PELINKEHITYS UNITY-PELIMOOTTORILLA WINDOWS PHONE -LAITTEILLE

HAMK
HÄMEEN AMMATTIKORKEAKOULU

Ammattikorkeakoulun opinnäytetyö

Tietojenkäsittelyn koulutusohjelma

Visamäki, syksy 2014

Markus Uotila

HAMK
HÄMEEN AMMATTIKORKEAKOULU

TIIVISTELMÄ

VISAMÄKI
Tietojenkäsittelyn koulutusohjelma
Systeemityö

Tekijä Markus Uotila **Vuosi** 2014

Työn nimi Pelinkehitys Unity-pelimoottorilla Windows Phone -laitteille

TIIVISTELMÄ

Opinnäytetyön tavoite oli oppia kehittämään pelejä Unity-pelimoottorilla Windows Phone -laitteille. Työn aihe oli itse keksitty, mutta toimeksiantajana toimi HAMK. Opinnäytetyöstä valmistui kaksi tuotosta, joita ovat kuulalabyrinttipeli ja malliprojekti. Näiden tekemisessä oleellisia asioita olivat kaksiulotteinen grafiikka, puhelimen kosketusnäytön ja anturien käyttäminen sekä fysiikkamoottorin hyödyntäminen. Malliprojektia voidaan mahdollisesti käyttää tulevaisuudessa opetuksen apuna ja se oli toimeksiantajan vaatimus opinnäytetyöstä.

Aiempaa kokemusta ohjelmoinnista ja pelien kehittämisestä oli saatu koulun kursseilta. Mobiilisovellusten kehittämistä oli tehty erikoistumisprojektin ja työharjoittelun yhteydessä, mutta Android-käyttöjärjestelmään keskittyen. Perusteet olivat siis hallussa, mutta Unity oli täysin uusi asia opinnäytetyön tekemisen yhteydessä. Toimeksiantajan kanssa sovittiin niin, että malliprojekti olisi yksinkertainen ja toimisi perehdytyksenä Unityyn. Malliprojektista saatua osaamista ja sen tekemiseen kirjoitettua koodia käytettiin varsinaisessa pelissä.

Työn teoriaosuudessa käytiin läpi Unityn ominaisuuksia sekä sen käytön perusteita, kuten käyttöliittymää ja projektien koostumusta. Windows Phonesta käytiin läpi sovelluksen julkaisuprosessi sen sovelluskauppaan. Materiaalina toimivat verkkolähteet, jotka olivat pääosin Unityn ja Microsoftin omilta sivuilta.

Työtä tehdessä Unity todettiin toimivaksi ja laajat ominaisuudet tarjoavaksi pelimoottoriksi. Sillä saatiin nopeasti kuvaa näkyviin, eikä sitä käyttäessä törmätty suuriin ongelmiin. Joitakin ongelmia tehdessä tuli vastaan, mutta niistä selvittiin dokumentaatiota tutkimalla. Tehdystä pelistä jatkokehitettäväksi jäi graafisen ulkoasun viimeistely, eikä sitä tämän vuoksi julkaistu Windows Phone Storessa.

Avainsanat pelinkehitys, Unity, Windows Phone, mobiilipeli

Sivut 29 s.

VISAMÄKI
Degree Programme in Business Information Technology
System Development

Author Markus Uotila **Year** 2014

Subject of Bachelor's thesis Game development with Unity engine for Windows Phone devices

ABSTRACT

The goal of this Bachelor's thesis was to learn to develop games using the Unity game engine for mobile devices and in particular for Windows Phone devices. The thesis was commissioned by HAMK University of Applied Sciences. The aim was to produce two products i.e. a game and a sample project using two-dimensional graphics, the phone's touch screen and sensors and the physics engine. The sample project can be potentially used in the future as a teaching aid at HAMK.

In the theoretical part of the thesis Unity's features, the basics of its use, such as the user interface and project components were discussed. The publishing applications of the Windows phones to the Windows Phone Store were also dealt with. The sources used were mainly Unity's and Microsoft's own websites.

As a result of the thesis a ball maze game and a sample project were developed. Unity was found to be a functional game engine offering extensive features. It enabled a quick graphics drawing and no major problems were encountered. However, there were some minor problems but they were solved by studying the documentation. Finishing the graphics of the game produced was left for further development. Therefore it was not yet released in the Windows Phone Store.

Keywords game development, Unity, Windows Phone, mobile game

Pages 29 p.

TERMISTÖ

3D-malli on matemaattinen esitys kolmiulotteisesta kohteesta. Kohteita voivat olla esim. ihminen, pöytä, tai talo jne.

Grafiikka tarkoittaa kaiken pelissä näkyvän ulkoasua.

Kaksiulotteinen peli tarkoittaa, että kaiken näkyvän sijainti määritellään kahdella koordinaatilla, eli vaaka- ja pystysuunnassa.

Kolmiulotteinen peli tuo mukaan kolmannen koordinaatin, eli syvyyden.

Luokka määrittää ohjelmoinnissa olion rakenteen. Luokasta voidaan tehdä useita olioita.

Metodi tarkoittaa ohjelmoinnissa käskyä, joka suorittaa jonkin asian.

Ohjelmointikieli on tietokoneen ohjelmointiin suunniteltu kieli, jolla on oma sanasto ja kielioppisäännöt. Ohjelmointikieliä on useita erilaisia.

Projekti on kokoelma tiedostoja, jotka on järjestetty ja määritetty siten, että jokin ohjelma osaa avata sen käyttäjäystävällisellä tavalla.

Tekstuuri on kaksiulotteinen kuva. Peleissä kaiken ulkoasu koostuu tekstuureista.

SISÄLLYS

1 JOHDANTO

Opinnäytetyöstä valmistuvia tuotoksia ovat kuulalabyrinttipeli ja malliprojekti, jota voidaan myöhemmin mahdollisesti hyödyntää opetuksessa. Molemmat tuotokset toteutetaan kaksiulotteisena. Työssä käydään läpi pelimoottorin tukemia ominaisuuksia, sen käyttämistä ja pelin tekemisessä oleellisia asioita, kuten ohjelmointi, sekä grafiikan ja äänien käyttäminen. Työn aihe on itse keksitty, joten toimeksiantajana toimii HAMK. Toimeksiantajan vaatimus työstä on sen ensimmäinen tuotos, eli yksinkertainen malliprojekti, jossa on grafiikkaa, kosketuksen ja kiihtyvyysanturien lukeminen, sekä törmäyksen tunnistaminen. HAMK Visamäen yksikössä ei tällä hetkellä ole Unity-kursseja, mutta malliprojektin tarkoitus on opetuskäyttö. Malliprojektista ei kuitenkaan tehdä valmista kurssia tai ohjetta. Työssä tullaan kertomaan Windows Phone -mobiilisovelluksen julkaisuprosessista ja sen eri vaiheista. Mobiililaitteita tukevia pelimoottoreita on Unityn lisäksi monia muita, mutta työssä ei tulla vertailemaan niitä.

Aiheeseen päädyin, koska pelit ovat minusta mielenkiintoinen aihe ja toivon pääseväni työskentelemään niiden parissa tulevaisuudessa. Ohjelmoinnista minulla on kokemusta myös mobiililaitteilla, mutta ei kovin paljon pelien kehittämisestä. Grafiikan ja äänien suhteen voidaan käyttää jo olemassa olevaa materiaalia, mikäli niiden lisenssi sen sallii. Mikäli sopivaa ei ole, nämä tehdään itse. Työssä ei keskitytä tarkasti materiaalin luomiseen, kuten äänien nauhoittamiseen ja tekstuuriksi kelpaavien valokuvien ottamiseen.

Aiheena pelinkehitys on erittäin ajankohtainen, sillä peliteollisuus on nopeasti kasvava ala, joka on jo mm. ohittanut arvollaan elokuvat ja musiikin Britanniassa (BBC 2012). Pelkästään mobiilipelien tämänhetkinen vuosittainen myyntitulojen arvo on 17 miljardia dollaria ja vuonna 2015 sen arvioidaan olevan 22 miljardia dollaria (Gartner 2013). Windows Phone on käyttöjärjestelmänä melko uusi ja tämän vuoksi sen sovelluskauppa ei ole yhtä täynnä sisältöä, kuten Android- ja iOS-käyttöjärjestelmien vastaavat. Tästä johtuen näen Windows Phonen sovelluskauppaan tähtäämisessä paljon potentiaalia.

Opinnäytetyön tutkimuskysymyksiä ovat:

- Mistä koostuu Unity-projekti?
- Miten Unitylla piirretään grafiikkaa, luetaan laitteen antureita ja käytetään fysiikkamoottoria?
- Mikä on mobiilisovelluksen julkaisuprosessi?

2 UNITY

Unity on pelimoottori, joka on pelikehittäjien keskuudessa noussut suuren suosioon. Pelien lisäksi sitä käytetään myös simulaattorien luomiseen ja rakennusten arkkitehtuurin esittelyyn.

2.1 Historia

Unity pelimoottorin kehittämisen aloittivat vuonna 2001 David Helgason, Joachim Ante ja Nicholas Francis. Myöhemmin vuonna 2004 he perustivat yrityksen nimeltään Unity Technologies. Ensimmäinen versio julkaistiin vuonna 2005, jolloin ainoa tuettu käyttöjärjestelmä oli Applen kehittämä OS X. Myöhemmin lisättiin myös tuki web-selaimille ja Windows-käyttöjärjestelmälle. (Brodkin 2013.)

Unity oli ensimmäinen pelimoottori, joka tuki vuonna 2008 Applen julkaisemaa iPhone-älypuhelinta. Samana vuotena Cartoon Network käytti pelimoottoria julkaistessaan pelin FusionHall, joka on saavuttanut jo kahdeksan miljoonaa pelaajaa. Pelimoottori sai myös tue Nintendo Wii -pelikonsolille. Nopeasti kasvaneen suosion johdosta Unity Technologies kolminkertaisti työntekijöidensä määrän. (Brodkin 2013.)

Ilmainen versio pelimoottorista julkaistiin vuonna 2009. Rekisteröityjä kehittäjiä Unitylla oli tässä vaiheessa jo 10 000 ja sitä oli käytetty yli 325 iPhonelle julkaistussa pelissä. (Unity Technologies 2009.)

Unity 3 julkaistiin vuonna 2010 ja sen mukana tuli tuki Android-käyttöjärjestelmälle. Rekisteröityjen kehittäjien määrä oli vuoden loppuun mennessä 250 000 ja Unity pelien pelaamisen mahdollistava selainliitännäinen oli asennettu 35 miljoonaa kertaa. (Unity Technologies 2014a.)

Vuonna 2013 Unityyn lisättiin tuki Windows Store -sovelluksille ja Windows Phone-, Tizen- ja BlackBerry-käyttöjärjestelmille. Pelimoottoriin myös lisättiin virallinen tuki kaksiulotteisille peleille. (Balasevicius 2013; Marketwired 2013.)

Tuki Windows Phone 8.1 ja sen mukana tulleille universaaleille sovelluksille lisättiin 2014. (Tautvydas Zilys 2014.) Tällä hetkellä Unitya käyttää 47 % prosenttia pelikehittäjistä ja rekisteröityjä kehittäjiä on 3,3 miljoonaa. Yhdysvaltojen markkinoilla parhaiten myyvistä kolmiulotteisista mobiilipeleistä 50 % käyttää Unitya. (Unity Technologies 2014a.)

2.2 Ominaisuudet

Unity on pelimoottori, jonka mukana tulevat työkalut, joilla voidaan useissa tapauksissa kehittää peli alusta loppuun. Virallisesti tuettuja alustoja pelin julkaisulle ovat Android, BlackBerry 10, iOS, Linux, OS X, Playstation 3, Playstation Vita, Wii, Wii U, Windows, Windows Store, Windows Phone, Xbox 360 ja useat verkkoselaimet liitännäisen avulla. Alustasta riippuen Unity käyttää grafiikan piirtämisen rajapintana Direct3D,

OpenGL, OpenGL ES, tai laitteen omaa rajapintaa. Nopeaa julkaisua useille laitteille helpottaa pelimoottorin tuki automaattiseen materiaalin muokkaamiseen laitteelle sopivaksi lennossa valmista projektia luodessa. Näin voidaan esimerkiksi helposti käyttää korkealaatuisia tekstuureita tietokoneella ja skaalata ne pienemmiksi puhelimille ilman että tarvitsee säilyttää kahta versiota samasta tekstuurista. (Unity Technologies 2014b.)

Unityn mukana tuleva Unity Editor on visuaalinen työkalu pelin materiaalien lisäämiseen projektiin ja niiden käyttämiseen pelin kohtauksia luodessa. Editori koostuu eri näkymistä, joista yleisimmät ovat projektin selain, tutkija, peli, kohtaus ja hierarkia. Projektin selainnäkymässä listataan kaikki projektin materiaalit, tutkijanäkymässä voidaan muokata pelin materiaalien ja olioiden arvoja, kohtausnäkymässä rakennetaan peli ja hierarkianäkymässä näytetään kaikki pelin oliot. Pelinäkymässä näkyy pelin lopullinen näkymä, joten editorilla voidaan nopeasti muokata haluttua kohtausta ja nähdä heti sen vaikutukset. Editorin näkymät voi muokata haluamaansa järjestykseen ja peli voidaan nopeasti käynnistää pelinäkymään esimerkiksi skriptin testaamista varten. (Unity Technologies 2014c.)

Fysiikoiden laskemiseen Unity käyttää kahta eri fysiikkamoottoria. Kolmiulotteisiin peleihin käytetään Nvidia PhysX ja kaksiulotteisiin avoimen lähdekoodin Box2D. Unitysta löytyy myös valmis komponentti auton renkaiden tarkkaan mallintamiseen. (Unity Technologies 2014d.)

2.3 Lisenssi

Unity on mahdollista saada erilaisilla lisensseillä ja hinnoilla, mutta ilman erityisiä neuvotteluja vaihtoehdot ovat ilmainen Unity Free ja maksullinen Unity Pro. Unityn ilmaista lisenssiä käyttämällä on mahdollista tehdä ja julkaista peli, mutta se asettaa rajoituksia pelimoottorin käytön, ominaisuuksien ja pelien julkaisemisen suhteen. Ilmainen lisenssi on rajoitettu yksityisille henkilöille, yrityksille ja yhteisöille, joiden bruttotulot, tai koko vuoden budjetti on alle 100 000$. Lisenssit ovat henkilökohtaisia, mutta ohjelmistoa voidaan käyttää usealla laitteella, kuhan sitä ei tehdä samaan aikaan. (Unity Technologies 2014e.)

2.4 Maksullisen lisenssin ominaisuudet

Maksullisen lisenssin hankkiminen antaa käyttöön lisää mm. graafisia ja pelin kehitystä helpottavia ominaisuuksia. Tässä luvussa käydään läpi Unityn Pro-lisenssin tarjoamia ominaisuuksia ja niiden hyötyjä.

Unity Pron yleisiin ominaisuuksiin kuuluu tuki 3D-mallin laadun heikentämiselle etäisyyden mukaan. Näin suorituskykyä voidaan parantaa tekemällä 3D-mallista vähemmän yksityiskohtaisia versioita, jonka Unity osaa automaattisesti vaihtaa lennossa, kun kamera menee halutun etäisyyden päähän. Parempien äänien luomiseen on lisätty tuki tehosteille, kuten kaiulle ja erilaisten ympäristöjen simuloimiselle. Unity voi muuttaa äänensävyä automaattisesti kameran siirtyessä pois äänen lähteen luota, eli simuloida etäisyyttä. Peleissä käytetään usein videopätkiä ja näiden käyttö

on maksullisen lisenssin alla. Videoita voidaan käyttää myös tekstuureina, jolloin yksi käyttökohde niille voisi olla pelimaailmassa sijaitseva televisio. Videoita, pelin tasoja ja materiaaleja voidaan ladata suoratoistona kesken pelin. Tämän avulla voisi esimerkiksi toteuttaa pelin sisäisiä ostoksia uusille tasoille, eikä jokaisen peliä pelaavan tarvitsisi ladata ylimääräisiä tiedostoja turhaan. Ne voitaisiin ladata vain, jos niitä todella tarvitaan. (Unity Technologies 2014f.)

Animaatiopuolella parannuksia ovat Unityn Mecanim-animointijärjestelmän lisäominaisuudet Mecanim: IK Rigs ja Mecanim: Sync Layers & Additional Curves. IK Rigs lisää Unityyn tuen animaatioille käyttäen käänteiskinematiikkaa. Sync Layers & Additional Curves mahdollistaa animaation eri kerroksien synkronoinnin, jota hyödyntäen voidaan esimerkiksi synkronoida normaalin kävelyn ja hoipertelun animaatiot ja vaihtaa niitä luonnollisen näköisesti. (Unity Technologies 2014f.)

Grafiikan lisäominaisuuksissa mahdollistetaan hienompien efektien tekemistä ja suorituskyvyn parantamista. Lisätty on tuki kolmiulotteisille tekstuureille, joita hyödyntäen voidaan toteuttaa esimerkiksi valon säteiden näyttäminen. Valaistukseen on lisätty tuki HDR- (High Dynamic Range) ja Deferred-renderöinnille. Deferred-renderöijä tarjoaa Unityssa edistyneimmän valaistuksen ja varjot mahdollistaen rajattoman määrän valon lähteitä. Spotti- ja pistevalaistus on mahdollista tehdä reaaliaikaisten ja pehmeiden varjojen kanssa, sekä kenttään valmiiksi laskettu valaistus saadaan vaikuttamaan myös pelihahmoihin Light Probeja käyttäen. Valmiiksi laskettu valaistus voidaan tehdä realistisemmaksi Global Illumination tuen ansiosta. (Unity Technologies 2014f.)

Tekstuurina voidaan käyttää piirrettyä pelikuvaa, jonka avulla voidaan toteuttaa esim. auton taustapeili ja kuvaan pystyy lisäämään jälkikäsittelyä, kuten väripaletin muuttamisen, tai liikkeen sumentamisen. Suorituskykyä parantaviin ominaisuuksiin kuuluvat näytönohjaimen hyödyntäminen varjostimien (shader) käyttämisessä ja 3D-mallien nylkemisessä. 3D-mallin nylkemisellä tarkoitetaan animointia tarvitsevan 3D-mallin ja sitä varten tehdyn luurangon toisiinsa yhdistämistä. Näkökentän ulkopuolelle jäävät pelioliot pystytään myös automaattisesti jättämään piirtämättä ja ilman tätä ominaisuutta näytönohjain piirtää turhaan esimerkiksi seinän takana olevan peliolion, vaikka sitä ei olisi mitenkään mahdollista nähdä. Pelin käynnistyessä näkyvä Unityn logo on myös mahdollista vaihtaa omaan. (Unity Technologies 2014f.)

Ohjelmointiin saatavat ominaisuudet ovat dynaamiset esteet navigointiverkossa ja tuki C-, C++-, Objective-C- ja Java-ohjelmakirjastoille. Dynaamisilla esteillä tekoälyn käyttämä navigointiverkko päivittyy reaaliaikaisesti, jolloin ne osaavat kiertää pelin aikana luotuja esteitä, eivätkä ole rajoittuneita vain kentän lataamisen yhteydessä generoituun navigointiverkkoon. (Unity Technologies 2014f.)

Editoriin tulee saatavaksi suorituskyvyn profiloija. Profiloija auttaa optimoimaan pelin suorituskykyä tallentamalla ja näyttämällä, missä pelin

kohdissa suorituskyky on heikko ja mihin pelikuvan piirtämisessä käytettyä aikaa kuluu eniten. Viimeisinä ominaisuuksina ovat editorin laajennettu tuki skripteille, joka mahdollistaa sovelluksen versioiden jatkuvan paketoinnin ja tumma tyyli editoriin. (Unity Technologies 2014f.)

2.5 Ohjelmointirajapinta (API)

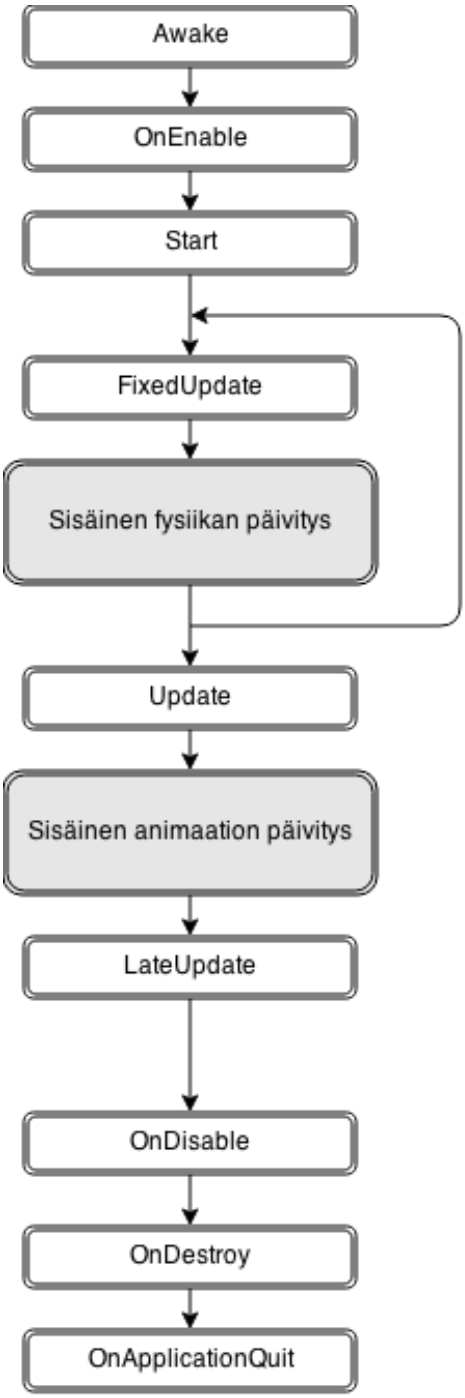
Unityn tukemia ohjelmointikieliä ovat C#, JavaScriptistä johdettu UnityScript ja Boo. Nämä toimivat Unityn käyttämän Monon kanssa, joka on avoimen lähdekoodin toteutus Microsoftin kehittämästä .NET-ohjelmistokomponenttikirjastosta. Projekteille ei erikseen valita ohjelmointikieltä, vaan on mahdollista käyttää jokaista yhdessä projektissa. Unityn nimeämiskäytäntö on kutsua kaikkia koodinpätkiä skripteiksi. Unityn API-dokumentaatio on nähtävillä Unityn sivuilla ja se on järjestetty käytettävissä olevien luokkien mukaan. Luokkien alla selitetään mm. luokan muuttujat ja metodit. Näiden käytön helpottamiseksi on myös esimerkkikoodia, joka on saatavilla kaikilla kolmella tuetulla ohjelmointikielellä. Koodissa käytettyjä muuttujia on mahdollista muokata suoraan Unityn editorissa, jos ne asetetaan julkisiksi. Editorissa näkyvien muuttujien arvoja voidaan vaihtaa pelin ollessa käynnissä, joka nopeuttaa testaamista. Editorissa pyörivän pelin pysäyttäessä muuttujien arvot palaavat ennalleen.

2.6 Gameloop

Unityssa jokainen luokka periytyy MonoBehaviour-luokasta, jolloin niihin pätee Unityn skriptien elinkaari. Koodia suoritetaan elinkaaren eri vaiheissa käyttämällä siihen tarkoitettuja funktioita. Elinkaaren vaiheita on lukuisia, mutta näistä ehkä olennaisimpia ovat skriptin alustus, fysiikan päivityksen sykli, ruudunpäivityksen sykli ja skriptin tuhoutuminen. (Unity Technologies 2014g.)

Alustukseen kuuluvat kohtauksen ensimmäisellä latauskerralla kutsutut Awake- ja OnEnable-funktiot, sekä ennen ensimmäistä ruudunpäivitystä kutsuttava Start. Toistuvia syklejä ovat FixedUpdate, Update, sekä LateUpdate ja ne suoritetaan mainitussa järjestyksessä. Update kutsutaan kerran ruudunpäivityksen yhteydessä. Tätä käytetään mm. pelaajan syötteiden lukemiseen ja peliolioiden liikuttamiseen, kun ne eivät käytä fysiikkamoottoria. Oletuksena ruutuja piirretään sekunnissa niin paljon, kuin on mahdollista, mutta projektille on mahdollista määrittää tavoite ruudunpäivitysnopeus. Tavoite toimii vain ylärajana ja siihen pääseminen on laitteiston suorituskyvystä ja pelin optimoinnista kiinni. Pelit pyrkivät yleensä tasaiseen 30 tai 60 ruudun päivitysnopeuteen. Myös LateUpdate kutsutaan kerran ruudunpäivityksen yhteydessä, mutta se tehdään vasta kun Update on kutsuttu ja se on saanut suoritettua koodinsa. Yksi käyttötarkoitus tälle voisi olla kameran ohjaaminen, joka suoritetaan vasta, kun hahmon kääntyminen on saatu Update-funktiossa varmasti suoritettua. FixedUpdate kutsutaan tasaisin väliajoin, joka voidaan määrittää projektin asetuksista. Se voidaan siis suorittaa useasti, tai ei kertaakaan ruudunpäivityksen aikana. Kaikki fysiikkamoottorin laskut ja toiminnot suoritetaan heti FixedUp-

daten jälkeen. Tämän vuoksi suositellaan suorittamaan fysiikkamoottoria käyttävät toiminnot FixedUpdate-funktiossa. Skriptin tuhoutumiseen kuuluvat OnDisable, OnDestroy ja OnApplicationQuit. Näitä voidaan käyttää esimerkiksi tallentamaan pelaajan tiedot peliä sulkiessa. Kuviossa (1) on yksinkertaistettu versio skriptin elinkaaresta, jossa mukana vain aiemmin mainitut vaiheet. (Unity Technologies 2014g.)



Kuvio 1. Yksinkertaistettu skriptin elinkaari

2.7 Unityn perusteet

Unity-projekti koostuu kohtauksista, peliolioista ja niiden komponenteista, sekä tiedostoista. Uutta projektia luodessa valitaan, onko peli kaksiulotteinen vai kolmiulotteinen, mutta tämä vaikuttaa vain kameran ja tekstuurien tuonnin oletusasetuksiin ja niitä voi muokata jälkeenpäin. Projektia luodessa voi myös ottaa projektiin mukaan Unityn tarjoamia paketteja, joissa on mm. valmiita skriptejä ja graafisia efektejä.

Jokaisella projektilla on oltava vähintään yksi tallennettu kohtaus, jotta sitä voidaan testata, tai se voidaan julkaista. Yksinkertaistettuna kohtauksia voidaan pitää pelin tasoina, mutta myös valikot, tai välianimaatiot voivat olla omia kohtauksia. Kohtauksilla on omat peliolionsa, mutta jokaisessa kohtauksessa on oltava kameraolio. Unity luo uuden projektin mukana kohtauksen kameraolion kanssa, mutta tämä tulee itse tallentaa. (Unity Technologies 2014h.)

Kaikki kohtauksissa nähtävät asiat ovat peliolioita. Peliolio ei kuitenkaan itsekseen ole muuta, kuin säiliö komponenteille. Pelioliolla voi olla useita komponentteja, mutta jokaisella on muunnoskomponentti. Muunnoskomponentti on yksi tärkeimmistä komponenteista, sillä se määrittää peliolion sijainnin, kierron ja mitat kohtauksessa. Yleensä pelioliolla on kuitenkin, muitakin komponentteja, sillä peliolio ei näy, tai toimi millään tavalla kohtauksessa pelkällä sijainnin määrittämisellä. Peliolioita ilman komponentteja on kuitenkin mahdollista käyttää kansioiden tapaan säiliönä muille peliolioille. Tämä voi helpottaa kohtauksen peliolioiden tarkastelua hierarkianäkymässä, mutta niiden tekeminen ei ole välttämätöntä. Komponentteja on useita ja ne on jaettu kategorioihin verkko, efektit, fysiikat, fysiikat 2D, navigointi, ääni, renderöinti, sekalaiset ja skriptit. (Unity Technologies 2014i; Unity Technologies 2014j.)

Pelissä useasti käytettävistä peliolioista on hyvä idea tehdä valmisolio. Valmisoliot ovat malleja, joista luodaan uusia peliolioita samoilla komponenteilla ja niiden parametreilla. Kun peliolio on luotu valmisoliosta, voidaan valmisoliota, tai yhtä pelioliota muokkaamalla muuttaa kaikkia valmisoliosta luotuja peliolioita. On kuitenkin myös mahdollista muokata vain yksittäisen valmisoliosta luodun peliolion komponentteja ja niiden parametreja. Peliolioita voi myös perinteisesti kopioida ja liittää, mutta näin tehdessä joutuu itse muistamaan niiden muutosten päivittämisen. Valmisoliota olisi esimerkiksi hyvä käyttää yhden vihollistyypin luomiseen, sillä näin voitaisiin helposti muuttaa kaikkien valmisoliosta luotujen vihollisten kestävyyttä, jos ne huomataan testauksessa liian vaikeaksi. (Unity Technologies 2014k.)

Projektin tiedostoihin kuuluvat pelissä käytettävät tekstuurit, äänet, 3D-mallit ja skriptit. Nämä tuodaan projektiin viemällä ne projektin Assets nimiseen kansioon, tai raahaamalla ne Unityn ikkunaan. Tiedostojen nimiä voi muuttaa ja niitä voi järjestellä kansioihin Unityn selainnäkymässä, mutta ei käyttöjärjestelmän tiedostoselaimen kautta, sillä Unity pitää tallessa tietoja tuoduista tiedostoista. Tiedostoja voi muokata niiden Unityyn tuonnin jälkeen myös muissa ohjelmissa, kunhan nimi pysyy samana. Uni-

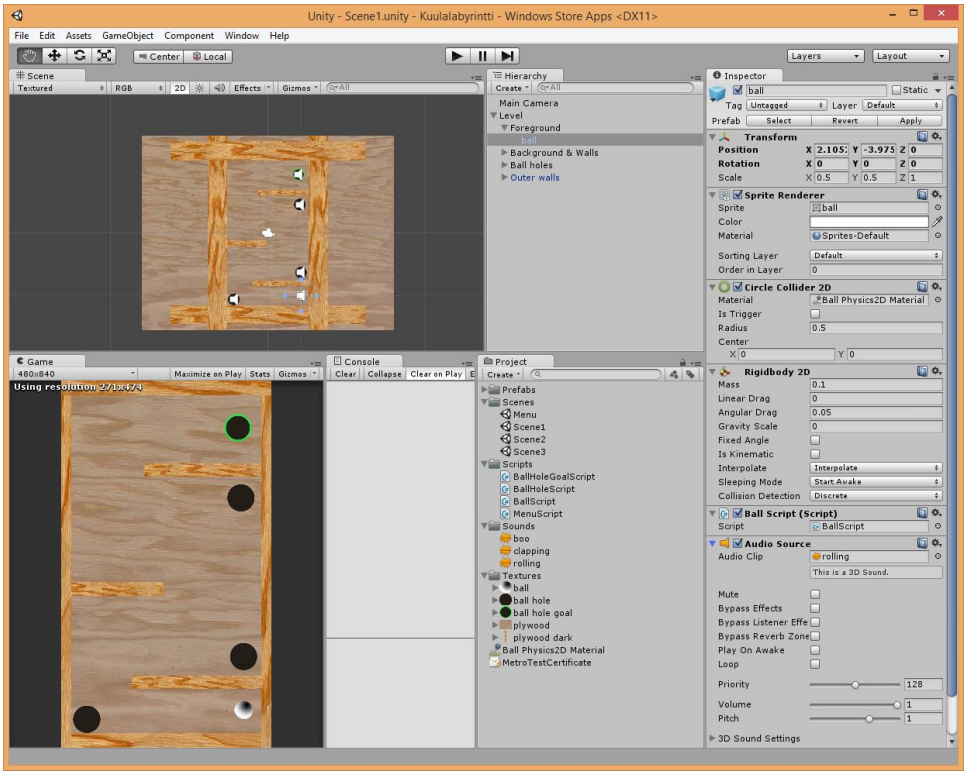
ty huomaa tehdyt muutokset automaattisesti ja päivittää projektin näkymän. (Unity Technologies 2014l.)

Tekstuurina voidaan käyttää useita formaatteja mukaan lukien JPG, PNG, BMP, sekä Photoshopin käyttämä työskentelyformaatti PSD. Tekstuurien mittasuhteen suositellaan olevan kahden potenssi, eli esimerkiksi resoluutio 256x256. (Unity Technologies 2014m.)

Ääniformaateista tuettuja ovat mm. MP3, OGG ja WAV. Tiedostoille ei tehdä oletuksena mitään niitä projektiin tuodessa, mutta peliä julkaistaessa ne on muunnettava kohdelaitteen tukemaan muotoon. Tietokoneilla käytetty formaatti on OGG, mobiililaitteilla MP3 ja molemmat tukevat pakkaamatonta WAV-formaattia. Pakkaamattoman formaatin etuna ovat parempi suorituskyky ja äänenlaatu, mutta tiedostojen koko on pakattuja suurempi. Äänien pakkaus voidaan valita tiedostokohtaisesti ja pakkaamattomien äänien suuren koon vuoksi niitä ei suositella kuin lyhyille äänille. (Unity Technologies 2014m.)

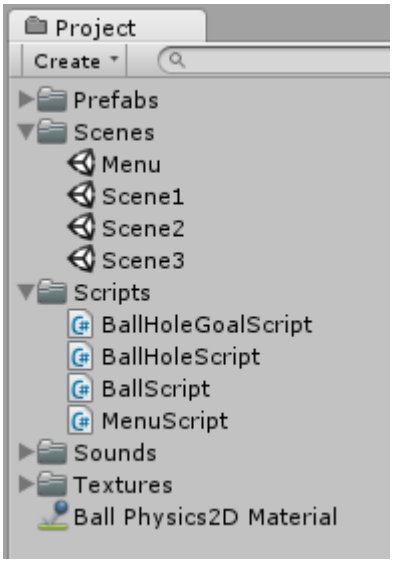
2.8 Unityn editori

Unityn editori on ohjelma missä peli pääosin tehdään. Editorissa mm. luodaan kohtaukset, lisätään hahmot, maasto, valaistus, otetaan kirjoitetut skriptit käyttöön, testataan peliä ja lopuksi julkaistaan se. Editori koostuu eri näkymistä ja opinnäytetyössä käytetyt näkymät ovat kuvassa (1). Niitä olivat projektin selain, tutkija, hierarkia, kohtaus, peli ja konsoli. Näkymiä on mahdollista siirrellä, tai sulkea kokonaan, joten editori on mahdollista muokata mieleisekseen.



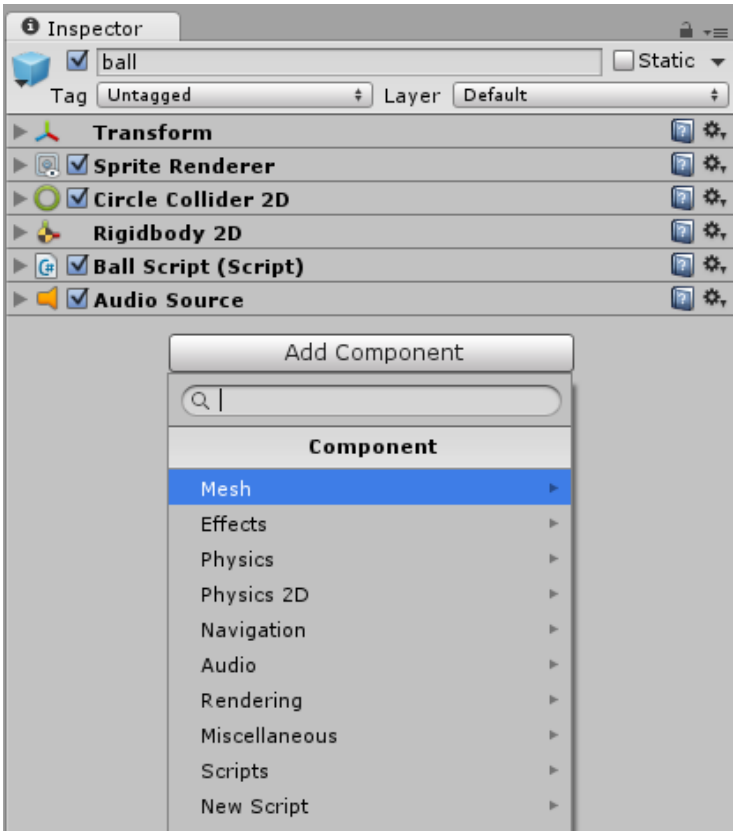
Kuva 1. Editorin päänäkymä

Kuvassa (2) olevassa projektin selaimessa näytetään kaikki käytettävissä olevat tiedostot. Niihin kuuluvat esimerkiksi äänet ja tekstuurit, mutta myös tallennetut kohtaukset.



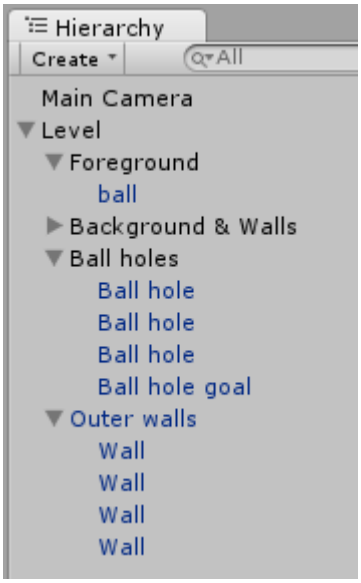
Kuva 2. Selainnäkymä

Tutkijanäkymässä näytetään ja muokataan projektin tiedostoja ja peliolioita. Peliolioille näytettäviä tietoja ovat niiden koordinaatit, komponentit ja komponenttien tiedot. Kuvassa (3) on yhden opinnäytetyössä käytetyn peliolion komponentit.



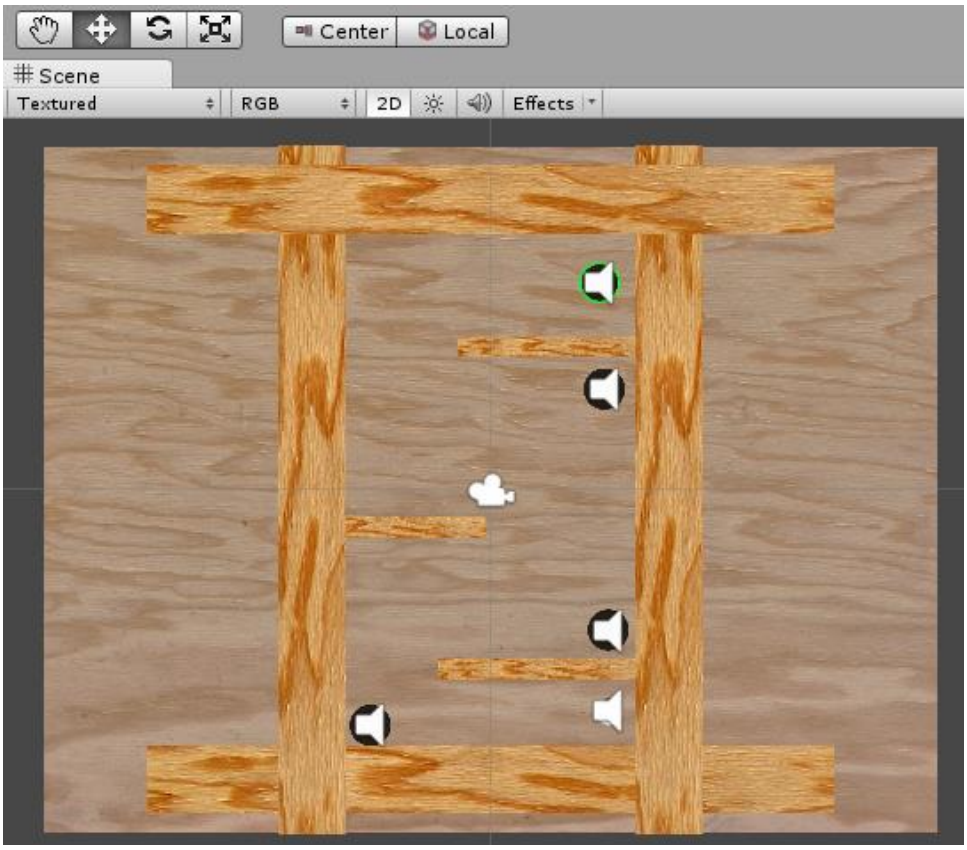
Kuva 3. Tutkijanäkymä ja komponenttien pääkategoriat

Hierarkianäkymässä näytetään kaikki kohtauksen pelioliot. Peliolioiden järjestely tyhjien ja uudelleen nimettyjen peliolioiden alle voi helpottaa työskentelyä ja näin on tehty kuvassa (4).

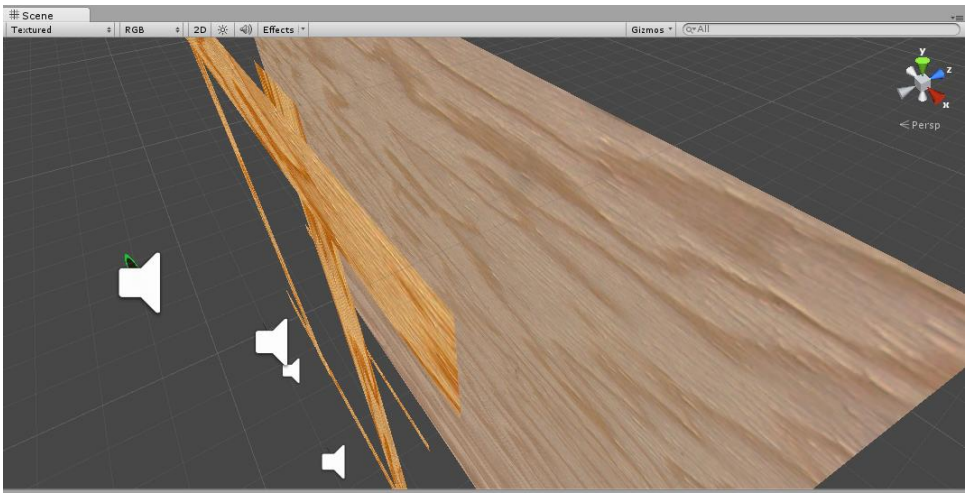


Kuva 4. Hierarkianäkymä

Kohtausnäkymässä nähdään koko kohtaus ja voidaan hiirellä siirtää peliolioita suunnitteluvaiheessa, tai myös testatessa pelin pyöriessä. Pelioloille näytetään myös niiden tunnistamista helpottavia kuvioita. Projektin alussa valitun asetuksen mukaan kohtaus näytetään kaksiulotteisessa, tai kolmiulotteisessa tilassa. Kaksiulotteisessa tilassa näkymä on oletuksena ortografinen ja kolmiulotteisessa perspektiivinen. Ortografinen näkymä säilyttää peliolioiden välisen kokosuhteen etäisyydestä riippumatta. Kaikki Unityn kohtaukset käsitellään todellisuudessa kolmiulotteisesti, ja tämä ilmenee helposti kytkemällä kaksiulotteisen tilan pois päältä. Tällöin nähdään, kuinka litteät pelioliot leijuvat ilmassa eri tasoilla. Kaksiulotteisesta ja kolmiulotteisesta näkymästä on esimerkit kuvissa (5)(6).

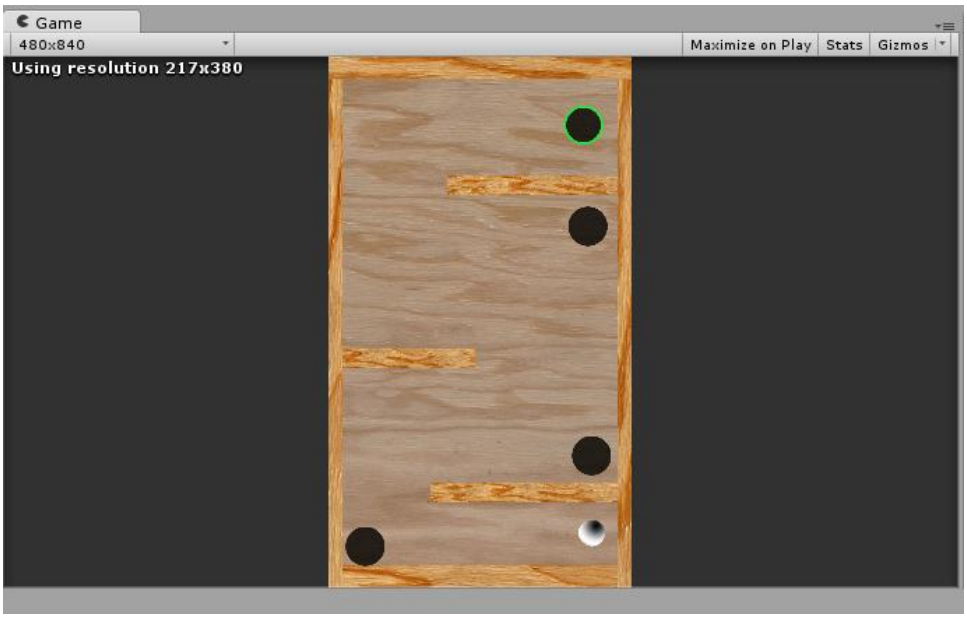


Kuva 5. Kohtausnäkymä kaksiulotteisessa tilassa



Kuva 6. Kohtausnäkymä kolmiulotteisessa tilassa

Pelinäkymässä nähdään, miltä peli oikeasti näyttää, kun otetaan huomioon kameran sijainti kohtauksessa. Kuvasta (7) voi huomata, kuinka kaikkea kohtausnäkymässä näkyvää ei enää piirretä. Varsinainen peli on myös mahdollista käynnistää ja testata näkymässä. Pelinäkymää voikin yhdessä konsolin kanssa käyttää nopeaan testaamiseen ja virheiden etsimiseen.



Kuva 7. Pelinäkymä

2.9 MonoDevelop

Skriptien kirjoittamista varten Unityn mukana asentuu ohjelmointiympäristö MonoDevelop. MonoDevelop julkaistaan avoimena lähdekoodina ja se on saatavana Windows-, OS X- ja Linux-käyttöjärjestelmille. Se on pääosin suunniteltu C#-ohjelmointikieltä sekä .NET- ja sen avoimen lähdekoodin vastike Mono-ohjelmistokomponenttikirjastoja varten, mutta tuetut kielet ja ominaisuudet vaihtelevat käyttöjärjestelmän mukaan. (MonoDevelop 2014.) Unityn skriptien kirjoittamista varten on myös mahdollista käyttää muita ohjelmointiympäristöjä, kuten Microsoft Visual Studioa.

3 WINDOWS PHONE JA SOVELLUSKAUPPA

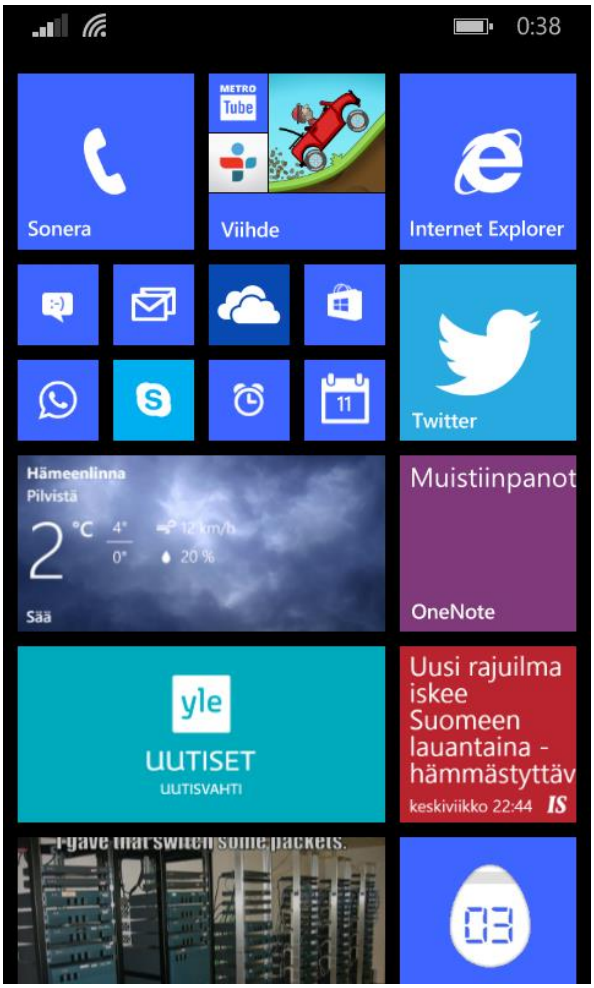
Windows Phone on Microsoftin kehittämä mobiilikäyttöjärjestelmä. Se tehtiin vuodesta 2007 asti markkinaosuuttaan menettäneen Windows Mobilen seuraajaksi ja vuoden 2014 toisella neljänneksellä käyttöjärjestelmä oli 2,5 % myyntiin menneistä älypuhelimista. (IDC 2014.)

3.1 Windows Phone

Ensimmäiset Windows Phone -laitteet julkaistiin lokakuussa 2010 ja käyttöjärjestelmän koko nimi oli Windows Phone 7. Käyttöjärjestelmä perustui edelleen Windows Mobilen käyttämään Windows CE -ytimeen, mutta käyttöliittymä uudistettiin täysin ja Windows Mobilen sovellukset eivät olleet yhteensopivia (Kolakowski 2010; Ziegler 2010). Käyttöliittymässä oleelliseen osaan nousi kotinäkymässä olevat tapahtumaruudut, jotka toimivat pikakuvakkeina, mutta voivat myös näyttää tietoja, kuten sään ja uutisia kuvassa (8) näkyvällä tavalla. Käyttöjärjestelmään julkaistiin kaksi merkittävää päivitystä, jotka mm. toivat uudemman version selaimesta ja sallivat käyttöjärjestelmän käytön heikkotehoisemmilla puhelimilla. Käyttöjärjestelmän tuki loppui lokakuussa 2014, eikä sitä käyttäviä laitteita ole mahdollista päivittää uudempaan. (Allison 2014.)

Lokakuussa 2012 Microsoft julkaisi käyttöjärjestelmän seuraavan version nimeltään Windows Phone 8. Käyttöliittymä pysyi pääosin ennallaan, mutta käyttöjärjestelmän ydin vaihtui Windows 8 käyttämään Windows NT -ytimeen (Foley 2012). Windows Phone 7 varten julkaistut sovellukset ovat yhteensopivia, mutta jos kehittäjät haluavat hyödyntää Windows Phone 8 tarjoamia uusia ominaisuuksia, heidän pitää tehdä erillinen versio Windows Phone 7 varten. Uusi versio toi mukanaan myös yrityksille suunnattuja ominaisuuksia, kuten tiedostojen salauksen. Käyttöjärjestelmä sai kolme päivitystä, joissa mm. lisättiin ruudun asennon lukitseminen, sekä tuki uudelle prosessorille ja tarkemmalle näytölle. (Allison 2014.)

Käyttöjärjestelmän uusin merkittävä versio Windows Phone 8.1 julkaistiin huhtikuussa 2014 ja se on saatavana kaikkiin Windows Phone 8 -laitteisiin (Whitney 2014). Merkittävimmät uudistukset käyttäjille olivat ilmoituskeskus, uusi selain ja virtuaalinen assistentti Cortana. Cortana on kuitenkin toistaiseksi saatavilla vain englanniksi, kiinaksi ja hindiksi. Kehittäjille mielenkiintoisinta on mahdollisuus tehdä ns. universaaleja sovelluksia, jotka toimivat Windows Phone 8.1 ja Windows 8 -laitteilla. Lähes kaikki koodi voidaan jakaa, pois lukien käyttöliittymä ja jotkin laitekohtaiset ohjelmointirajapinnat. Käyttöjärjestelmään on julkaistu yksi pienempi päivitys, jossa parannettiin eri näyttöjen tukea ja selaimen yhteensopivuutta. (Allison 2014.)



Kuva 8. Windows Phonen tapahtumaruutuihin perustuva käyttöliittymä

3.2 Windows Phone Store

Windows Phone Store on Microsoftin sovelluskauppa, jonka kautta Windows Phone -sovelluksien asentaminen, ostaminen ja päivittäminen tapahtuvat. Sovelluskaupan käyttäminen vaatii käyttäjiltä ilmaisen Microsoft-tilin hankkimisen. Maksullisten sovellusten ostamiseen tuettuja vaihtoehtoja ovat mm. luottokortit, PayPal ja lahjakortit. Kauppa on pääosin erillinen sovellustarjonnaltaan tietokoneilla ja tableteilla käytettävään Windows Store -sovelluskauppaan verrattuna. Windows Phone 8.1 mahdollistamien universaalien sovelluksien myötä kehittäjät voivat antaa yhdellä maksulla saman sovelluksen molempiin kauppoihin. Marraskuuhun 2014 mennessä Windows Store ja Windows Phone Store -sovelluskauppoihin oli julkaistu yhteensä 525 000 sovellusta. (Microsoft 2014a.)

Sovellusten kehittäminen on ilmaista, mutta niiden julkaisu vaatii kertaluonteisen maksun. Microsoft ottaa maksullisten sovellusten myynnistä saaduista nettotuloista 30 %. Sovelluksen tulee noudattaa monia Microsoftin asettamia sääntöjä ja läpäistä automaattinen tarkastus ennen julkaisua. (Microsoft 2014b.)

4 MOBIILISOVELLUKSEN JULKAISUPROSESSI

Unity tukee neljää mobiilikäyttöjärjestelmää, jotka ovat Android, BlackBerry, iOS ja Windows Phone. Kun sovellusta halutaan testata oikeilla laitteilla, tai se halutaan julkaista, niin Unityssa valitaan, mille käyttöjärjestelmälle sovellus paketoidaan. Sovelluksen pitäisi toimia tuetuilla käyttöjärjestelmillä ilman muutoksia, kunhan laitteista löytyy sovelluksen vaatimat ominaisuudet, kuten kiihtyvyysanturi. Opinnäytetyön tarkoitus oli kuitenkin keskittyä Windows Phone käyttöjärjestelmään, eikä muita laitteita ollut saatavilla, joten tätä ei testattu.

4.1 Sovelluskauppaan rekisteröityminen

Kehittäjän täytyy omistaa kehittäjäksi rekisteröity Microsoft-tili, jotta hän voi julkaista sovelluksen Windows Phonen sovelluskaupassa. Rekisteröinti ja sovelluksen julkaisu tehdään Windows Phone Dev Center -sivustolla. Kehittäjäksi rekisteröityessä annetaan yleisiä tietoja, vaaditaan sitoutumaan sovelluskehittäjäsopimukseen ja valitaan, onko tilin tyyppi yksityinen, vai yritys. Tilin tyypistä riippuen käytettävissä on joitakin lisäominaisuuksia, mutta myös vaatimuksia. Mikäli aikoo ansaita julkaisemilla sovelluksillaan rahaa, niin täytyy myös antaa tilinumero ja verotietoja. Molemmat tilit vaativat kertaluontoisen maksun, mutta opiskelijoiden on mahdollista saada yksityinen tili ilmaiseksi Microsoftin DreamSpark-ohjelman kautta (Microsoft 2014c). (Microsoft 2014d.)

Taulukko 1. Yksityisen- ja yritystilin erot (Microsoft 2014e.)

Yksityinen tili	**Yritystili**
Sovellusten joidenkin ominaisuuksien käyttö on rajoitettu	Laajempi tuki sovellusten ominaisuuksille
Ei voi julkaista työpöytäsovelluksia Windows Storessa	Voi julkaista työpöytäsovelluksia Windows Storessa
Maksaa 14€	Maksaa 75€
	Vaatii varmennuksen Symantecin kautta.
	Mahdollisuus EV-sertifioinnille
	Yrityksen on oltava maassa virallisesti rekisteröity

4.2 Sovelluksen paketointi

Sovellus tulee paketoida julkaisua varten ja tämä tehdään Microsoft Visual Studio -ohjelmalla. Koska kyseessä on Unity-projekti, pitää peli ensin paketoida sen kautta Visual Studion ymmärtämään muotoon. Tämä onnistuu helposti valitsemalla Unityn paketoinnin asetuksista kohdealustaksi Windows Phonen, sekä valitsemalla mukaan halutut kohtaukset. Kohdealustan valinnassa on kuitenkin tärkeää huomata, että Windows Phone 8 ja 8.1 ovat eri alustoja. Windows Phone 8 -projekti on oma kohteensa, mutta Windows Phone 8.1 -projekti on Windows Storen alla. Windows Phone 8.1 -projektia luodessa tulee myös valita alustan SDK kohtaan Phone 8.1,

tai Universal 8.1. SDK valinta vaikuttaa siihen, luodaanko projektista molempia sovelluskauppoja tukeva universaali projekti, vai pelkästään puhelimille tarkoitettu. Unityn luoma projekti voidaan tämän jälkeen avata ja paketoida Visual Studiossa. Luotu paketti on XAP-formaatissa, jos kyseessä on Windws Phone 8 -projekti, tai APPX-formaatissa, jos se on Windows Phone 8.1 -projekti.

Sovelluksen tulee noudattaa Microsoftin laatimia sääntöjä läpäistäkseen Microsoftin sertifioinnin ja päästäkseen sovelluskauppaan. Mikäli kyseessä on Windows Phone 8 -projekti, niin sertifioinnin läpäisyn helpottamiseksi voidaan ajaa Visual Studiossa Store Test Kit, jossa on automaattinen ja manuaalinen testi. Automaattisessa testissä tutkitaan XAP-paketin sisältöä, vaadittuja ikoneja ja ruutukaappauksia. Manuaalisessa testissä on yhteensä 61 kohtaa, jotka täytyy testata itse. Jokaiselle testille on olemassa selkeät ohjeet, mitenkä toimia ja ohjeet vaihtelevat sovelluksen tyypin mukaan. Jos projekti on Windows Phone 8.1 -laitteille, niin Store Test Kitin testejä ei ole mahdollista ajaa läpi. Sen tilalle on tullut Windows App Certification Kit, joka testaa vaatimuksia automaattisesti. Testien läpäisy ei takaa, että varsinainen sertifiointi menisi läpi. (Microsoft 2014f.)

Valmis paketti ladataan Microsoft Dev Center -sivustolle. Sovellukselle annetaan tietoja, joihin kuuluvat nimi, kategoria, hinta ja markkina-alue. Sovellus on myös mahdollista julkaista testiversiona, jolloin sovellusta ei julkaista kauppaan, mutta sen voi ladata valittujen Microsoft-tilien omistajat. Sovellus voidaan valita julkaistavaksi heti, kun sen sertifiointi on mennyt läpi, tai myöhempänä ajankohtana. Microsoftin tekemän sertifioinnin etenemistä ja mahdollisia virheitä on mahdollista seurata sivujen kautta. (Microsoft 2014g.)

5 KUULAN OHJAUS KOSKETUSNÄYTÖLLÄ JA ANTUREILLA

Toimeksiantajan vaatimus opinnäytetyöstä oli yksinkertainen Unity-projekti, josta saa käsityksen, kuinka Unitylla piirretään grafiikkaa ja luetaan mobiililaitteen antureita. Päätimme projektin olevan sovellus, jossa kuula liikkuu kahdeksikon sisällä puhelinta kallistelemalla. Tämä osuus työstä toimi myös tutustumisena Unityyn ja sitä tehdessä opittuja taitoja ja sen osia hyödynnetään projektin toisessa tuotoksessa. Lukuja seuraamalla ja soveltaen on mahdollista tehdä itse vastaava sovellus. Kuvassa (9) on lopullinen tuotos.

5.1 Grafiikan piirtäminen

Unityn dokumentaatioon tutustumisen jälkeen lähdettiin toteuttamaan ensimmäistä Unity-projektia. Kolmiulotteista peliä tehdessä on mahdollista käyttää Unityn tarjoamia yksinkertaisia 3D-malleja, kuten kuutio, kuula ja sylinteri. Projektin tulee olla kuitenkin kaksiulotteinen, joten tekeminen aloitettiin piirtämällä projektissa tarvittavat komponentit, eli kahdeksikon ja kuulan. Näiden piirtämiseen käytettiin Adobe Photoshop CS6 kuvankäsittelyohjelmaa. Unityn dokumentaatio suosittelee tekemään tekstuureista niin isoja, kuin se on mahdollista, sillä niitä on helpompi skaalata alaspäin laatu säilyttäen. Tekstuurien suurin tuettu resoluutio Unityssa on 4096x4096 pikseliä. Koska kyseessä oli kuitenkin mobiilisovellus ja malliprojekti, jossa graafinen hienous ei ole tärkeintä, niin kahdeksikosta tehtiin 1000x1000x pikseliä ja kuulasta 100 pikseliä. Luodut tekstuurit olivat hyvin yksinkertaisia, joten taiteellista silmää näiden tekemiseen ei tarvinnut. Tekstuurit olivat piirrettyä osaa lukuun ottamatta läpinäkyviä ja ne tallennettiin omiksi Photoshop-projekteiksi PSD-muotoon. Aiemmin luomaan Unity-projektiin tehtiin kansio tekstuureille ja sinne siirrettiin PSD-tiedostot, jonka jälkeen ne näkyivät Unityn editorin selainnäkymässä.

Projektia jatkettiin luomalla peliolioita. Projektissa tulisi tarvitsemaan vain kolmea pelioliota, joten niiden järjesteleminen kansioiden tapaan tyhjien peliolioiden alle ei välttämättä tekisi siitä yksinkertaisempaa. Projektia voitaisiin kuitenkin hyödyntää myöhemmin, joten loin kahdeksikolle ja kuulalle tyhjät pelioliot kansioiksi, jotka nimesin etualaksi ja taustaksi.

Kahdeksikko- ja kuulapelioliot tarvitsivat sprite renderöijä komponentin, jotta ne näkyisivät ruudulla. Tämä onnistui lisäämällä komponentin ja raahaamalla haluttu tekstuuri projektin selainnäkymästä komponentin sprite-attribuutin päälle. Komponentin muita attribuutteja olivat väri, materiaali, lajittelukerros ja järjestys kerroksessa. Näistä sprite oli automaattisesti valittu aiemmin peliolion päälle raahatuksi tekstuuriksi. Materiaalina oli Unityn oletus, jonka vuoksi spritet näkyivät normaalilla kirkkaudella ja ne eivät reagoineet peliin tuotuihin valoa luoviin peliolioihin. Kaksiulotteisissa peleissä ei kuitenkaan usein tarvita erillistä valaistusta, eikä sitä tarvita tässäkään projektissa. Jos peli kuitenkin haluttaisiin valaista erikseen, niin sille täytyisi antaa materiaaliksi varjostin, joka on tyypiltään Sprites/Diffuse. Kaksiulotteinen kuva on litteä, joten spritellä tulee olla järjestys, jotta voidaan toimia tilanteessa, jossa kaksi spriteä ovat samassa kohdassa. Järjestys voidaan määrittää sprite renderöijä komponentissa, mutta

se voidaan tehdä myös asettamalla arvo peliolion muunnoskomponentin z-akselilla. Aiemmin luomille kansioina toimiville tyhjille pelilioille asetettiin z-akseleiden arvoiksi taustalle yksi ja etualalle nolla. Z-akselilla nolla on päällimmäisin ja sen jälkeen loput nousevassa numerojärjestyksessä. On myös mahdollista käyttää negatiivisia numeroita ja nämä edeltävät järjestyksessä nollaa. Näin tekemällä tyhjien peliolioiden alla olevat pelioliot noudattavat z-akselin järjestyksessä tyhjää pelioliota.

5.2 Törmäyksen tarkastaminen

Nyt kuula ja kahdeksikko näkyivät ruudulla ja niitä pystyttiin siirtelemään kohtausnäkymässä. Seuraavaksi määritettiin pelinäkymään puhelimen kuvasuhde 4:7 ja asetettiin kahdeksikko näkymään kokonaan pelinäkymässä ja kuula sen sisällä keskellä. Kun peli yritettiin käynnistää nyt, niin huomattiin, että mitään ei tapahdu. Syynä oli se, että projektissa oli peliolioita, jotka piirtyvät ruudulle, mutta eivät muuten reagoi mihinkään. Kuulan tarvitsi liikkua laitteen kallistuksen mukaan ja sitä varten kuulan täytyi käyttää Unityn fysiikkamoottoria. Tämä onnistui lisäämällä kuulalle Rigidbody2D-komponentin ja kun peli nyt käynnistettiin, niin kuula tippui kahdeksikon läpi ja pois ruudulta saman tien. Kuulaan ja kahdeksikkoon tarvittiin törmäyksen tarkastus.

Unityn tarjoamia komponentteja törmäyksen tarkastukseen ovat ympyrä-, neliö-, polygoni- ja kulmatarkastaja. Koska kuula on pyöreä, niin ympyrätarkastaja oli looginen vaihtoehto kuulalle ja se myös osasi lisäämisen jälkeen automaattisesti muuttaa kokonsa täysin kuulan kokoiseksi. Ympyrätarkastajan sädettä ja sen keskipistettä on mahdollista muokata. Törmäyksentarkastuksen luominen kahdeksikolle ei ollutkaan niin yksinkertaista, sillä kuulan piti päästä kahdeksikon ylemmästä osasta alempaan ja ympyrätarkastajan on oltava symmetrinen ympyrä, joka sulkeutuu. Polygonitarkastajaa kokeiltiin, jonka pitäisi toimia automaattisesti, mutta tulos ei ollut miellyttävä. Tarkastajan kärkiä oli 263 ja osa niistä meni poikittain kahdeksikossa, jolloin ne olisivat estäneet kuulan kulkemisen. Kahdeksikon ulkoreunoille päädyttiin käyttämään kulmatarkastajaa. Kulmatarkastajalla voidaan piirtää itse haluttu alue luomalla kulmia ja vetämällä niistä viivoja. Siitä on mahdollista tehdä niin tarkka, kuin haluaa ja kahdeksikon ulkoreunaan käytettiin 72 kulmaa. Kahdeksikon sisäreunaan käytettiin kahta ympyrätarkastajaa.

5.3 Kiihtyvyysanturin ja kosketuksen hyödyntäminen

Kun peli käynnistettiin tässä vaiheessa, niin keskellä oleva kuula tippui alaspäin ja vieri lopulta kahdeksikon pohjalle. Koska kuulaa oli kuitenkin tarkoitus ohjata puhelinta kääntelemällä, piti kuulan Rigidbody2D-fysiikkakomponentilta kytkeä pois painovoima ja kirjoittaa skripti, joka lukee puhelimen kiihtyvyysanturia. Unityn dokumentaatiota tutkimalla löydettiin Unityn API -dokumentaatiosta Input-luokka, jolla on staattinen muuttuja acceleration. Muuttujan arvot esitetään jokaiselle akselille G-voimana, jonka arvo on jotain 1 ja -1 välillä. Mikäli puhelin on pystyssä siten, että kotinäppäin on alhaalla, niin y-akselin arvo on -1 ja jos se kään-

netään ylösalaisin, niin y-akselin arvo on 1. Kuulan liikkumisen toteutettiin käyttäen Transform-luokan translate-metodia, jolla voidaan liikuttaa mitä tahansa pelioliota. Kuula saatiin liikkumaan tällä, mutta ongelmaksi muodostui, kuinka se ei ottanut huomioon fysiikkamoottoria, vaan liikkui suoraan juuri sinne, minne se ohjattiin. Kuula meni välillä läpi törmäyksentarkastajista, eikä siihen voinut vaikuttaa fysiikkamoottorin kautta esimerkiksi massaa muuttamalla. Dokumentaatiosta löydettiin Rigidbody2D-luokasta metodi, jossa käytetään Unityn fysiikkamoottoria voiman antamiseen kuulalle. Sitä käyttäen kuula pysyi halutulla alueella ja koodista tuli myös yksinkertaisempi, joka näkyy esimerkissä (1).

```
Vector2 suunta = Vector2.zero;

void Update()
{
    suunta.x = Input.acceleration.x;
    suunta.y = Input.acceleration.y;
}

void FixedUpdate()
{
    rigidbody2D.AddForce(suunta);
}
```

Esimerkki 1. Anturien arvojen lukeminen ja voiman antaminen kuulalle

Unitysta löytyy tuki usean sormen kosketukselle, mutta työssä käytettiin vain yhden sormen kosketusta. Ohjausta kosketuksen avulla tehdessä täytyi ottaa huomioon Unityn TouchPhase-luokan kosketuksen vaihetta kuvaavat muuttujat Began (alkoi), Moved (liikkui), Stationary (paikoillaan), Ended (loppui) ja Canceled (keskeytyi). Lopuksi kuulaa liikutetaan Rigidbody2D-luokan MovePosition-metodilla, joka ottaa huomioon törmäyksentarkastajat. Aluksi ongelmana oli, että kuulaa voitiin liikuttaa koskettamalla mihin tahansa kohtaan ruudulla. Tämä korjaantui lukemalla kosketuksen ja kuulan sijainnit, sekä tarkastamalla niiden törmäyksen. Tämän toteutuksesta alla esimerkki (2).

```
private bool kosketusAlkoi;

void Update()
{
    if (Input.touchCount == 1)
    {
        Vector3 tasoPiste = Camera.main.
           ScreenToWorldPoint(Input.GetTouch(0).position);
        Vector2 kosketusPiste = new Vector2(tasoPiste.x,
           tasoPiste.y);

        if (collider2D == Physics2D.
           OverlapPoint(kosketusPiste))
        {
            kosketusAlkoi = true;
        }

        if (Input.GetTouch(0).phase == TouchPhase.Ended)
        {
```

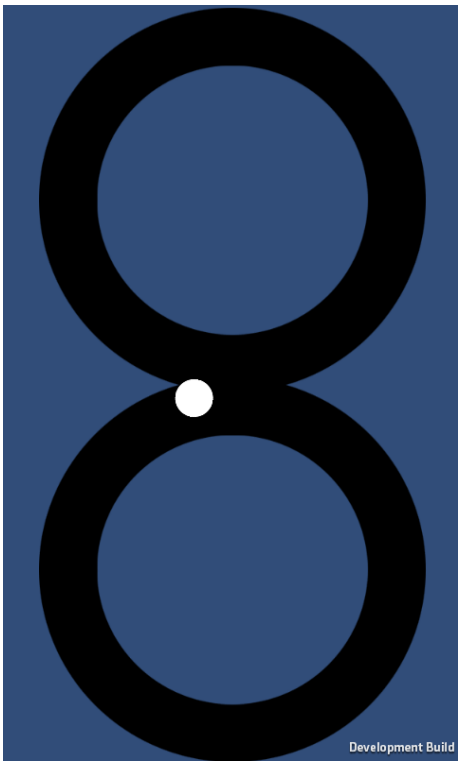
```
            kosketusAlkoi = false;
        }
    }
}

void FixedUpdate()
{
    if (kosketusAlkoi)
    {
        Vector2 kosketusDeltaPiste = Input.
           GetTouch(0).deltaPosition;
        rigidbody2D.MovePosition(rigidbody2D.position +
           kosketusDeltaPiste * Time.deltaTime);
    }
}
```

Esimerkki 2. Kosketuksen ja kuulan sijainnin vertailu sekä liikuttaminen

Pelin asennon eli kuvan pysyminen paikoillaan on oleellista varsinkin, kun ohjaus perustuu puhelimen kääntelyyn. Windows Phonessa on mahdollista lukita asento manuaalisesti, mutta tämä aiheuttaisi aluksi hämmennystä pelaajalle. Puhelin menee myös automaattisesti lukkoon, mikäli näyttöön ei kosketa, mutta tämä voidaan estää skriptin avulla. Windows Phone 8 -projektia paketoidessa oletuksena peli on lukittu puhelimen pystyasentoon ja sen pystyy vaihtamaan paketoinnin asetuksista. Windows Phone 8.1 mukana tämä kuitenkin muuttui siten, että pelin asento ei ole enää lukittu ja se täytyy tehdä kirjoittamalla skripti. Puhelimen takaisin-nappi täytyy myös ohjelmoida sulkemaan peli, sillä muuten se on turhan hankalaa. Tarvittavat määritykset esimerkissä (3).

```
void Start ()
{
            Screen.sleepTimeout = SleepTimeout.NeverSleep;

            Screen.orientation = ScreenOrientation.Portrait;
            Screen.autorotateToPortrait = false;
            Screen.autorotateToPortraitUpsideDown = false;
            Screen.autorotateToLandscapeLeft = false;
            Screen.autorotateToLandscapeRight = false;
}

void Update ()
{
            if (Input.GetKey (KeyCode.Escape)) {
                        Application.Quit ();
            }
}
```

Esimerkki 3. Ruudun lukkiutumisen ja kääntymisen estäminen sekä takaisin-napin määrittäminen



Kuva 9. Kuvakaappaus malliprojektista pyörimässä Windows Phone -laitteella

6 KUULALABYRINTTI

Opinnäytetyön toinen tuotos oli kuulalabyrinttipeli. Idea pelin tekemiseen tuli lapsuuden muistoista varsinaisen fyysisen pelin kanssa. Pelin tavoite on ohjata puhelinta kallistelemalla kuula labyrintin lähtöpisteestä maaliin. Labyrintin varrella on koloja, joita pelaajan tulee välttää, sillä koloihin tippuessa peli aloitetaan alusta. Pelissä on eri tasoja joissa labyrintin muoto, sekä kolojen sijainnit vaihtelevat. Seuraavaan tasoon päästäkseen pelaajan tulee ensin suorittaa nykyinen. Pelissä on hyvin yksinkertainen käyttöliittymä. Tämän toteuttaminen aiheutti kuitenkin pientä päänvaivaa. Yksi virhe tätä tehdessä oli laatikon ja painikkeiden koon määrittäminen pikseleinä. Tätä tekniikkaa käyttäen käyttöliittymä tulisi näyttämään hyvin erilaiselta laitteen näytöstä riippuen. Ongelma ratkaistiin käyttämällä ruudun kokoa, jota jakamalla määritettiin nappuloiden koko esimerkin (4) tavoin. Kuvassa (10) on näkyvillä pelin ensimmäinen taso ja läpäisyn jälkeen näytettävä käyttöliittymä.

```
void OnGUI()
{
    int painikeLeveys = Screen.width / 2;
    int painikeKorkeus = Screen.height / 8;

            if (GUI.Button(new Rect(Screen.width / 2
            - (painikeLeveys / 2),
            (Screen.height / 3) - (painikeKorkeus / 2),
            painikeLeveys, painikeKorkeus), "Seuraava taso"))
            {
                        Application.LoadLevel
                        (Application.loadedLevel + 1);
            }
}
```

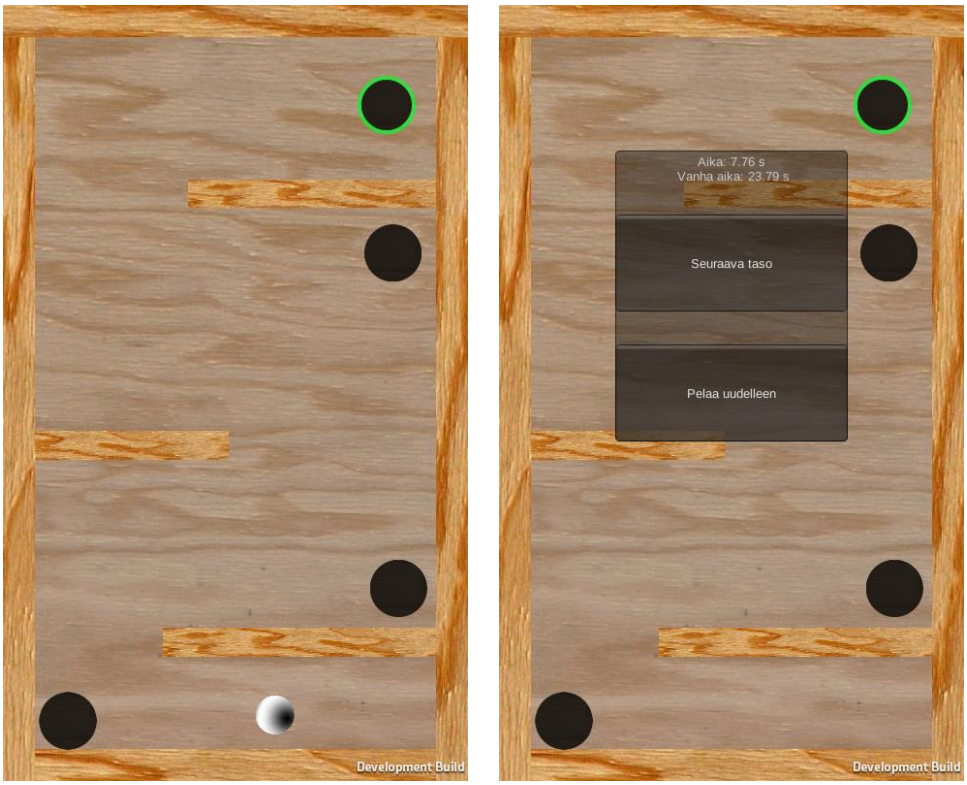
Esimerkki 4. Painikkeen piirtäminen käyttäen ruudun kokoa ja kohtauksen vaihtaminen

Osa pelin tekstuureista otettiin CGTextures-sivustolta ja muokattiin tarpeen mukaiseksi. Sivusto on tarkoitettu 3D-mallinnusta tekeville kehittäjille ja siellä on tällä hetkellä ladattavana 107347 tekstuuria. Niitä on käytetty mm. tunnettujen pelien Assassins Creed ja Killzone 2 tekemisessä. Tekstuurien lataamiseen vaaditaan ilmainen tunnus, mutta ne ovat pääosin maksuttomia joitakin variaatioita lukuun ottamatta. Lisenssi niiden käyttöön on hyvin avoin ja mahdollistaa myös kaupallisen käytön. (CGTextures 2014.)

Tason alkaessa käynnistyy ajastin, joka mahdollistaa pelaajan kilpailemaan aikansa parantamisessa. Tasokohtaiset parhaat ajat myös tallennetaan muistiin. Aikojen tallentaminen toteutettiin käyttäen Unityn PlayerPrefs-luokkaa, joka on paremminkin tarkoitettu pelaajan asetusten tallentamiseen. Se kuitenkin sopi myös tähän tarkoitukseen hyvin, sillä tallennettavaa tietoa oli vain taso ja aika. Esimerkissä (5) on näiden toteutus. Alustasta riippuen pelaajan voi olla helppo päästä käsiksi PlayerPrefs-luokan avulla kirjoitettuihin tiedostoihin ja muokata itselleen haluamansa aika. Windows Phonella tämän ei kuitenkaan pitäisi olla mahdollista ja se ei ole suuri ongelma, koska pelaaja kilpailee lähinnä itsensä kanssa.

```
private float ajastin = 0F;

void Update()
{
    ajastin += Time.deltaTime;
}

void OnTriggerEnter2D(Collider2D other)
{
    TallennaAika();
}

void TallennaAika()
{
    string nykyinenTaso = "taso:" + Application.
           loadedLevel.ToString();
    float vanhaAika = PlayerPrefs.GetFloat(nykyinenTaso);

    if (vanhaAika == 0 || ajastin < vanhaAika)
    {
        PlayerPrefs.SetFloat(nykyinenTaso, ajastin);
        PlayerPrefs.Save();
    }
}
```

Esimerkki 5. Ajastin ja ajan tallentaminen



Kuva 10. Pelin ensimmäinen taso ja valikko tason suorittamisen jälkeen

6.1 Fysiikkamoottori

Pelin kuulan ohjaus on perusteiltaan sama, kuin aikaisemmin tehdyssä malliprojektissa. Kuula ei kuitenkaan käyttäytynyt halutulla tavalla Unityn oletusasetuksia käyttäen. Se tuntui hyvin raskaalta ja kallistukseen hitaasti reagoivalta. Kuulan kankeus saatiin korjattua muokkaamalla sen massan arvoa. Kuula ei tämänkään jälkeen käyttäytynyt täysin realistisesti, sillä seinään törmätessä se pysähtyi välittömästi, eikä pomppinut ollenkaan. Kuulan törmäyksentarkastajakomponenttiin tarvitsi asettaa materiaaliksi Physics Material 2D, joka mahdollisti peliolion kitkan ja pomppivuuden säätämisen.

6.2 Äänien käyttäminen

Pelissä käytetyt äänet saatiin Freesound-sivustolta ja itse nauhoittamalla. Freesoundsista otettuja ääniä olivat yleisön taputus ja viheltäminen. Lisenssi näissä oli public domain, eli tekijät luopuivat kaikista oikeuksistaan teoksiinsa. Taputuksien toistaminen maaliin päästessä oli helppo toteuttaa, mutta kuulan rullaaminen pelilaudalla aiheutti pientä päänvaivaa. Kuulan liikkuessa äänen täytyisi toistua täydellisesti, mutta kuitenkin lakata toistumasta nopeasti kuulan pysähtyessä. Jatkuvasti toistuvan äänen kanssa pitää myös tarkastaa esimerkin (6) tavoin, että ääntä ei toisteta monta kertaa päällekkäin. Välineet äänien nauhoittamiseen ja taidot niiden käsittelyyn eivät olleet parhaasta päästä, joten äänien hienosäätäminen jätettiin myöhemmin tehtäväksi. Peliin kuitenkin saatiin yksinkertaiset ääniefektit tekemään siitä miellyttävämmän.

```
if (rigidbody2D.velocity.magnitude >= 0.3)
{
    if (!audio.isPlaying)
    {
        audio.Play();
    }
}
```

Esimerkki 6. Äänen toistaminen kuulan saavuttaessa tietyn nopeuden

6.3 Jatkokehitys

Peli on tarkoitus saada tulevaisuudessa julkaistua Windows Phone Storessa. Olisi myös kiinnostavaa kokeilla, kuinka hyvin Unity käytännössä kääntää projektin Androidille ja julkaista se myös Google Play -sovelluskaupassa Tätä varten peli tulee kuitenkin saada viimeisteltyä.

Pelissä on sinänsä toimivat grafiikat, mutta ne ovat kuitenkin melko karua katseltavaa ja vaihtelevuutta ei ole. Huonot grafiikat eivät tee hyvästä pelistä huonoa, mutta ulkoasulla on varmasti vaikutusta ensivaikutelmaan. Pelistä voisi toteuttaa myös kolmiulotteisen version. Pelin muodot koostuvat ympyröistä ja nelikulmioista, joten visuaalista silmää siihen ei erityisemmin tarvittaisi. Ääniefektit ovat yksinkertaisia ja kaipaavat myös parantelua ja lisää variaatioita. Osuttuaan koloon kuula katoaa, mutta siihen voitaisiin tehdä animaatio, jossa kuula siirtyy koloon.

Tasoja on mahdollista pelata uudelleen ja yrittää parantaa läpäisyyn kuluvaa aikaa. Olisi kuitenkin paljon mielenkiintoisempaa, jos parhaat ajat tallennettaisiin palvelimelle, jolloin pelaaja voisi kilpailla suoraan muiden kanssa. Tämän toteuttamisessa ongelmaksi luultavasti tulisi palvelimen hankkiminen. On olemassa palveluita, jotka tarjoavat ilmaisen palvelimen ilmoitettuun käyttörajaan saakka, mutta on hankalaa etukäteen arvioida, kuinka moni peliä tulee pelaamaan. Tätä olisi toki mahdollista yrittää rahoittaa lisäämällä peliin mainoksia, tekemällä siihen maksullisia tasoja, tai asettamalla sen kokonaan maksulliseksi. Pelissä tulisi olla mahdollista katsoa aikoja päävalikon kautta ja valita suoraan haluamansa taso. Aikojen vertailussa hieno ominaisuus olisi rallipeleistä tuttu kummitus, eli pelaaja näkisi toisen pelaajan kuulan liikkuvan pelatessaan.

Peliin mietittiin useita lisäyksiä pelattavuuteen, mutta niitä ei kuitenkaan toteutettu. Yksi näistä oli kuulan hyppääminen esteen yli puhelinta nopeasti nostamalla. Tässä olisi kuitenkin potentiaalinen ongelma, että nopeassa nostoliikkeessä puhelin voisi lentää kädestä. Toinen oli labyrintin pimentäminen, jolloin pelaaja näkisi tasosta vain pienen alueen kuulan ympärillä. Tämä toisi peliin lisää haastetta. Tasoja pelissä on tällä hetkellä vain muutama, mutta projektin valmisoliot luotiin siten, että uuden tason tekeminen ei kestä muutamaa minuuttia enempää.

7 YHTEENVETO JA JOHTOPÄÄTÖKSET

Unity tarjoaa pelinkehittäjille varsin laajat ominaisuudet. Työssä ei tarvinnut hakea mitään erillisiä lisäpalikoita, vaan kaikki tarvittava oli Unityssa mukana. Unityn dokumentaatio on hyvin tehty ja täynnä esimerkkejä, joiden avulla pääsee hyvin liikkeelle. Hakukoneita käyttämällä löytyy myös paljon tietoa ja apua ongelmiin. Mielestäni parhaat asiat Unityssa ovat sen ilmainen lisenssi ja laitetuki. En lähtenyt käytännössä tutkimaan, kuinka hyvin se toimii. Uskon sen kuitenkin toimivan hyvin, sillä ainakaan Windows Phonelle paketoidessa ei tarvinnut tehdä mitään erityistä alustaan liittyvää.

Windows Phone on vielä nuori käyttöjärjestelmä ja sovelluksia sen kaupassa ei ole niin paljon, kuin kilpailijoilla. Tässä on mielestäni hyvä markkinarako erottua, sillä samankaltaisella sovelluksella ei ole niin montaa kilpailijaa. Microsoftin suunta käyttöjärjestelmien suhteen on selkeä ja se on niiden yhdistäminen. Windows Phone 8.1 oli askel tätä kohti, mutta ehkä seuraavassa versiossa päästään täysin tavoitteeseen. Suosittelen kaikkia vähänkään kiinnostuneita hankkimaan kehittäjätunnukset, ainakin jos heillä on käytössä DreamSpark. Sillä saa ilmaisen elinikäisen julkaisuoikeuden molempiin Windows-sovelluskauppoihin, eikä se myöskään ilman DreamSparkia maksa enää paljon.

Työ ei ollut kovin syvällinen katsaus pelinkehitykseen tai Unityyn. Pelinkehityksen vaiheista varsinainen suunnittelu, testaaminen ja niihin liittyvä dokumentointi jätettiin pois. Unityn puolella keskityttiin hyvin vahvasti kaksiulotteisen pelin kehittämiseen, vaikka Unity alun perin suunniteltiin kolmiulotteisia pelejä varten. Mielestäni kaksiulotteinen peli on kuitenkin hyvä lähtökohta aloittelijoille ja siinä kuitenkin oppii asioita, jotka pitäisi osata myös kolmiulotteisia pelejä tehdessä.

LÄHTEET

Allison Michael. 2014. A History of Windows Phone : The Road to Threshold. Viitattu 11.12.2014. http://wmpoweruser.com/a-history-of-windows-phone-the-road-to-threshold/

BBC. 2012. Game sales surpassed video in UK, says report. Viitattu 1.12.2014. http://www.bbc.com/news/technology-17458205

Brodkin Jon. 2013. Dice. How Unity3D Became a Game-Development Beast. Viitattu 3.11.2014. http://news.dice.com/2013/06/03/how-unity3d-become-a-game-development-beast/

CGTextures. 2014. CGTextures Memberships. Viitattu 6.12.2014. http://www.cgtextures.com/signup.php

Foley Mary Jo. 2012. Microsoft's Windows Phone 8 finally gets a 'real' Windows core. Viitattu 10.12.2014. http://www.zdnet.com/article/microsofts-windows-phone-8-finally-gets-a-real-windows-core/

Gartner. 2013. Gartner Says Worldwide Video Game Market to Total $93 Billion in 2013. Viitattu 1.12.2014. https://www.gartner.com/newsroom/id/2614915

IDC. 2014. Smartphone OS Market Share, Q2 2014. Viitattu 24.11.2014. http://www.idc.com/prodserv/smartphone-os-market-share.jsp

Marketwired. 2013. Unity Reveals 2D tools. Viitattu 3.11.2014. http://www.marketwired.com/press-release/Unity-Reveals-2D-Tools-1825422.htm

Microsoft. 2014a. Microsoft by the Numbers. Viitattu 24.11.2014. http://news.microsoft.com/bythenumbers/ms_numbers.pdf

Microsoft. 2014b. App Developer Agreement. (Viitattu 24.11.2014). http://msdn.microsoft.com/library/windows/apps/hh694058.aspx

Microsoft. 2014c. Windows Store and Windows Phone Dev Center Access. Viitattu 10.12.2014. https://www.dreamspark.com/Student/Windows-Store-Access.aspx

Microsoft. 2014d. Registration info. Viitattu 10.12.2014. http://msdn.microsoft.com/en-us/library/windows/apps/jj206719.aspx

Microsoft. 2014e. Account types, locations, and fees. Viitattu 10.12.2014. http://msdn.microsoft.com/en-us/library/windows/apps/jj863494.aspx

Microsoft. 2014f. Create an app package. Viitattu 10.12.2014. http://msdn.microsoft.com/en-us/library/windows/apps/hh975357.aspx

Microsoft. 2014g. Upload and describe your package(s). Viitattu 10.12.2014. http://msdn.microsoft.com/en-us/library/windows/apps/jj206723.aspx

MonoDevelop. 2014. FAQ. Viitattu 10.12.2014. http://www.monodevelop.com/help/faq/

Nicholas Kolakowski. 2010. Microsoft Explains Windows Phone 7 Lack of Compatibility. Viitattu 10.12.2014. http://www.eweek.com/c/a/Mobile-and-Wireless/Microsoft-Explains-Windows-Phone-7-Lack-of-Compatibility-588900/

Unity Technologies. 2009. Unity Technologies Launches Version 2.6 of Its Platform and Makes Unity Freely Available. Viitattu 3.11.2014. http://web.archive.org/web/20130308073717/http:/unity3d.com/company/news/unity2.6-press.html

Unity Technologies. 2014a. Viitattu 3.11.2014. http://unity3d.com/public-relations

Unity Technologies. 2014b. Viitattu 3.11.2014. http://unity3d.com/unity/multiplatform

Unity Technologies. 2014c. Viitattu 3.11.2014. http://unity3d.com/unity/workflow/integrated-editor

Unity Technologies. 2014d. Viitattu 3.11.2014. http://unity3d.com/unity/quality/physics

Unity Technologies. 2014e. Viitattu 3.11.2014. http://unity3d.com/legal/eula

Unity Technologies. 2014.f Viitattu 24.11.2014. http://unity3d.com/unity/licenses

Unity Technologies. 2014g. Viitattu 6.12.2014. http://docs.unity3d.com/Manual/ExecutionOrder.html

Unity Technologies. 2014h. Viitattu 15.12.2014. http://docs.unity3d.com/Manual/CreatingScenes.html

Unity Technologies. 2014i. Viitattu 15.12.2014. http://docs.unity3d.com/Manual/GameObjects.html

Unity Technologies. 2014j. Viitattu 15.12.2014. http://docs.unity3d.com/Manual/UsingComponents.html

Unity Technologies. 2014k. Viitattu 15.12.2014. http://docs.unity3d.com/Manual/Prefabs.html

Unity Technologies. 2014l. Viitattu 15.12.2014. http://docs.unity3d.com/Manual/AssetWorkflow.html

Unity Technologies. 2014m. Viitattu 10.12.2014. http://docs.unity3d.com/Manual/ImportingAssets.html

Vilmantas Balasevicius. 2013. Unity 4.2 has arrived. Viitattu 3.11.2014. http://blogs.unity3d.com/en/2013/07/22/unity-4-2-has-arrived/

Whitney Lance. 2014. Windows Phone 8.1 now available to developers. Viitattu 10.12.2014. http://www.cnet.com/news/windows-phone-8-1-now-available-to-developers/

Ziegler Chris. 2010. Windows Phone 7 based on a hybrid Windows CE 6 / Compact 7 kernel? Viitattu 10.12.2014. http://www.engadget.com/2010/05/04/windows-phone-7-based-on-a-hybrid-windows-ce-6-compact-7-kerne/

Zilys Tautvydas. 2014. Introducing Universal Windows Applications in Unity. Viitattu 14.11.2014. http://blogs.unity3d.com/2014/08/07/introducing-universal-windows-applications-in-unity/