aktivoituivat kun pelaaja oli pelannut riittävän pitkällä. Sen lisäksi pelaaja pystyi maksamaan pelin sisällä pienen summan, jolla sai mainokset pois näkyvistä. Jälkeenpäin ajateltuna peli olisi kannattanut julkaista ilman mitään rahoitusmallia, sillä mainostulot ovat merkittäviä vasta päivittäisten käyttäjien lukumäärän kasvettua riittävän suureksi. Microsoft ja AdDuplex suosittelevat rahoitusmallin lisäämisen peliin vasta myöhemmin. Tämän saimme tietoomme vasta kauan pelin julkaisun jälkeen AppCademyn kautta. Idea siinä perustuu siihen, että pelin antaminen ilmaiseksi on tehokas markkinointikeino. Käyttäjät ovat todennäköisemmin tyytyväisempiä ja suosittelevat sovellusta tuntemilleen henkilöille. (29)

3.5 Pelin kääntäminen Unitylle

Jatkoimme julkaisun jälkeen pelin jatkokehitystä pitkään. Korjasimme pelin ongelmia, lisäsimme siihen uusia ominaisuuksia sekä pelattavaa sisältöä. Jatkokehitys ei ollut ongelmatonta, joten lopulta ratkaisuna oli kehitysympäristön vaihtaminen Unityyn.

Päätös siirtää pelin jatkokehitys Unitylle ei tapahtunut ilman syytä. Cocos2d-x:lle kehitetty peli oli julkaistu ja toiminnassa sekä sillä oli jo yli 100 000 latausta. Cocos2d-x sovelluskehyksessä

oli kuitenkin ongelmia Windows Phone 8 alustalla ja halusimme korjata ongelmat. Ongelmia oli suorituskyvyn kanssa HD-tarkkuuksisilla puhelinmalleilla, sekä muutamia erikoisia näytöntarkkuusongelmia tietyillä puhelinmalleilla. Selvitimme asiaa viikkokausia, kunnes lopulta tultiin päätepisteeseen, että ongelmat eivät olleet ratkaistavissa ilman valtavaa työmäärää. Ongelmia oli ollut muillakin kehittäjillä, ja ongelma johtui Cocos2d-x:n Angle Project kirjastosta, jonka tehtävänä on kääntää cocos2d-x:n OpenGL grafiikkakomennot Windows Phone:n käyttämään Direct X järjestelmään. Jotta näytöntarkkuus- ja suorituskykyongelmat olisi saatu ratkaistua, olisi meidän pitänyt kirjoittaa cocos2d-x:n renderöintimoduuli kokonaan uudelleen natiivia DirectX:ää käyttäen. Tämä olisi voinut olla jopa isompi työ kuin koko pelin ohjelmointi, joten päätin tässä vaiheessa etsiä ratkaisua laatikon ulkopuolelta. Tätä työtä tehdessä yhteisö on korjannut ongelman tekemällä natiivin DirectX renderöintimoduulin cocos2d-x:ään, mutta sitä ei ole vielä lisätty viralliseen julkaisuun. (30)

Olimme käyttäneet Unityä pienemmissä projekteissa paljon, joten meillä oli siitä paljon kokemusta. Päätimme kääntää pelin keskeisimmät mekaniikat Unitylle, jotta saisimme alustavan arvion koko kääntämisen kannattavuudesta ja kestosta. Saimme pelin pelattavaan muotoon jo muutamassa päivässä, joten teimme päätöksen kääntää koko peli Unitylle.

Ongelmat yksinään eivät olleet syynä Unityyn siirtymiseen, mutta ne toivat ilmi kehityksen hitauden. Unityn C# mahdollistaisi nopeamman jatkokehityksen sekä helpottaisi pelin tuomista muille alustoille.

Peliimme suunnitellut uudet ominaisuudet olisivat helpompi ja nopeampi toteuttaa Unityllä. Uskomme, että lopulta säästimme paljon aikaa ja vaivaa siirtymällä Unityyn.

3.6 Peliprojektin hallinta

Projekti on hanke, jonka on tarkoitus saavuttaa tietty päämäärä. Projektinhallinta on työvoiman ja resurssien hallintaa tavalla, jolla projekti voi valmistua suunniteltuna aikataulun ja budjetin mukaisesti. Resursseihin kuuluvat esimerkiksi tila, palkat ja raaka-aineet. Projektinhallinnassa huomioidaan myös laatu ja projektiin liittyvät riskit. (31)

Pelien kehityksessä käytetään nykyisin paljon Agile-menetelmää, joka tarkoittaa ketterää ohjelmistokehitystä. Agile on kehitetty projekteja varten, joissa on vaikea tarkasti määritellä projektin tavoitteita ja aika-arvioita. Agilen idea on jakaa projekti nopeisiin iteratiivisiin prosesseihin, joissa kehitetään projektin osatavoitteita lyhyiden ajanjaksojen ajan. (32) (33) (34)

Pelin kehittäminen on usein juuri edellämainitun kaltainen iteratiivinen prosessi, johon Agile sopii hyvin. Pelien kehittämisessä iterointi on tärkeää, koska sillä voidaan saada aikaan parempia pelejä. Iteroinnissa kehitettävästä osa-alueesta hankitaan palautetta ja sitä analysoidaan jonka jälkeen siitä luodaan mahdollisesti parempi uusi versio. Tätä voidaan toistaa niin monta kertaa kun koetaan tarpeelliseksi ja yleensä lopullinen tuote paranee iteraatiomäärän kasvaessa. (35)

Projektinhallinta Hopping Penguinin kehityksessä oli alkeellista. Aiempaa kokemusta projektinhallinnasta oli tiimillämme Scrumista ja Pivotal Trackeristä aiempien peliprojektien kanssa. Hopping Penguinin kehityksessä tiimiimme ei missään vaiheessa kuulunut tuottajaa, joka olisi voinut vastata projektinhallinnasta.

Ennen varsinaista kaupallista kehitystä projektinhallinta oli lähinnä täysin suullisen kommunikaation varassa, eikä projektilla ollut tarkkoja määritelmiä. Tämä ei kuitenkaan pienessä mittakaavassa aiheuttanut merkittäviä haittoja, sillä kokeilimme vielä eri pelimekaniikkojen toimivuutta spontaanisti. Kun myöhemmin aloitimme varsinaisen kaupallisen kehityksen tavoitteenamme kaupallinen sovellus, oli projektin määritelmätkin jo hieman tarkempia. Varsinaista projektinhallintaa ei kuitenkaan vieläkään ollut. Tavoittet olivat ylhäällä tietokoneiden muistilappu-sovelluksissa ja projektinhallinta oli melko olematonta. Myöhemmin kehitystä saimme ammattikorkeakoulun sponsoroimana käyttöömme Pivotal Tracker websovelluksen, joka helpottaa projektinhallinnassa. Pivotal Tracker on helppokäyttöinen projektinhallintatyökalu Agile-menetelmän soveltamiseen. Saimme myös konsultointia Jukka Jylängiltä, jonka varsinaista osaamista oli ohjelmointi, mutta hän hallitsi erinomaisesti myös projektinhallintaa. Saimme häneltä hyviä vinkkejä ja neuvoja projektinhallintaan ja sen osa-alueisiin, kuten tehtävien määrittelyyn ja arviointiin. Ongelmana oli kuitenkin edelleen se, että tiimimme pääohjelmoijan aika ei hänen omasta mielestään riittänyt projektinhallinnan hallintaan, ja sen hallinta oli heikkoa.

4 HOPPING PENGUIN PELIN TEKNINEN TOTETUTUS

Hopping Penguinin kehityksessä käytettiin lukuisia valmiita teknisiä ratkaisuja sekä kehitettiin omia järjestelmiä, joita voi hyödyntää myös muissa projekteissa. Valmiita ratkaisuja käyttämällä säästää reilusti aikaa, sillä kehittämisen lisäksi ratkaisut ovat myös valmiiksi testattuja.

4.1 Kenttäeditori

Kehitimme peliä varten oman kenttäeditorin, sillä emme halunneet rajoittaa itseämme valmiiden kenttäeditorien ominaisuuksiin. Halusimme kenttäeditorin, joka mahdollistaisi pelin pelaamisen samanaikaisesti kentän muokkauksen yhteydessä. Kehitimme editorin coco2d-x:llä Saimme nopeasti kasaan työkalun, jolla oli mahdollista luoda kenttiä ja pelata niitä samanaikaisesti. Lisäksi teimme kenttäeditoriin ominaisuuksia jotka vähensivät manuaalisen työn tarvetta. Esimerkiksi automaattinen kenttäelementin valinta ympäröivien elementtien perusteella. Käytännössä tämä mahdollisti tasohyppelypelin tasojen maalaamisen helposti hiirtä käyttäen. Nämä ominaisuudet nopeuttivat selvästi kenttien kehitystä, sillä kenttien prototypointi, ongelmien testaaminen ja uusien ominaisuuksien kokeileminen nopeutui huomattavasti.

Kenttäeditorin ominaisuuksia:

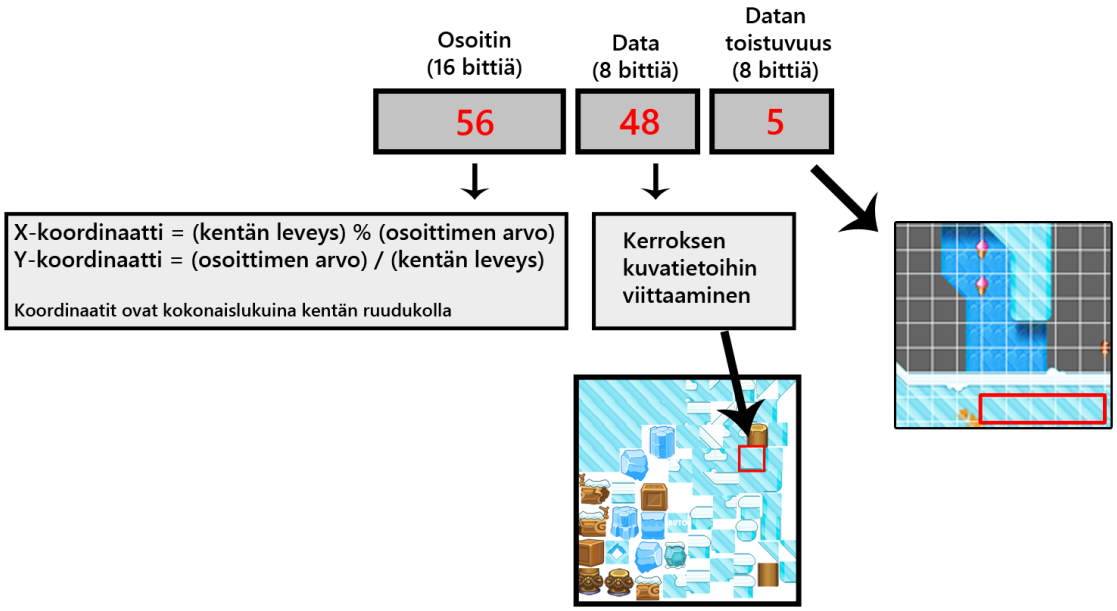
- kentän maalaaminen nopeasti hiirellä ilman tiiligrafiikan valintaa
- manuaalinen tiiligrafiikan valinta
- peliobjektien asettelu kentälle
- alueellinen valinta, siirtäminen, kopioiminen ja poistaminen
- pelin pelaaminen muokkauksen yhteydessä
- tallentaminen ja lataaminen omaan tiedostoformaattiin

Kehitimme kentillemme oman tiedostoformaatin, sillä se oli helpoin ratkaisu omasta mielestämme. Halusimme kentistä mahdollisimman pieniä levytilan suhteen, jotta lopullisen pelin koko ei tulisi olemaan liian iso. Näinpä itse kenttätieto on binaarikoodattuna ja pakattuna mahdollisimman tiiviiksi, mutta tiedoston kehys on xml muodossa helpon luettavuuden vuoksi. Kenttätiedosto määrittelee useita kenttään liittyviä ominaisuuksia ja kaiken kentässä käytettävän pelitiedon. Xml määritelmän vuoksi kentän kerrosten binaaridata piti koodata base64 muotoon, sillä xml sallii vain tekstimuotoista tietoa. Base64 käyttää tiedon esittämiseen vain Xml-formaatissa sallittuja merkkejä joten sitä voidaan sisällyttää xml tiedostoon, eikä se aiheuta lukuongelmia. (36)

```
<level>
    <starvalues one="10" two="18" three="26" total="9"/>
    <size x="33" y="23"/>
    <tilesize w="60" h="60"/>
    <layers>
        <layer name="BACKGROUND" tileset="background1" double="0">
            <tiles data="ywADAuwABQINAQUFhAEBAYUBAgKHAQEDpQEBAaYBAgKoAQEDxgE
        </layer>
        <layer name="GROUND" tileset="ground1" double="0">
            <tiles data="pgABBacAEQG4AAEGxwABB8gAAwLLAAESzAABCc0AARbOAAsC2QA
        </layer>
        <layer name="OBJECT" tileset="object1" double="1">
            <tiles data="qgQBQVsFASVzBQE1gwUBOPoFAQH9BQFPEgYBOSAGATUoBgE38gY
        </layer>
        <layer name="FOREGROUND" tileset="foreground1" double="0">
            <tiles data=""/>
        </layer>
    </layers>
    <tutorials/>
</level>
```

Kuvio 11. Esimerkki kenttätiedostosta ja siitä ladatusta kentästä.

Binääritieto koostuu kenttätiedon lohkoista, jotka ovat 32 bittiä pitkiä. 32-bittinen lohko koostuu 16-bittisestä osoitin, joka määrittää lohkon sisältävän kenttätiedon paikan kentän koordinaateissa. Seuraavat 8 bittiä ilmaisevat kerroksen objektitunnuksen, ja viimeiset 8 bittiä ilmoittavat lukumäärän, kuinka monta objektia tulee peräkkäin.



Kuvio 12. Kuvaaja kenttälohkon tiedon lukemisesta

Kun siirryimme käyttämään Unityä, käytimme silti vielä vanhaa editoria. Jätimme kenttäeditorin kääntämättä koska ei ollut järkevää kääntää toimivaa editoria uuteen järjestelmään. Kentät saatiin Unityyn kirjoittamalla yksinkertainen editoriskripti, joka latasi kenttätiedoston Unity-näkymään.

4.2 Pelin sisäiset järjestelmät

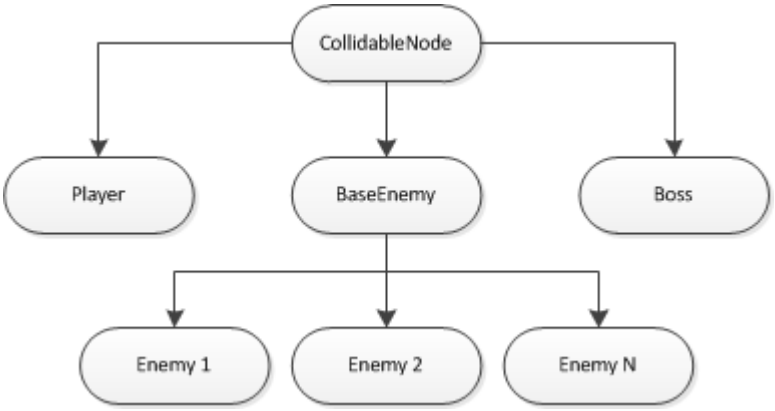
Hopping Penguin koostuu näkymistä. Näkymä voi olla esimerkiksi päävalikko, lopputekstiruutu tai yksittäinen kenttä. Jokainen kenttä on siis oma näkymänsä, jossa on staattisesti määritelty kentän geometria ja törmäysdata, kerättävät objektit, viholliset sekä erilaiset interaktiiviset objektit, kuten tallennuspiste (checkpoint) tai maali. Staattisella geometrialla ja törmäysdatalla tarkoitetaan sitä, että se ei ole muokattavissa pelin ajon aikana.

Koska kenttädataa ei ladata ajon aikana, se on nopeampi ladata koska Unity osaa tehokkaasti optimoida näkymien lataamisen. Staattisesta kenttädatasta on hyöty myös suunnitelijalle, sillä kentän objekteja voi muokata Unityn kehitysympäristössä ja lopputuloksen näkee välittömästi graafisessa käyttöliittymässä, joka taas nopeuttaa kehitystä.

Erikoiskentät ladataan poikkeuksellisesti ajon aikana, mutta niiden tekeminen ja muokkaaminen tapahtuu samalla tavalla kuin muissakin kentissä. Erikoiskentät ovat kenttiä, joita voi ladata internetistä erikseen, joten niitä ei ole liitettynä alkuperäiseen peliin ja tästä syystä erikoiskentät tulee ladata ajon aikana.

Hopping Penguin pyrittiin ohjelmoimaan siten, että kenttään ei luoda tai poisteta objekteja ajon aikana, sillä se on hidasta ja näin ollen vaikuttaa suorituskykyyn negatiivisesti. Esimerkiksi vihollisen kuoltua objektia ei poisteta, vaan se deaktivoidaan. Deaktivoiminen tarkoittaa sitä, että objekti on vielä ohjelman muistissa mutta sitä ei päivitetä ennen kuin se aktivoidaan uudestaan. Hopping Penguin kuluttaa verrattavan vähän muistia, joten toimintatavasta ei koidu ongelmia.

Hopping Penguinissa yleisin luokka on CollidableNode. CollidableNode hoitaa liikkumisen pelimaailmassa sekä törmäystarkistuksen kentän kanssa. CollidableNode-luokasta periytetään mm. pelaajaluokka, pomoluokka sekä vihollisluokat, siis kaikki objektit joiden pitää törmätä kentän kanssa.



Kuvio 13. Kaavio CollidableNode-luokasta ja siitä perittävistä luokista.

Liikkumisen ja kentän törmäystarkistuksen lisäksi tärkeitä luokkia ovat VisibilityManager sekä RestoreManager sekä. VisibilityManager on optimointiin tarkoitettu luokka, jolla rajoitetaan aktiivisien objektien määrää. Ei ole mitään hyötyä piirtää tai päivittää sellaista objektia, joka ei ole ruudulla näkyvissä. RestoreManager on tarkoitettu peliobjektien hallintaan. Koska RestoreManager sisältää viittaukset kaikkiin kentässä oleviin vihollisiin, sitä käytetään myös pelaajan ja vihollisen väliseen törmäystarkistukseen. RestoreManager hoitaa myös peliobjektien palautuksen kuoleman jälkeen, jolloin pelaaja siirtyy edelliselle tallenuspisteelle (checkpoint) ja tallennuspisteen jälkeiset objektit palautetaan alkutilaan.

Hopping Penguin sisältää myös graafisia järjestelmiä, joiden tarkoitus on tuottaa tyylikkäitä efektejä mahdollisimman kevyesti. Järjestelmiä ovat mm. partikkelijärjestelmät sekä järjestelmä monitasoisen taustan piirtämiseen (parallax).

CloudSync on järjestelmä, jonka avulla pelin eteneminen voidaan ladata tai tallentaa palvelimelle (pilveen). Järjestelmä on suunniteltu siten, että tieto menee varmasti palvelimelle: tallennettava tieto pysyy välimuistissa niin kauan, kunnes se on varmistetusti tallennettu palvelimelle. Järjestelmä koostuu pelin sisäisestä ohjelmakoodista (client) sekä palvelimesta (server).

4.3 Kehityksessä käytettävät ohjelmistot

Pelimme cocos2d-x version kehittämiseen käytimme Microsoftin Visual Studio 2010 C++ kehitysympäristöä pelin ohjelmointiin. Unity version kanssa käytimme myös Visual Studiota, mutta C# versiota. Unity versiota kehittäessämme halusimme keskittyä mahdollisimman paljon itse pelin kehittämiseen, joten ostimme Unity Asset Storesta lisenssejä peliin tarvittaviin teknologioihin. Pelin grafiikka on tehty lähinnä PhotoShop CS5-ohjelmistolla. Pelin ensimmäisessä versiossa oli apuna käytetty myös Autodesk 3DS MAX- mallinnusohjelmaa sulavien animaation aikaansaamiseksi. Lopulliseen versioon teimme kuitenkin myös animaatiot käsin PhotoShopilla.

4.3.1 2D Toolkit

2D Toolkit on järjestelmä, joka tuo monipuoliset 2D työkalut Unityyn. Unity on tarkoitettu lähinnä 3D-pelien kehittämiseen, ja aiemmin Unityssä ei ollut 2D työkaluja lainkaan. Nykyään Unity sisältää 2D-työkalut, mutta mielestämme 2D toolkit tarjoaa silti huomattavasti monipuolisemmat ja tehokkaat välineet 2D pelien kehittämiseen. 2D toolkit tekee 2D-pelien kehittämisestä Unityllä helppoa ja nopeaa. Se sisältää kuvien pakkaustyökalun, joka korvasi TexturePacker työkalun tarpeen ja nopeutti kehitystä. Lisäksi siinä on kaikki tarvittava teknologia kuvien, animaatioiden ja muiden 2D-peleissä tarpeellisten ominaisuuksien luomiseen.

4.3.2 Unibill

Unibill on Unityn lisäosa, jonka avulla pelien sisäiset maksut voi hoitaa ilman alustakohtaista koodia. Normaalisti pelien sisäiset maksut joutuu ohjelmoimaan alustakohtaisesti, sillä jokaisella laitevalmistajalla on eri API niiden käyttöön. Unibill abstraktoi tämän yhdeksi universaaliksi järjestelmäksi eikä kehittäjien tarvitse näin kuluttaa aikaa maksujärjestelmien ohjelmointiin. Cocos2d-x versiossa käytimme itse ohjelmoitua järjestelmää, jonka tekeminen oli varsin iso työ. Valmistajat vaativat, että maksujärjestelmä on turvallinen ja luotettava. Windows Phonella tämä tuli ilmi siinä vaiheessa kun Microsoft oli valmis mainostamaan peliämme Windows Phonen kauppapaikassa. Saimme sähköpostin, jossa vaadittiin meitä korjaamaan maksujärjestelmäämme, sillä siinä oli aukko. Kun siirryimme Unity-kehittämiseen ostimme heti Unibill

lisenssin, jotta säästyimme ylimääräiseltä työltä ja pystyimme keskittyä enemmän käyttäjille suunnatun sisällön luomiseen. (37)

4.3.3 TextMesh Pro

TextMesh Pro on lisäosa, joka parantaa peleissä käytettävän tekstin suorituskykyä ja kuvanlaatua. Lisäksi se poistaa manuaalisia vaiheita tekstin lisäämisestä peliin, eli siis nopeuttaa kehittämistä. Monissa pelimoottoreissa tyylitellyn ja tehosteita sisältävän tekstin piirtämiseen on joutunut käyttämään bittikarttafontteja. Niiden lisääminen peliin yleensä vaatii erillisen ohjelmiston käyttöä. Lisäksi fontteja on hyvä luoda useita jokaiselle käytettävälle fonttikoolle, koska fonttien koon muuttaminen ohjelmallisesti heikentää piirrettävän tekstin laatua. Text Mesh Pro on distance field- renderöinnin toteutus Unitylle ja se perustuu Valven soveltamaan renderöintitekniikkaan. Text Mesh Pro:n avulla voidaan piirtää näytölle tarkkaa tekstiä koosta riippumatta, näin ei tarvitse joka fonttikoolle luoda omaa tekstuuria ja voidaan säästä käytettävää näyttömuistia ja laatu säilyy erinomaisena. Lisäksi Text Mesh Pro mahdollistaa tekstitehosteiden lisäämisen tekstiin. Tekstin väriä voi muuttaa, sille voi lisätä reunaviivat, varjostuksen tai muita tehosteita.

Kuvio 14. Esimerkki kuvanlaadusta eri renderöintitekniikoilla. Oikean puoleisin on piirretty signed distance field tekniikkalla.

4.3.4 Optimointi ja hyötyohjelmat

Käytimme useita työkaluja kehityksen nopeuttamiseksi ja pelin toiminnan parantamiseksi. Pelissä käytettävien kuvatiedostojen kokoamiseen yhteen tekstuuriin käytimme TexturePacker ja ShoeBox ohjelmia cocos2d-x version kehityksen aikana. Lisäksi optimoimme png-kuvatiedostot pngquant-ohjelmalla. Pngquant optimoi png-kuvatiedostoja poistamalla tarpeettomat meta-datat sekä rajoittamalla väripalettia. Väripaletin rajoittaminen heikentää luonnollisesti kuvanlaatua, mutta pngquantin käyttämä ditherointi saa monissa tapauksissa alkuperäisen ja optimoidun näyttämään silmämääräisesti identtisiltä. Optimoinnista saatu hyöty on merkittävä, sillä kuvatiedostojen viemä tallennustila voi olla optimoinnin jälkeen jopa 10% alkuperäisestä koosta. Meidän tapauksessa säästimme useamman megatavun, joka on mobiilipelien kannalta melko merkittävää. Optimointi on merkittävää etenkin tapauksissa, joissa pelin julkaisuvaiheen viemä levytila on yli valmistajan salliman rajan 3G-verkon kautta ladattaessa. Tällöin peli voidaan ladata vain Wifi-yhteyden välityksellä ja se rajoittaa pelaajien mahdollisuuksia kokeilla peliä.

4.4 Analytiikka

Nykyään hyvät mobiiliverkkoyhteydet mahdollistavat anonyymin käyttäjädatan keräämistä mobiilipeleistä. Vuonna 2009 lähes kolmasosa matkapuhelinten käyttäjistä käytti puhelintansa internetin käyttämiseen. Matkapuhelimia oli BBC:n mukaan vuonna 2010 yli viisi miljardia, joten voidaan sanoa että internetiä käyttäviä käyttäjiä on nykyään huomattava määrä. (39)

Pelianalytiikka toimii samoilla periaatteilla kuin web-sivujen kanssa käytettävä web-analytiikka. Analytiikan avulla voidaan saada tärkeää tietoa pelin ja pelaajien toiminnasta sekä tietoja, joilla voidaan kehittää liiketoimintaa. Jotta pelien tuottavuudesta ja toimivuudesta voidaan tehdä johtopäätöksiä, niitä on voitava mitata, ja mittaustuloksia analysoitava. Analytiikan tarkoitus on mahdollistaa tuotteen tai palvelun kehittäminen.

Jotta sovelluksen sisällä tapahtuneita tietoja voidaan saada tarkasteltavaksi on käyttäjän oltava yhteydessä internettiin jossakin käytön vaiheessa.

4.4.1 Analytiikkapalvelut

Mobiilipeleille on tarjolla useita ilmaisia ja maksullisia analytiikkajärjestelmiä, jotka yleensä ovat helppo lisätä peliin palveluntarjoajan sovelluskehitysliitännäisen avulla. Yksi suosituimpia on Flurry, jota käyttää yli 500 000 mobiilisovellusta. Googlen lähinnä web-analytiikkaan keskittynyt Google Analytics on saatavilla myös mobiilisovelluksiin. Lisäksi tässä opinnäytteessä käsittelemme GameAnalytics palvelua, joka on erityisesti pelien analytiikkaan ja liiketoiminnan seurantaan tarkoitettu palvelu. (40)

4.4.2 Käyttömahdollisuudet

Analytiikoille on useita käyttömahdollisuuksia, ja niitä voidaan hyödyntää peleissä monin tavoin. Tietoa kerätään ja tutkitaan yleensä tarpeen mukaan.

Pelit ovat monimutkaisia järjestelmiä, jotka nykyisin toimivat useilla eri laitteilla. Analytiikkapalvelun avulla kehittäjät voivat saada reaaliaikaista tietoa pelin suorituskyvystä ja mahdollisista virheistä.

Pelin menestymisen kannalta on tärkeää, että peli toimii hyvin kaikilla laitteilla, millä peliä on pelattavissa. Syy tähän on se, että käyttäjät antavat käyttäjäpalautetta ja arvosteluja herkemmin kun ongelmia sattuu kohdalle, kuin mitä arvosteluja annetaan hyvästä pelistä. Etenkin pelin elinkaaren alkuvaiheessa tällä on ratkaiseva merkitys pelin tulevaisuuden kannalta.

Pelin toiminnan kannalta kriittisten ongelmien seurantaan emme käyttäneet cocos2d-x version kanssa mitään. Valmistajat tarjoavat oman hallintapaneelin kautta päivittäisten kaatumisten lukumäärän, sekä kaatumisiin johtuneet syyt teknisen raportin muodossa. Tämä ei kuitenkaan ole reaaliaikaista ja helppolukuista. Crittercism, BugSense ja jopa GameAnalytics tarjoavat reaaliaikaista virheiden seurantaa. Tällöin kehittäjät voivat helposti selvittää yleisimmät virheet ja niihin johtaneet syyt. Yksittäisiä sekä vähäisiä määriä tapahtuneita virheitä ei välttämättä kannata korjata, sillä korjausten kustannukset olisivat todennäköisesti saatuja hyötyjä suuremmat.

Teknisten käyttömahdollisuuksien lisäksi analytiikalla on paikkansa pelin sisällön analysoimisen. Analytiikkapalveluiden käyttö mahdollistaa kehittäjien selvittää pelin käytettävyyttä ja tarkastella esimerkiksi AB-testauksen toimivuutta.

Hopping Penguinia kehittäessämme halusimme selvittää kenttäsuunnittelumme ongelmakohtia, ja etenkin paikkoja missä pelaajat kuolivat tai epäonnistuivat eniten. Keräsimme talteen mahdollisimman paljon tietoa pelaajan pelaamisesta jotta voimme sen perusteella analysoida missä pelaajilla on eniten ongelmia pelin kanssa. Tärkeimmistä tiedoista, kuten paikoista, missä pelaajat kuolivat pelissä, teimme erityiset järjestelmät, jotta voimme näyttää tiedon helposti tulkittavassa kontekstissa. Esimerkkinä tästä seuraava kuva.

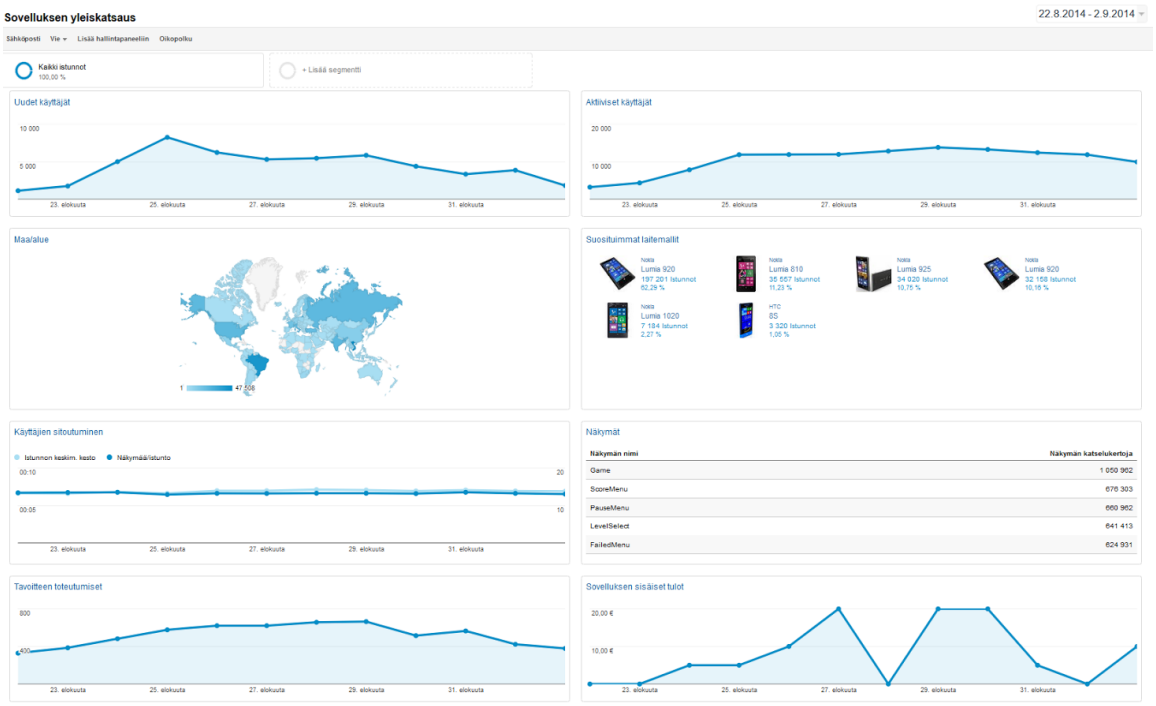
Kuvio 15. Kuvassa näkyvät punaiset alueet ovat paikkoja, joissa pelaajat ovat kuolleet. Mitä suurempi alue on, sitä enemmän kuolemia samassa kohdassa on.

Kaikki edeltävät toimet liittyvät myös pelin tuottavuuteen, sillä parempi tuote yleensä johtaa parempiin tuloksiin myös taloudellisella puolella. Analyytikoiden avulla tuottavuutta voidaan

kuitenkin selvittää tarkemmin esimerkiksi tarkastelemalla pelaajien ostoon johtaneita tapahtumia. Analytiikoiden tulkinta on yleensä sovelluskohtaista, eikä yleisiä johtopäätöksiä voida tehdä. Siksi yleistä onkin käyttää A/B-testausta eri vaihtoehtoiden paremmuuden mittaamiseen. A/B-testaus on yksinkertaisuudessan kahden eri toiminnon testaamista seuraamalla KPI-arvoja. (Key performance indicator) KPI-arvot ovat yleensä myös sovelluskohtaisia, mutta yleisiä KPI:ta ovat pelin käyttö, käyttäjän elinkaaren arvo, retentio, session pituus, keskimääräinen tulo maksavalta asiakkaalta, käyttäjien hankinta. (41)

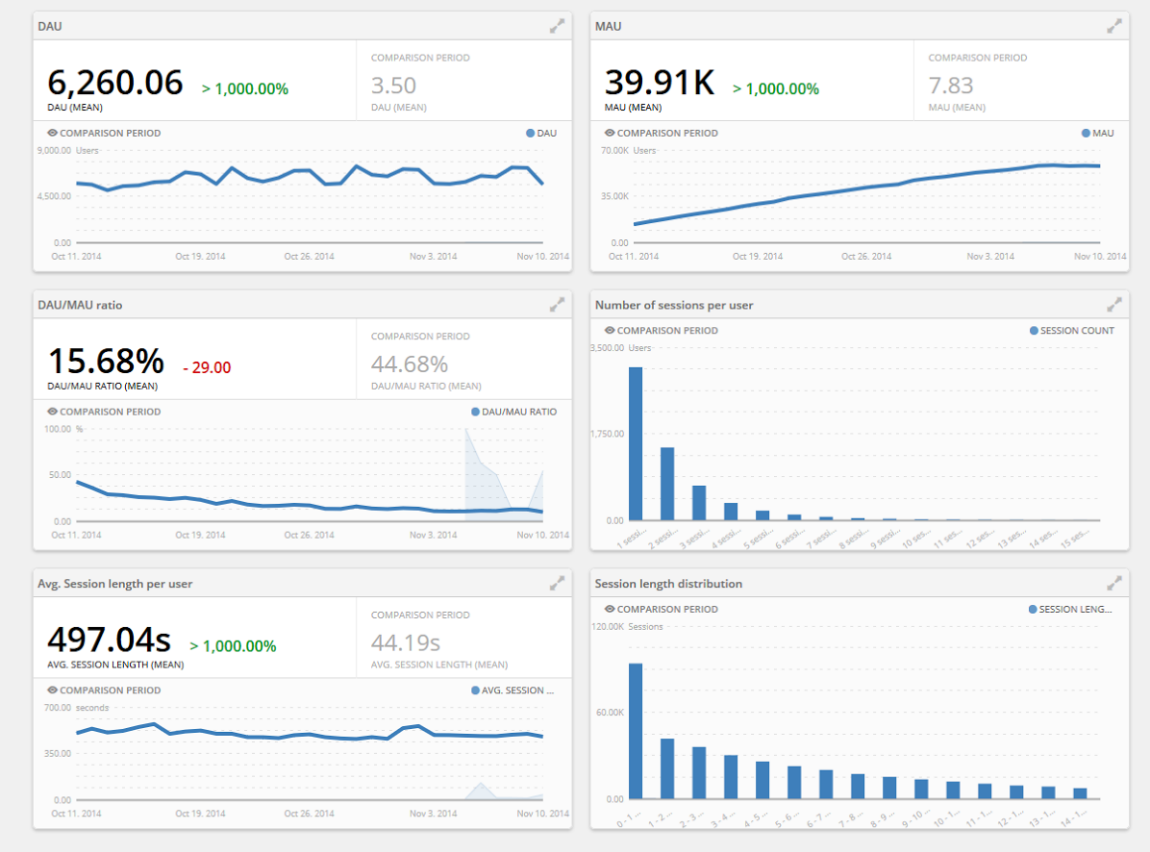
4.4.3 Analytiikat Hopping Penguin –pelin kehityksessä

Käytimme cocos2d-x-versiossa Google Analyticsiä, koska Flurry:n silloinen SDK ei toiminut odotetusti. Osa tiedoista jäi testiemme perusteella matkalle, joten emme luottaneet siihen riittävästi. Game Analytics jäi poissa laskuista sen vuoksi, että se oli uusi tulokas ja lähinnä Unityyn keskittynyt. Sen käyttö olisi vaatinut REST-API:n käyttöä, mikä olisi aiheuttanut paljon ylimääräistä työtä. Google Analytics oli toimiva ratkaisu, mutta sen käyttöliittymä vaikutti olevan enemmän web- ja sovellusanalytiikkaa varten. Lisäksi tiedon saamisessa pois järjestelmästä oli rajoituksia, jotka hidastivat tiedon analysoimista.



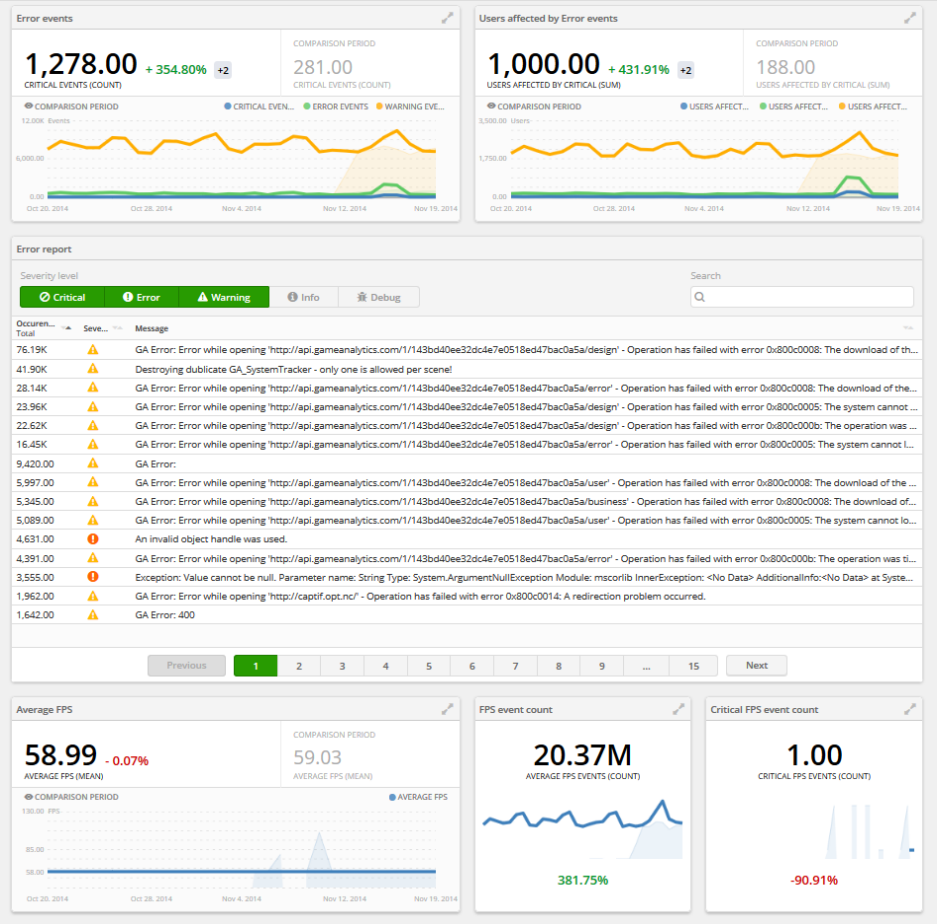
Kuvio 16. Google Analyticsin sovelluksen yleisnäkymä.

Kun siirryimme Unity-kehittämiseen halusimme analytiikkapalvelun, joka toimisi kaikilla alustoilla ilman erityistä alustariippuvaa ohjelmointia, joten päädyimme käyttämään Game Analyticsia. Se oli meidän selvittämistämme vaihtoehdoista ainoa, jolla oli kaikkia mobiilialustoja tukeva Unity liitännäinen. Game Analytics on erityisesti peleille tarkoitettu ilmainen analytiikkapalvelu. Sen käyttäminen Unityn kanssa on erittäin helppoa, suurin osa toiminnasta on automaattista. Tietojen saamiseksi ei tarvitse kuin asentaa liittännäinen ja lisätä seurantaobjekti ensimmäiseen pelin näkymään. Game Analyticsin hallintapaneelit ovat erittäin helppolukuisia ja kätevästi muokattavia. Hallintapaneeli näyttää vakiona pelien menestymisen kannalta tärkeitä mittareita kuten pelaajien sitoutumisen, pelin liiketoiminnan tapahtumat sekä pelin suorituskykyyn liittyviä tietoja. Kuviossa 17 Game Analyticsin sitoutumisnäkymä, josta voidaan nähdä pelaajien sitoutumiseen liittyviä tilastotietoja.



Kuvio 17. Kuvakaappaus Game Analyticsin sitoutumisnäkymästä.

Kokeilimme Unity-version kanssa myös Crittercism-palvelua. Crittercism on palvelu, joka antaa erittäin seikkaperäistä tietoa mobiilisovelluksen suoritusongelmista, ohjelmavirheistä ja suorituskyvystä. Sen avulla kehittäjät voivat nopeasti selvittää yleisimmät sovelluksen ongelmat ja korjata ongelmat tärkeysjärjestyksessä. Tieto on erittäin kattavaa, joten kehittäjät voivat yleensä selvittää ongelman syyn hallintapaneelin kautta jonka jälkeen itse korjaaminen on nopeampaa. Perinteisesti tieto ongelmista saadaan käyttäjä- tai testipalautteen perusteella, mikä vaatii ongelman syyn selvittämisen ennen ongelman korjausta. Crittercism oli palveluna erinomainen, mutta lopetimme sen käyttämisen ilmaisversion rajoitteiden vuoksi. Pelimme oli niin suosittu, että ilmaisversion kuukausittainen käyttäjäraja ylittyi ja jatkaaksemme palvelun käyttöä olisi pitänyt hankkia maksullinen versio. Maksullinen versio oli niin kallis, että sen hankkimista emme voineet edes harkita. Lopulta päädyimme käyttämään Game Analyticsin sisältämää virheseurantaa. Se ei ole niin seikkaperäinen kuin Crittercism, mutta ilmaiseksi ja rajoittamattomaksi palveluksi se on meille riittävä.



Kuvio 18. Game Analyticsin laatunäkymästä

4.5 Palvelinjärjestelmä Hopping Penguinissa

Hopping Penguinin palvelinta käytetään paljon ja se suorittaa tärkeitä tehtäviä peliin liittyen. Esimerkiksi pelin sisäisten ostosten toteuttamiseksi vaaditaan palvelin, jossa käyttäjän tekemät ostokset pysyy tallessa.

Palvelin pyörii Google App Engine -alustan päällä, joka mahdollistaa helpon skaalauksen ja siten samaa palvelinta voidaan käyttää myös muissa projekteissa Hopping Penguinin käyttökokemukseen vaikuttamatta.

4.5.1 UDP

UDP-protokollan avulla voidaan lähettää dataa kahden verkkoon kytketyn laitteen välillä ilman, että yhteyttä tarvitsee erikseen muodostaa. UDP on täten yhteydetön. Kun UDP-datapaketti lähetetään, ei siitä sen jälkeen pidetä enää mitään tietoa tallessa lähdelaitteessa. Tästä syystä myös suurien asiakasjoukkojen palveleminen onnistuu UDP:n avulla helposti, sillä yhteyttä ei tarvitse pitää auki jokaisen asiakkaan kanssa.

Koska UDP-protokollaa käytettäessä ei tarvitse muodostaa erillistä yhteyttä, eikä pakettien saapumista kohdelaitteelle varmisteta, se on erittäin nopea yhteysmuoto. Tästä syystä UDP:tä käytetään usein esimerkiksi reaaliaikaisissa moninpeleissä sekä videon suoratoistopalveluissa (streaming). (42)

4.5.2 TCP ja HTTP

TCP-protokolla on UDP:ta varmempi, mutta useissa tapauksissa hitaampi vaihtoehto. Protokolla pitää huolen siitä, että paketit saapuvat kohdelaitteelle oikeassa järjestyksessä, sekä mahdollisesti lähettää paketin uudestaan jos se ei koskaan saavuta kohdelaitetta. TCP vaatii myös

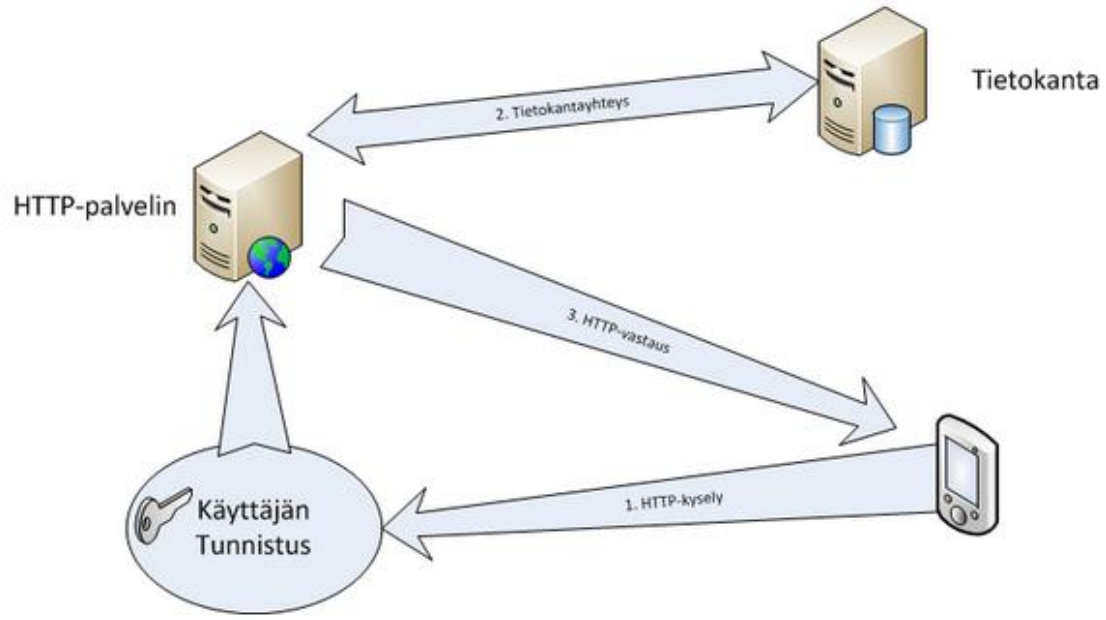
aktiivisen yhteyden laitteiden välille. Edellämainittujen seikkojen ansiosta TCP:llä saadaan huomattavasti korkeampi varmuus datan vastaanottamiselle oikeassa muodossa.

HTTP-protokolla on TCP-protokollan (yhteyden) päälle rakennettu järjestelmä, joka on tarkoitettu web-selainten ja -palvelimien väliseen kommunikointiin. Käyttäjä lähettää HTTP-kyselyn (request), johon palvelin vastaa HTTP-vastauksella (response). Sekä kysely, että vastaus sisältää metadataa, esimerkiksi tiedon siitä, missä muodossa tai minkä kielinen vastaus hyväksytään.

Vaikka HTTP on rakennettu TCP:n päälle, on se käytännössä yhteydetön protokolla. Jokaiselle kyselylle muodostetaan uusi yhteys, joka suljetaan heti vastauksen jälkeen. Tiedon sisällyttäminen metadataan mahdollistaa saman tiedon käyttämisen eri kyselyiden (yhteyksien) välillä. (43) (44)

4.5.3 Hopping Penguinin palvelin

Hopping Penguinissa valittiin pelaajan ja palvelimen väliseksi yhteysmuodoksi HTTP. TCP:n varmuus sekä HTTP:n metadata luo hitautta verrattuna esimerikiksi moninpeleissä käytettyyn UDP:hen, mutta Hopping Penguinin tapauksessa järjestelmän nopeudella ei ole suurta merkitystä, sillä nopeuserot näiden järjestelmien välillä on sekunnin murto-osien tasolla. (45)



Kuvio 19. Pelaajan ja palvelimen välinen kommunikaatio

Koska HTTP on laajasti käytetty protokolla, sen käyttämiseen löytyy runsaasti valmiita työkaluja ohjelmointikieleen tai alustaan katsomatta. HTTP:n käyttö myös mahdollistaa testauksen normaalilla web-selaimella.

Hopping Penguinin alkuvaiheessa palvelimelle tallennettiin vain virtuaalinen kuitti käyttäjän pelin sisäisistä ostoksista, sekä palvelinta käytettiin välikätenä mainosten välittämisessä pelaajalle. Käytimme samaa Suncometin ylläpitämää web-palvelinta, joka pyöritti Hopping Penguinin kotisivua.

Kun peliin lisättiin pilvitallennus (pelitilan tallennus palvelimelle), Suncometin kanssa olisi tullut rajoitukset vastaan. Suncomet rajoittaa HTTP-kyselyiden määrän 50 000:een vuorokaudessa. Pelissä tehdään runsaasti HTTP-kyselyitä, sillä jokainen pelin lataus, tallennus, mainoksen näyttökerta tai erikoiskentän lataus käyttää yhden kyselyn ja peliin kehitetään jatkuvasti uusia ominaisuuksia, jotka vaativat yhteyden palvelimeen ja näinollen käyttää kyselyitä. 50 000:n kyselyn katto rajoittaisi pelaajien määrän n. 2000-5000 pelaajaan vuorokaudessa, riippuen pelaajasta sekä pelisession kestosta. (46)

Palvelimen rajoitusten takia päätettiin hankkia parempi, rajoittamaton palvelin. Kriteereinä palvelimelle olivat hyvä skaalautuvuus, hinta sekä luotettavuus. Potentiaalisia vaihtoehtoja olivat Amazon Web Services, Microsoft Azure ja Google App Engine, joista päätettiin valita Google App Engine, koska siinä oli käyttöömme soveltuvin hinnoittelu ja se vaikutti helppokäyttöiseltä verrattuna muihin vaihtoehtoihin. App Engine skaalautuu automaattisesti, oli käyttäjiä sitten 1,000 tai 1,000,000. App Enginen hinnoittelu on tuntipohjainen, joten palvelimesta maksetaan käytännössä vain sen verran, mitä sitä käytetään.

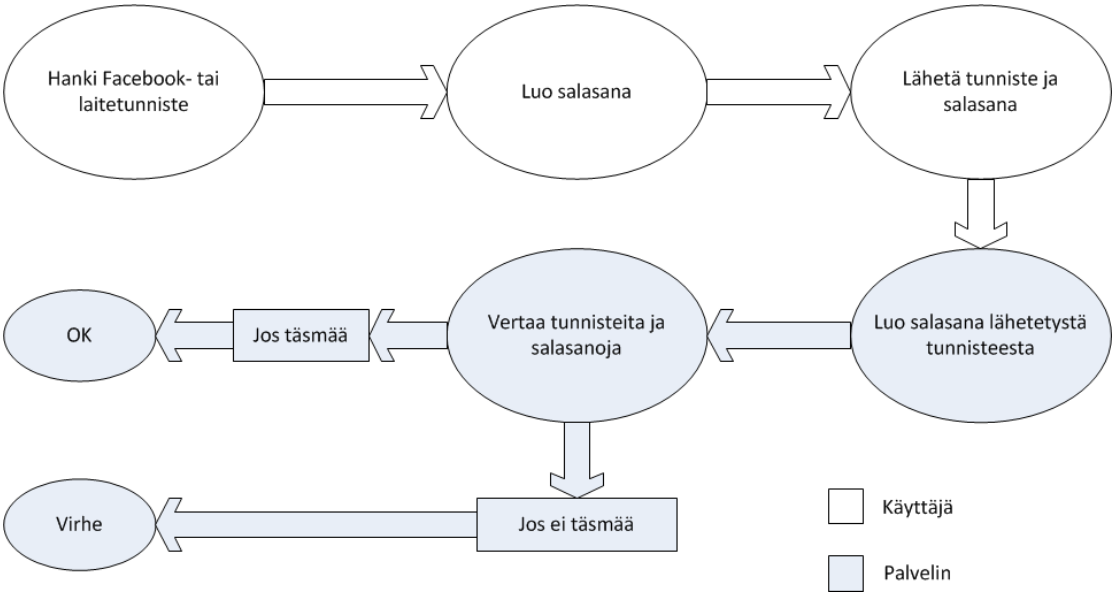
App Engine tarjoaa useita eri ohjelmointikieliä käytettäväksi. Tässä projektissa päädyttiin käyttämään PHP:ta, josta kehittäjillä oli vuosien kokemus.

Palvelimen tehtäviä ovat pilvitallennus, mainosten välittäminen, Facebook-ostosten käsittely, erikoiskenttien toimittaminen, sekä kehittäjien omat työkalut.

4.5.4 Käyttäjän tunnistaminen

Hopping Penguinin Facebook-versiossa käyttäjä tunnistetaan Facebook-tunnisteen avulla. Käyttäjä kirjautuu sisään omilla Facebook-tunnuksillaan hyväksyen Hopping Penguinin ehdot. Hyväksytyn kirjautumisen jälkeen käyttäjästä otetaan talteen Facebook-tunniste. Käyttäjä ei pääse päävalikosta pidemmälle, ellei käyttäjä hyväksy tietojen luovuttamista.

Android ja iOS -alustoilla käyttäjällä on mahdollisuus kirjautua Facebookiin pelin sisällä käyttäen virallista Facebook SDK:ta. Facebookin käyttö Androidilla ja iOS:lla ei ole pakollista, mutta se mahdollistaa pelin sisäisen palkinnon saamisen Hopping Penguinin Facebook-sivun tykkäyksestä sekä saman tallennuksen jatkamisen Facebook-versiolla. (50)



Kuvio 20. Käyttäjän tunnistaminen

4.5.5 Pelitietojen tallentaminen

Hopping Penguinin tallennus tapahtuu siten, että peli lähettää kutsun web-palvelimelle, joka edelleen tallentaa tiedon SQL-tietokantaan. Pelaajalla ei ole suoraa pääsyä tietokantaan. Pilvitallennus mahdollistaa pelin tallennuksen säilymisen, vaikka käyttäjä poistaisi pelin. Tavallisesti peleissä tallennetaan pelin etenemiseen liittyvät tiedot laitteen paikalliseen muistiin, ja siinä tapauksessa tiedot häviävät jos käyttäjä poistaa sovelluksen. Pilveen tallentamisesta on siis käyttäjälle merkittävää hyötyä. Samalla voidaan helposti toteuttaa saman tallennuksen käyttäminen eri laitteilla, sillä tallennus voidaan liittää esimerkiksi Facebook tunnisteeseen.

Järjestelmään tallennetaan jokainen läpäisty kenttä, saavutukset, pelin sisäiset ostokset, kuolemat sekä jäätelömäärä. Näiden lisäksi järjestelmään saatetaan tallentaa tieto onko käyttäjä tykännyt pelistä Facebookissa sekä käyttäjän Facebook-tunniste.

Jokainen järjestelmään tallennettu tapahtuma on yksittäinen rivi SQL-tietokannassa, joka mahdollistaa yksityiskohtaisen ja nopean tiedon hakemisen ja muokkaamisen.

Jokaisella pelin käynnistyskerralla yritetään ladata pelaajan tiedot palvelimelta. Palvelimelle lähetetään parametreinä laitekohtainen tunniste sekä alusta, millä peliä pelataan. Jos tunnistetta ei löydy tietokannasta, käyttäjä lisätään tietokantaan.

Tiedot toimitetaan XML-formaatissa, joka mahdollistaa pelaajan tietojen katsomisen helposti esimerkiksi web-selaimessa.

Jokaisella latauskerralla verrataan paikallista tallennustietoa pilvestä saatuu tietoon ja lähetetään puuttuvat tiedot palvelimelle seuraavan tallennuksen yhteydessä.

Kun pelissä tapahtuu jokin järjestelmää kiinnostava tapahtuma (esim. kuolema, saavutettu saavutus, kentän läpäiseminen, jne.), tallennetaan tieto välimuistiin. Tieto pysyy välimuistissa niin kauan, kunnes palvelimen vastaus osoittaa tapahtuman tallentuneen pilveen. Palvelimeen ei koskaan oteta yhteyttä pelin aikana, sillä se saattaa vaikuttaa pelin suorituskykyyn. Välimuistin käyttö myös vähentää palvelimen kuormaa, kun useampi tieto on yhdistetty yhteen kyselyyn.

Peli yritetään tallennetaan pilveen aina kun pelaaja läpäisee kentän tai ostaa pelin sisäisen oston.

4.5.6 Mainosten välittäminen

Hopping Penguinin Windows Phone -versiota rahoitetaan mainoksilla. Mainosverkoilla ei ole aina tarjota mainosta näytettäväksi sekä niiden palvelimet saattavat olla joskus tavoittamattomissa, joten on järkevää käyttää useaa mainosverkkoa samanaikaisesti.

Kun käyttäjälle päätetään näyttää mainos pelissä, lähettää puhelin kyselyn Hopping Penguin -palvelimelle. Palvelin lähettää kyselyn edelleen useille mainosverkoille yksi kerrallaan, kunnes mainos löytyy tai kaikki mainosverkot on käyty läpi. Kyselyihin liitetään kaikki mahdollinen tieto käyttäjän puhelimesta, jotta oikeanlainen mainos löytyy todennäköisemmin. Jos mainos löytyy, lähetetään se eteenpäin käyttäjän puhelimeen.

Järjestelmä vähentää sellaisia tilanteita, joissa mainosta ei ole näyttää, joka taas lisää mainoksista saatavaa tuottoa. Järjestelmä antaa myös staistiikkaa kehittäjälle, jotta mainosverkkojen suorituskykyä voidaan vertailla reaaliajassa.

4.5.7 Kehittäjän työkalut

Hopping Penguinissa on paljon käyttäjälle näkymättömiä ominaisuuksia. Näillä ominaisuuksilla pyritään siihen, että mahdollisimman paljon pelistä on muokattavissa ilman, että käyttäjän

tarvitsee päivittää peliä uudempaan versioon. Palvelimelta ladataan esimerkiksi tieto siitä, miten käyttäjä palkitaan Facebook-tykkäyksestä. Esimerkiksi Android-version julkaisussa voidaan antaa mainokset pois ilmaiseksi, jos tykkään Hopping Penguinista Facebookissa. Kun käyttäjämäärä on tarpeeksi korkea, voi mainosten poiston tykkäämällä ottaa pois käytöstä.

4.5.8 Pelin sisäiset ostokset

Peli lataa sisäisten ostosten hintatiedot internetistä jokaisen käynnistyksen yhteydessä. Oikealla rahalla tehtävien ostosten hinta ladataan kunkin alustan omasta järjestelmästä, joka hoitaa myös itse maksutapahtuman. Onnistuneesta maksusta annetaan tieto pelille, joka antaa ostetun tuotteen pelaajalle.

Pelin sisäisen valuutan (jäätelöt) tiedot taas ladataan Hopping Penguinin palvelimelta. Kehittäjä voi määritellä pelin sisäisellä valuutalla tehtävien tuotteiden hinnan omilla työkaluillaan, jolloin hintoja voi optimoida reaaliajassa siten, että käyttäjä käyttäisi mahdollisimman paljon oikeata rahaa ostosten tekemiseen.

4.5.9 Palvelimelta ladattavat kentät

Erikoiskentät ovat erikseen ostettavia, (palvelimelta) ladattavia kenttiä, joita kehittäjä voi lisätä ja päivittää reaaliajassa. Erikoiskentät ostetaan pelin sisäisellä valuutalla.

Erikoiskentissä käytettiin aluksi Unityn tarjoamaa Asset Bundle -ominaisuutta, joka on tarkoitettu dynaamiseen sisällön lataamiseen. Hyvin nopeasti järjestelmän julkaisun jälkeen huomattiin, että Asset Bundle -ominaisuuden kanssa oli ongelmia taaksepäin yhteensopivuuden kanssa. Jos pelaaja yritti ladata kenttää vanhemmalla versiolla millä kenttä oli tehty, kenttä ei toiminut.

Asset Bundle -ongelmien takia kehitettiin oma järjestelmä erikoiskenttien toimittamiseen. Erikoiskentät toimitetaan kahdessa vaiheessa. Ensin ladataan lista ladattavissa olevista kentistä, jonka jälkeen valittu kenttä ladataan ellei uusinta versiota ole ladattu aikaisemmin.

4.5.10 Mainosbannerit

Hopping Penguinia voidaan mainostaa muissa sovelluksissa mainosbannereilla. Sovellus kysyy mainosbanneria Hopping Penguinin palvelimelta, antaen paremetriksi käyttäjän maan. Palvelin antaa vastauksena annetulle maalle asetetun bannerin tai vakiobannerin, jos annetulla maalla ei löydy banneria. Mainostava sovellus ohjaa käyttäjän Hopping Penguinin palvelimelle, joka rekisteröi kyseisen bannerin painalluksen ja ohjaa laitteen Hopping Penguinin lataussivulle.

Kehittäjä voi seurata bannereiden suorituskykyä maakohtaisesti ja päivittää bannereita järjestelmän avulla.

4.5.11 Statistiikka ja yksittäisen pelaajan tiedot

Hopping Penguinissa on käytössä kolmannen osapuolen analytiikkajärjestelmiä, mutta omasta tietokannasta on helppo hakea yksityiskohtaista dataa sellaisessa muodossa, mitä analytiikkajärjestelmät eivät tarjoa. Koska analytiikkajärjestelmät analysoi dataa pelaajamassoina, tietynlaista dataa on hankala tai jopa mahdoton saada näiden järjestelmien kautta. Omasta tietokannasta voi hakea juuri sellaista tietoa, mille on tarvetta eikä tietoa rajoita kolmannen osapuolen järjestelmät.

Myös yksittäisen pelaajan tietoja (tallennusta) on mahdollista muokata. Tämä helpottaa testausta ja mahdollistaa hyvän asiakaspalvelun, sillä kehittäjä voi antaa esimerkiksi jäätelöitä tai erikoiskenttiä ilmaiseksi, jos käyttäjä on kokenut ongelmia pelin kanssa ja ottanut yhteyttä kehittäjään.

4.5.12 Tietoturva

Järjestelmä on suunniteltu siten, että käyttäjällä on oikeus tallentaa ja ladata vain omaan laitteeseensa liittyviä tietoja. Tiedot lähetetään selkokielisenä (salaamattomina), sillä kehittäjät eivät koe tarpeelliseksi salata anonyymiä pelidataa.

Pelin sisäiset ostot tehdään kunkin alustan omilla virallisilla kirjastoilla, jonka tietoturvaan kehittäjillä ei ole mahdollisuutta vaikuttaa.

Kehittäjän työkalut on suojattu Google App Enginen toimesta. Työkaluihin pääsee käsiksi vain jos on kirjautunut Google-tilille sekä kirjautunut käyttäjä omaa admin-oikeudet App Engine projektiin.

5 POHDINTA

Hopping Penguinin kehittäminen sujui mielestämme ilman isompia ongelmia, mutta parannettavaa toki löytyy. Seuraaviin projekteihin pitäisi kehittää jonkinlainen järjestelmä tiedon jakamiseen ohjelmoijien kesken. Nykyisellä mallilla kummallakaan ohjelmoijalla ei ollut tietoa siitä, miten toisen ohjelmoijan tekemä järjestelmä toimii, jonka seurauksena ohjelmoijilla kuluu turhaa aikaa kun joko selvittää itse miten järjestelmä toimii tai kysyy toiselta, joka taas kuluttaa molempien aikaa. Ohjelmakoodista ei ollut siis minkäänlaista dokumentaatiota.

Toinenkin ongelma liittyy ainakin osittain ohjelmoijien väliseen tiedonjakoon. Koodia ei suunniteltu etukäteen, joka johti siihen, että jokin järjestelmä saatettiin ohjelmoida kokonaan uudestaan, kun ohjelmoijilla oli eri näkemykset siitä, miten järjestelmän tulisi toimia.

Hopping Penguinin alkuvaiheessa olimme vielä sitä mieltä, että pienellä tiimillä ei tarvitse käyttää projektinhallintatyökaluja. Olimme väärässä. Lapulle kirjoitetut muistilistat eivät kulje toimiston ja kotikoneen välillä, eikä ohjelmoijalla ole tietoa siitä, mitä toinen ohjelmoija tekee. Muutamaan otteeseen ohjelmoijat olivat koodanneet samanlaista järjestelmää samaan aikaan. Bugien seuraaminen oli hankalaa, sillä koodaamiseen syventyessä ei välttämättä tiedosta suunnitelijan ilmoittamaa bugia tai se yksinkertaisesti unohtuu. Projektinhallintajärjestelmästä olisi huomattava etu myös testauksessa. Itse korjaamaa bugia ei osaa välttämättä testata niin monipuolisesti, kuin joku toinen osaisi. Projektinhallintajärjestelmä mahdollistaisi esimerkiksi sellaisen järjestelmän, jossa ohjelmoija merkkaa bugin korjatuksi, jolloin jokin toinen tiimin jäsenistä voi testata ja varmistaa että virhe on korjattu, joka taas merkkaisi bugin korjatuksi.

Hopping Penguinin tiimillä on reilusti osaamista, mutta projektinhallinnassa on paljon parannettavaa.

Kyseessä oli tiimin ensimmäinen vakavasti otettava kaupallinen peli ja se opetti meille todella paljon. Tekstissä manatun projektinhallinnan lisäksi opimme paljon pelin julkaisemiseen liittyviä asioita sekä miten peleillä ei tehdä rahaa.

LÄHTEET

(1) Business of Games: Mobile games market to double in size until 2016 and reach $23.9BN, http://www.businessofgames.com/mobile-games-market-double-size-2016-reach-23-9bn/, luettu 13.10.2014

(2) Start Developing iOS Apps Today: Setup, https://developer.apple.com/library/ios/referencelibrary/GettingStarted/RoadMapiOS/index.html, luettu 18.9.2014, 2013

(3) http://en.wikipedia.org/wiki/Integrated_development_environment

(4) Walkthrough: Using the Visual Studio IDE, http://msdn.microsoft.com/en-us/library/ms235632%28v=vs.80%29.aspx, luettu 15.10.2014

(5) Creating a Software Development Kit, http://msdn.microsoft.com/en-us/library/hh768146.aspx, luettu 15.10.2014

(6) API Definition from PC Magazine Encyclopedia, http://www.pcmag.com/encyclopedia/term/37856/api, luettu 3.11.2014

(7) Understanding UNIX permissions and chmod, http://www.perlfect.com/articles/chmod.shtml, luettu 5.9.2014

(8) Brian's 10 Rules for how to write cross-platform code | Backblaze Blog | The Life of a Cloud Backup Company, https://www.backblaze.com/blog/10-rules-for-how-to-write-cross-platform-code/, luettu 7.10.2014

(9) Supporting Multiple Screens | Android Developers, http://developer.android.com/guide/practices/screens_support.html, luettu 8.10.2014

(10) IDC: Smartphone OS Market Share 2014, 2013, 2012, and 2011, http://www.idc.com/prodserv/smartphone-os-market-share.jsp, luettu 12.11.2014, 2014

(11) What is a Game Engine? - GameCareerGuide.com , http://www.gamecareerguide.com/features/529/what_is_a_game_.php, luettu 5.10.2014, 2008

(12) Tutorial 5 : A Textured Cube | opengl-tutorial.org, http://www.opengl-tutorial.org/beginners-tutorials/tutorial-5-a-textured-cube/, luettu 4.9.2014, 2014

(13) Gamasutra - Adding Languages to Game Engines, http://www.gamasutra.com/view/feature/131641/adding_languages_to_game_engines.php, luettu 17.9.2014, 1997

(14) Cocos2d-x: World's #1 Open Source Game Development Platform, http://www.cocos2d-x.org/, luettu 13.8.2014, 2014

(15) Cocos2d is a family of open-source software frameworks for building cross-platform games&apps, http://cocos2d.org/, luettu 13.8.2014, 2014

(16) Engine Architecture | Cocos2d-x, http://www.cocos2d-x.org/wiki/Engine_Architecture, luettu 13.8.2014

(17) Games | Cocos2d-x, http://cocos2d-x.org/games, luettu 13.8.2014, 2014

(18) Unity - Game Engine, http://unity3d.com/, luettu 15.8.2014, 2014

(19) Unity - Multiplatform - Publish your game to over 10 platforms, http://unity3d.com/unity/multiplatform, luettu 15.8.2014, 2014

(20) GamaSutra 2012 (unityn käyttö on yleistynyt huomattavasti)

(21) Unity - Store, https://store.unity3d.com/, luettu 15.8.2014, 2014

(22) Unity - Asset Workflow - Line up your assets in your game with Unity's fast and no-fuss asset pipeline, http://unity3d.com/unity/workflow/asset-workflow, luettu 15.8.2014

(23) Git - Versionhallinnasta, http://git-scm.com/book/fi/v1/Alkusanat-Versionhallinnasta, luettu 18.9.2014

(24) Git - Git Basics, http://git-scm.com/book/en/v2/Getting-Started-Git-Basics, luettu 18.9.2014

(25) Measuring Game Engine Popularity | Learn & Master Cocos2D Game Development, http://www.learn-cocos2d.com/2013/07/measuring-game-engine-popularity/, luettu 21.10.2014, 2013

(26) Gamasutra: Grace Kuo's Blog - Marketing 101: Everything you need to know about marketing your mobile game, http://www.gamasutra.com/blogs/GraceKuo/20140624/219678/Marketing_101_Everything_you_need_to_know_about_marketing_your_mobile_game.php, luettu 22.10.2014, 2014

(27) Microsoft says Windows Phone now touts 300,000 apps, http://www.t3.com/news/microsoft-says-windows-phone-now-touts-300000-apps, luettu 14.9.2014

(28) Marketing and Monetization of Windows Phone Applications, http://www.slideshare.net/vkalve/windows-phone-promo-monetize, luettu 12.8.2011

(29) AppCademy

(30) W8 & WP8 D3D renderer - Cocos2d-x Forum http://discuss.cocos2d-x.org/t/w8-wp8-d3d-renderer/, luettu 4.10.2014,

(31) What is Project Management? | Project Management Institute, http://www.pmi.org/About-Us/About-Us-What-is-Project-Management.aspx, luettu 8.11.2014

(32) Agile Game Development: EVE Online Fanfest 2009 - Scrum & Agile, http://blog.agilegamedevelopment.com/2010/02/eve-online-fanfest-2009-scrum-agile.html, luettu 9.11.2014

(33) The four pillars of Agile game development | Analysis | Develop, http://www.develop-online.net/analysis/the-four-pillars-of-agile-game-development/0117636, luettu 9.11.2014

(34) Morris, Peter W.G, Jogh Wiley & Sons March 2013

(35) Gamasutra - Making Better Games Through Iteration, http://www.gamasutra.com/view/feature/132554/making_better_games_through_.php, luettu 1.10.2014, 2009

(36) Extensible Markup Language (XML) 1.0 (Fifth Edition), http://www.w3.org/TR/2008/REC-xml-20081126/, luettu 12.9.2014, 2008

(37) Outline Games, http://outlinegames.com/unibill-documentation/, luettu 20.8.2014

(38) http://en.wikipedia.org/wiki/Mobile_Internet_growth, luettu 20.8.2014

(39) BBC News - Over 5 billion mobile phone connections worldwide, http://www.bbc.co.uk/news/10569081, luettu 25.10.2014, 2010

(40) Analytics | Flurry, http://www.flurry.com/solutions/analytics, luettu 26.10.2014, 2014

(41) The Math Behind A/B Testing, https://developer.amazon.com/sdk/ab-testing/reference/ab-math.html, luettu 26.10.2014

(42) IPv6.com - User Datagram Protocol (UDP), http://ipv6.com/articles/general/User-Datagram-Protocol.htm, luettu 3.9.2014,

(43) A Protocol for Packet Network Intercommunication , http://ece.ut.ac.ir/Classpages/F84/PrincipleofNetworkDesign/Papers/CK74.pdf, luettu 5.10.2014, 1977

(44) RFC 2616 - Hypertext Transfer Protocol -- HTTP/1.1, https://tools.ietf.org/html/rfc2616, luettu 20.10.2014, 1999

(45) TCP - UDP Comparative analysis - Data Communications and Networks - Free Computer Science Tutorials - Provided by Laynetworks.com, http://www.laynetworks.com/Comparative%20analysis_TCP%20Vs%20UDP.htm, luettu 21.10.2014

(46) Webhotellit ja Verkkotunnukset - Suncomet | Ominaisuudet, http://suncomet.com/index.php?verkkohotellienominaisuudet#rajoitukset, luettu 15.10.2014

(47) Unleashing App Engine Scalability - Google Cloud Platform, https://cloud.google.com/developers/articles/unleashing-appengine-scalability/, luettu 16.10.2014

(48) Azure Websites and Apps, http://azure.microsoft.com/en-us/services/websites/, luettu 16.10.2014

(49) Amazon Web Services (AWS) - Cloud Computing Services, http://aws.amazon.com/, luettu 16.10.2014

(50) Unity, https://developers.facebook.com/docs/unity, luettu 27.9.2014