TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Arvutiteaduse instituut
Võrgutarkvara õppetool

Kahemõõtmeline mobiilimäng Unity mängumootori abil

Bakalaureusetöö

Üliõpilane: Stefan Jaanus Rüstern
Üliõpilaskood: 113088IAPB
Juhendaja: Roger Kerse

Tallinn
2014

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

---------- ----------

(*kuupäev*) (*allkiri*)

Annotatsioon

Bakalaureusetöö eesmärkideks on uurida mobiiliplatvorme, mobiilirakenduste arendamisviise, mitmeplatvormilisi arenduskeskkondi, valida eelistatud arenduskeskkond mobiilimängu arendamiseks ja arendada kahemõõtmeline mobiilimäng.

Autor käsitles uurimise käigus mobiiliplatvorme nagu iOS, Android ja Windows Phone, mobiilirakenduste tüüpe põlisrakendus ja veebipõhine mobiilirakendus, arenduskeskkondi Appecelrator, MoSync, RhoMobile, Xamarin ja mobiilimängu arenduskeskkondi Unity, Corona ja Marmalade. Arendades mobiilimängu valis autor arenduskeskkonnaks Unity mängumootori ja kirjeldas valikut, olulisemaid komponente arenduskeskkonnast ja mobiilimängu arenduskäiku.

Tulemustena tõi autor välja mobiiliplatvormide rohkuse. Lisaks on mainitud, et mobiilirakendusi, mis on arendatud kasutades platvormi spetsiifilisi arenduskeskkondi, on kulukas kohandada teistele mobiiliplatvormidele. Järeldusena on autor maininud veel, et mitmeplatvormilise arenduskeskkonna valik mobiilirakenduse arendamiseks sõltub arendaja eelistustest ja arendatavast mobiilirakendusest. Kahemõõtmelise mobiilimängu arendamiseks valis autor Unity mängumootori ja põhjendas oma valikut mängumootori mitmeplatvormilisuse toe, kommertskasutus litsentsi ja algaja mänguarendaja sõbralikkusega. Samuti hindas autor Unity mängumootori kasutamist positiivselt vaatamata mitmele puudujäägile.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 49 leheküljel, 2 peatükki, 17 joonist ja 1 tabelit.

Abstract

The purpose of the thesis is to research different mobile platforms, possibilities to develop mobile applications, multi-platform development tools, choose one favored development tool for mobile game development and develop a two dimensional mobile game.

The Author researched mobile platforms like iOS, Android and Windows Phone, types of mobile applications Native App and Mobile Web App, development tools Appecelerator, MoSync, RhoMobile, Xamarin and mobile game development tools Unity, Corona and Marmalade. For developing a mobile game the author chose Unity development tools and explained the development tool choice, explained the most important components from Unity and explained the development process.

As a result the author mentioned the abundance of mobile platforms. In addition it is mentioned that mobile applications, which are developed using platform-specific development tools, are costly to be adapted to different mobile platforms. Additionally the author has mentioned that the choice of multi-platform development tool depends on the developers preferences and the nature of the mobile application to be developed. For developing the two dimensional mobile game the author chose Unity development tools and explained his decision with Unity's multi-platform support, commerce licence and beginner mobile game developer friendliness. Despite of multiple cons the author graded the experience with Unity's game engine to be postive.

The thesis is in Estonian and contains 49 pages of text, 2 chapters, 17 figures and 1 table.

Lühendite ja mõistete sõnastik

API	***Application programming interface*** Määratud reeglistik, mille alusel rakendusprogramm kasutab operatsioonisüsteemi või rakendusprogrammi teenuseid.
CPU	***Central Processing Unit*** Kesktöötlusseade, keskprotsessor
OS	***Operating System*** Operatsioonisüsteem
RAM	***Random-Access Memory*** Muutmälu, suvapöördusmälu
GB	***Gigabyte*** Gigabait, mälumahu ühik, mis vastab ligikaudu ühele miljardile baidile
MB	***Megabyte*** Megabait, mälumahu ühik, mis vastab ligikaudu ühele miljonile baidile
SDK	***Software Development Kit*** Arendustarkvara programmipakett, mis võimaldab programmeerijal luua rakendusi konkreetsele platvormile.
iCloud	***iCloud*** Apple poot pakutva pilveteenus, kuhu Apple toodete omanikel on võimalik andmeid laadida, et need oleks kättesaadav ka teistest seadmetest.

Jooniste nimekiri

Tabelite nimekiri

„Mobile operating systems,“ Wikipedia, 20 5 2014. [Võrgumaterjal]. http://en.wikipedia.org/wiki/Mobile_operating_system. [Kasutatud 2014 5 25].

Sisukord

Sissejuhatus

Antud töö on tehtud Tallinna Tehnikaülikooli informaatika eriala bakalaureuse õppeastme lõputööna. Bakalaureusetöö esitatakse Arvutiteaduse instituuti ja on valminud 2014 aastal Tallinnas.

Bakalaureusetöö eesmärkideks on tutvuda populaarsemate mobiiliplatvormidgea, uurida levinumaid mobiilirakenduste arendamisviise ja mitmeplatvormilisi arenduskeskkondi. Uurimise tulemusena valib autor soositud mitmeplatvormilise arenduskeskkonna ja arendab kahemõõtmelise mobiilimängu.

Autori poolt püstiatud eesmärkideni jõudmiseks on kasutusel väljaannete uurimismetoodika ja arendusmetoodika.

Bakalaureusetöö esimeses osas „Mitmeplatvormiliste mobiilimängude arendus“ uuritakse populaarsemaid mobiiliplatvorme, mobiilsete seadmete mitmekülgsust, mobiilirakenduste põhitüüpe ja mitmeplatvormiliste mobiilirakenduste arendamiskeskkondi.

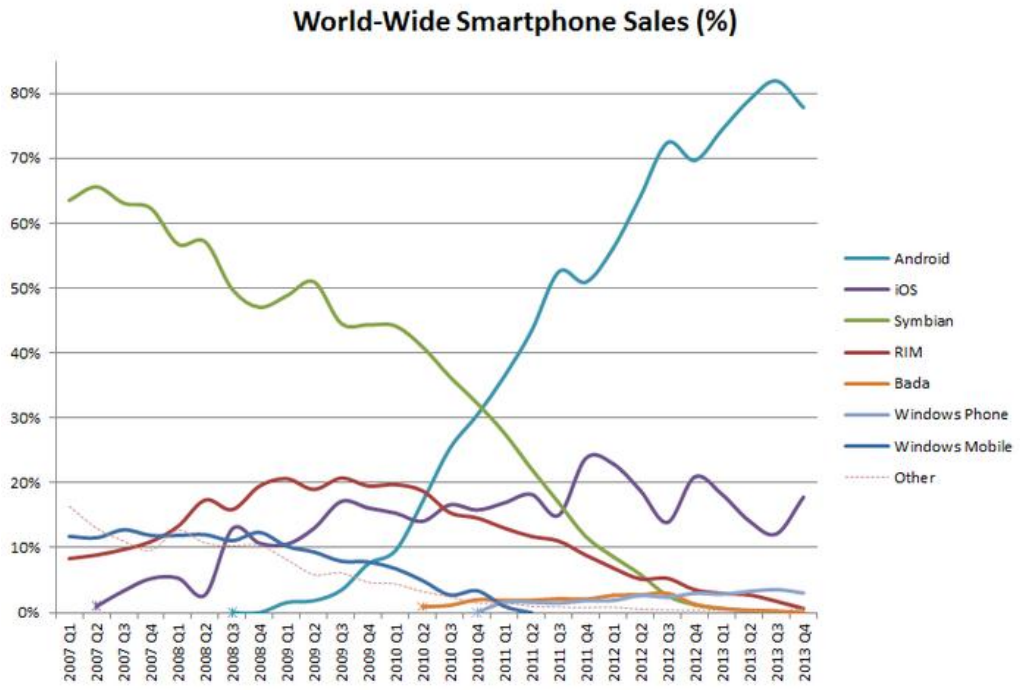
Teise osa „Kahemõõtmelise mobiilimängu loomine Unity mängumootori abil“ sisuks on mängumootori valiku põhjendus, arenduseks kasutatud mängumootori osade kirjeldus, mobiilimängu arendusprotsess ja hinnang kasutatud Unity arenduskskkonnale.

1. Mitmeplatvormiliste mobiilimängude arendus

Mitmeplatvormiliste mobiilimängude arenduse teema kerkib üha enam esile. Erinevate mobiilsete platvormide mitmekülgsus ja platvormide spetsiifilised arendusmeetodid seavad mobiilirakenduste arendajatele ette üha suuremaid takistusi. Arendajatel tuleb teha mitmeid otsuseid enne mobiilimängu arendamise algust seoses mobiilimängu mitmeplatvormilise toe ja arenduskeskkondade valikuga.

1.1 Populaarseimad mobiiliplatvormid

Kõige enim levinumad mobiiliplatvormid on Android, iOS, Windows Phone ja Blackberry. Lisaks neile eksisteerib veel mitmeid teisi vähem populaarseid mobiiltelefonidel kasutatavaid operatsioonisüsteeme – Firefox OS, Sailfish OS, Symbian, Tizen ja Ubuntu Touch OS. Platvormide populaarsust on hea võrrelda ülemaailmsete müüginäitajate alusel (vt. Joonis 1- Ülemaailmn ja Tabel 1 - Ülemaailmn). Jooniselt ja tabelist võib välja lugeda, et suurimad müüginumbrid on järjestatult Android, iOS, Windows Phone ja BlackBerry OS seadmetel. Kõigil neljal välja arvatud BlackBerry OS seadmetel on kasvavad müüginumbrid ja seega tõusev ja püsiv populaarsus. [1]



Joonis 1- Ülemaailmne nutitelefonide müük (%)

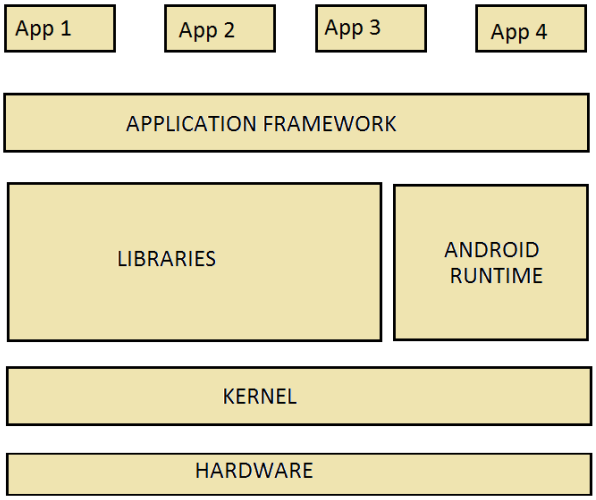
IDC: World-Wide Smartphone Shipments (Millions of Units)								
Quarter	**Android**	**iOS**	**Symbian**	**BlackBerry OS**	**Linux**	**Windows Phone**	**Other**	**Total**
2013 Q4	226.1	51.0	-	1.7	-	8.8	2.0	289.6
2013 Q3	211.6	33.8	-	4.5	-	9.5	1.7	261.1
2013 Q2	187.4	31.2	0.5	6.8	1.8	8.7	0.0	236.4
2013 Q1	162.1	37.4	1.2	6.3	2.1	7.0	0.1	216.2
2012 Q4	159.8	47.8	2.7	7.4	3.8	6.0	0.3	227.8
2012 Q3	136.0	26.9	4.1	7.7	2.8	3.6	0.0	181.1
2012 Q2	104.8	26.0	6.8	7.4	3.5	5.4	0.1	154.0
2012 Q1	89.9	35.1	10.4	9.7	3.5	3.3	0.4	152.3
2011 Q4	83.4	36.3	18.3	12.8	3.8	2.4	0.8	157.8
2011 Q3	67.7	16.3	17.3	11.3	3.9	1.4	0.1	118.1
2011 Q2	50.8	20.4	18.3	12.5	3.3	2.5	0.6	108.4
2011 Q1	36.7	18.6	26.4	13.8	3.2	2.6	0.3	101.6

Tabel 1 - Ülemaailmne nutitelefonide müük

1.1.1 Android

Android on Google poolt loodud maailma populaarseim operatsioonisüsteem mobiiliseadmetele ja tahvelarvutitele. Android-i operatsioonisüsteem on avatud lähtekoodiga ja tugineb Android-i raamistikule, mis on litsentseeritud. Antud litsents ei piira arendajatel mobiilirakenduste loomist Android-i operatsioonisüsteemile. Android-i raamistikule loodud mobiilirakenduste litsentsid kuuluvad arendajatele endile. [2]

Android-i operatsioonisüsteem on loodud tuginedes Linuxi operatsioonisüsteemile. Android-i raamistik on jaotatud neljaks tasandiks (vt. Joonis 2 – Android-i operatsioonisüsteemi tasandiline ülesehitus). [2]



Joonis 2 – Android-i operatsioonisüsteemi tasandiline ülesehitus

Esimene tasand, mis on kasutajale kõige omasem on aplikatsioonide tasand. Aplikatsioonide tasand moodustab Android-i seadmetes kasutajaliidese. Mobiilirakendusi saab mobiilses seadmes olla mitu, neil võivad olla erinevad eesmärgid ja väljaandjad. Teatud mobiilirakendused käivad Android-i operatsioonisüsteemiga kaasas, kuid enamasti koostab mobiilirakendustekogumi mobiilse seadme kasutaja ise. [2]

Järgmine tasand on aplikatsioonide raamistik. Aplikatsioonide raamistik pakub mobiilirakenduste arendajatele palju API-sid – aplikatsioonide programmeerimis liideseid. Need liidesed on vajalikud mobiilirakenduse elutsükli kujundamiseks. Antud liidesed tegelevad aplikatsiooni elutsükliga kontrollimisega, teavete kuvamisega, vaadete ja kujunduse loomisega, mobiilirakenduse väliste ressursside haldamisega ja andmete vahendamisega mobiilirakenduste vahel. [2]

Kolmandas tasandis sisalduvad teegid ja Android-i käivitusmootor. Teekide komponent hoiab endas Android-i operatsioonisüsteemile omaseid teeke, mis on kirjutatud C ja C++ keeltes. Antud teegid pakuvad sarnaselt aplikatsiooni raamistikule arendajatele võimalusi. Mõningateks kohalikeks teekideks on Surface Manager, System C Libraries, OpenGL ES Libaries ja SQLite. Antud teegid võimaldavad videote ja heli salvestamise, graafiliste objektide kuvamise, lokaalse andmebaasi kasutamise ja tuge C programmeerimiskeelele. Android-i käivitusmootor koosneb Dalvik-i virtuaalmasinast ja põhineb Java alamprogrammidel. [2]

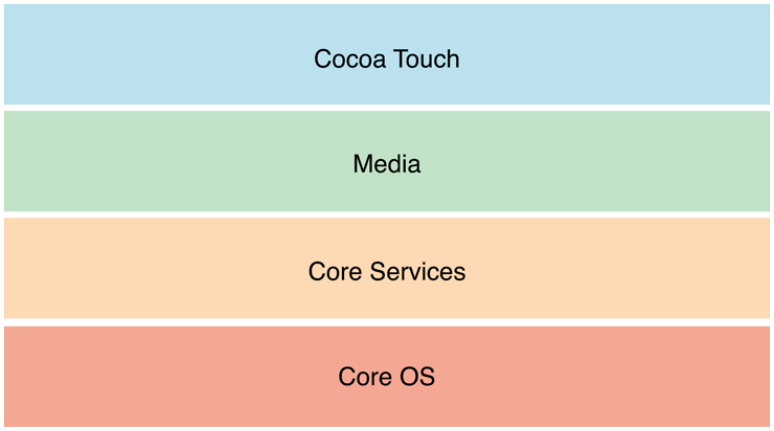
Järgmine raamistiku tasand on Kernel. Kernel on operatsioonsüsisteemi kiht, mille ülesandeks on mälu ja protsesside haldus. Kernel tasandi käitumine sarnaneb Linux Kernel 2.6 operatsioonisüsteemi käitumisele. Kernel tasand on abstraktseks kihiks mobiilse seadme riistvara ja Android-i raamistiku tarkvara vahel. [2]

Android-i mobiilirakendusi kirjutatakse Java programmeerimise keeles ja vastavalt Google enda juhenditele soovituslikult Eclipse arenduskeskkonnas. Lisaks on võimalik Android-i mobiilirakenduse teatud osasid kirjutada C ja C++ programmeerimiskeeltes kasutades selleks Android NDK-d. Asenduseks Eclipse-i arenduskeskkonnale pakub Google Android Studio arenduskeskkonda, mis baseerub IntelliJ IDEA-l. Kõik arendamiseks vajalikud juhendid ja tarkvarad võib leida Android-i ametlikult kodulehelt. Android-i aplikatsioonide arendamiseks vajalikud vahendid on tasuta kätte saadavad. [3]

1.1.2 iOS

Teine levinud mobiilne operatsioonisüsteem on iOS. iOS on arendatud Apple poolt ja jookseb sama firma poolt toodetud mobiilsetel seadmetel. iOS võrreldes Android-iga pole avatud lähtekoodiga mobiilne operatsioonisüsteem. [4]

iOS sarnaneb OS X operatsioonisüsteemiga, mis on kasutusel Apple poolt toodetud töölaua arvutitel. Sarnaselt OS X-ga on iOS-i arhitektuuril tasandiline ülesehitus (vt. Joonis 3 - iOS operatsioonisüsteemi tasandiline ülesehitus). [4]



Joonis 3 - iOS operatsioonisüsteemi tasandiline ülesehitus

Kõige ülemiseks tasandiks iOS raamistikus on *Cocoa Touch*. Antud tasand sisaldab võtme arendusraamistikke ja tehnoloogiaid iOS mobiilirakenduste arendamiseks. Nende arendusraamistike abil on võimalik defineerida mobiilirakenduste välimus, infrastruktuur, talletada faile, suhelda teiste mobiilirakendustega ja ka laadida üles mobiilirakenduste informatsiooni. Samuti peituvad tasandis mitme-protsessi ja ka puute sisendi haldustehnoloogiad. [5]

Teiseks tasandiks on meedia tasand, mis sisaldab peamisi tehnoloogiaid mobiilirakenduste multimeedia kogemuse loomiseks. Tehnoloogiate alla kuuluvad graafika, heli ja video. Nende tehnoloogiate ära kasutamiseks pakub tasand arendajatele mitmeid arendusraamistike, näiteks – ligipääsu fotodele ja videotele, audio ja video mängimis- ja salvestamisvõimalusi, heli mängimis- ja töötlemisvõimalusi, 2D ja 3D kujutiste joonistamisvõimalusi ja mängupultide tuvastamis võimalusi. [6]

Järgmine tasand on *Core Services* tasand, mis hõlmab endas põhilisi süsteemseid teenuseid mobiilirakenduste jaoks. Lisaks omab tasand toetavaid tehnoloogiaid lokatsiooni, *iCloud*-i, sotsiaalmeedia ja võrgu jaoks. Tasand pakub kõrgema taseme tehnilisi võimalusi ja kesksete teenuste raamistikke. [7]

Kõrgema taseme tehnilised võimaluste hulka kuuluvad *peer-to-peer* andmevahetus üle *bluetooth* võrgu, *iCloud*-i kasutamine, andmekaitse, aplikatsioonisisesed tehingud, SQL andmebaasi seadmesisene tugi ja XML dokumendi objektide lugemine. [7]

Kesksete teenuste arendusraamistikud pakuvad mobiilirakenduste arendajatele mitmeid võimalusi. Arendusraamistike hulka kuuluvad näiteks – kontode arendusraamistik, telefoni kontaktide arendusraamistik, reklaami toe arendusraamistik. Kaheks olulisimaks arendusraamistikuks loetakse *Core Foundation* arendusraamistikku ja *Foundation* arendusraamistikku. Need mõlemad arendusraamistikud pakuvad otsest tuge *Objective C* objektidele ja nendega teostavatele operatsioonidele. [7]

Kõige alumiseks tasandiks on operatsioonisüsteemi tuum tasand. Viimane tasand koosneb tehnoloogiatest, millele kõik ülejäänud arendusraamistikud ja tasandid on ehitatud. Põhilisteks arendusraamistikeks on *Accelerate*, *Core Bluetooth*, *External Accessory*, *Generic Security Services* ja *Security*. Nimetatud arendusraamistikud aitavad arendajatel teha digitaalsignaali töötlust, kasutada bluetooth võrgu võimalusi, suhelda mobiilse seadme külge

ühendatud väliste seadmetega ja kasutada andmeturvalisuse võimalusi. Alates iOS 7-st toetab operatsioonisüsteem ka 64 bitist arhitektuuri. [8]

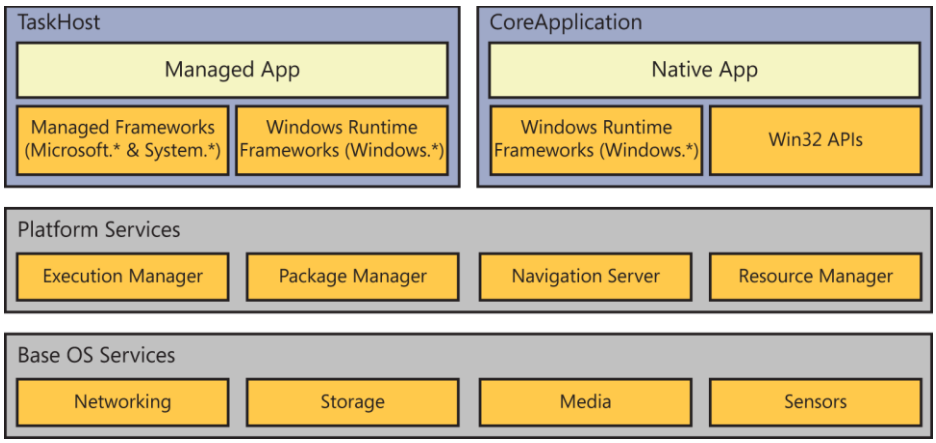
iOS operatsioonisüsteemile arendatakse aplikatsioone *Objective C* programmeerimiskeeles, kasutades selleks spetsiaalselt Xcode arendustarkvara. Xcode on arendustarkvara, mida saab kasutada OS X operatsioonisüsteemiga töötavatel töölaua arvutitel. Mac töölaua arvutite omanikele on iOS arendustarkvara tasuta kättesaadav. Apple pakub detailseid juhendeid Xcode arenduskeskkonnaga töötamiseks ja mobiilirakenduse arendamise võimalustest iOS operatsioonisüsteemile. [9]

iOS operatsioonisüsteem on teine levinuim mobiilsetel seadmetel kasutatav operatsioonisüsteem. iOS on arendatud Apple poolt luues mobiilse operatsioonisüsteemi töölaua operatsioonisüsteemi OS X sarnasena. Arendajatele pakutakse palju võimalusi ja juhendeid mobiilirakenduste arendamiseks Objective C programmeerimiskeeles. Ainsaks piiranguks on seatud arendamine Mac tüüpi töölaua arvutil.

1.1.3 Windows Phone

Windows Phone on kolmas enim kasutatud mobiilne operatsioonisüsteem. Windows Phone operatsioonisüsteemi tootjaks on Microsoft. Operatsioonisüsteem avalikustati 2010 aasta sügisel, millal see sai suure menu osaliseks oma uuendusliku kasutajaliidesega tõttu. Windows Phone operatsioonisüsteem sarnaneb Microsofti poolt töölaua arvutitele välja antud operatsioonisüsteemiga Windows 8. [10]

Samuti nagu eelnevalt mainitud Android ja iOS on Windows Phone-i arhitektuur jagatud mitmeks tasandiks (vt. Joonis 4 – Windows Phone operatsioonisüsteemi tasandiline ülesehitus). [10]



Joonis 4 – Windows Phone operatsioonisüsteemi tasandiline ülesehitus

Kõige ülemine tasand koosneb kahest erinevest mobiilirakenduste raamistikust. TaskHost raamistik toetab vanemaid Windows Phone 7 mobiilirakendusi ja alates Windows Phone versioonist 8 on kasutusel uus mobiilirakenduste raamistik CoreApplication. [10]

Mõlemad mobiilirakenduse raamistikud tuginevad operatsioonisüsteemi raamistiku järgmisele platvormi teenuste tasandile. Antud tasand pakub mobiilirakenduste raamistikule mitmeid platvormi spetsiifilisi teenuseid. Teenuste pakkujateks on neli komponenti – pakettide haldaja, teostuse haldaja, navigatsiooni server ja ressursside haldaja. [10]

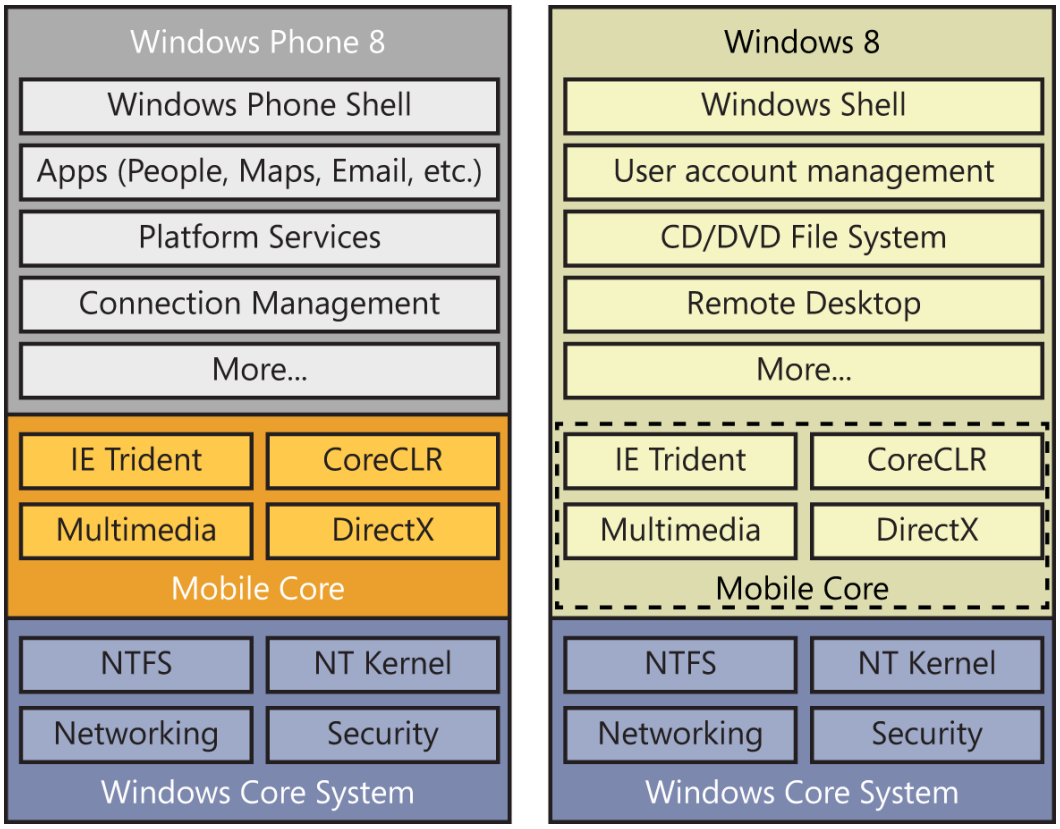
Paketihaldus komponent on vastutav rakenduste paigaldamise ja eemaldamise eest. Samuti hoiab komponent informatsiooni asukohtadest, kus ja millist mobiilirakendust kuvada ja ka informatsiooni mobiilirakenduse elutsükli kohta. [10]

Teostushaldus komponent kontrollib kogu loogikat, mis on seotud mobiilirakenduse teostus elutsükliga. Peamiselt kontrollib komponent iga mobiilirakenduse taustal jooksvaid protsesse. [10]

Navigatsiooni server haldab mobiilirakenduste vahelisi liikumisi. Ülesandeks on vastavalt kasutaja sisendile edastada teostushaldus komponendile teave, mis mobiilirakenduse käivitama peab. [10]

Ressurssihaldus komponent on vastutav, et mobiilneseade töötaks kiiresti ja oleks võimeline kasutajatele oma tegevuste kohta tagasisidet andma. Komponent mõõdab CPU ja mälu kasutust aktiivsete protsesside poolt. Kui protsesside mälu ja CPU kasutus ületab lubatut, siis piiratakse antud protsesside kasutust. Kui mõni mobiilirakenduse taustal jooksvad protsessid tekitatavad liiga suurt koormust, siis antud protsesside töö lõpetatakse. [10]

Viimaseks tasandiks on operatsioonisüsteemide baas teenuste tasand. Windows Phone 8 operatsioonisüsteem on ehitatud samale tehnoloogiale nagu Microsoft-i poolt toodetud Windows 8 töölaua arvutitele. See tasand koosneb NT kerneli, NT failisüsteemi, võrgu ja turvalisuse komponentidest (vt. Joonis 5 - Windows Phone 8 vs Windows 8). Võrreldes joonisega 4 pakuvadki antud komponendid võimalusi andmete talletamiseks ja lugemiseks, ühenduste saamiseks teiste seadmetega, meedia esitamiseks ja anduritelt info saamiseks. [10]



Joonis 5 - Windows Phone 8 vs Windows 8

Windows Phone rakendusi arendatakse .NET, C ja C++ keeltes kasutades selleks vastavalt Microsoft Visual Studio Express või Microsoft Blend Express arenduskeskkonda. Microsoft pakub mobiilirakenduste arendajatele detailseid juhendeid arenduskeskkonna kasutamise ja Windows Phone võimaluste kohta. Windows Phone mobiilirakenduste arendamise tarkvara on tasuta kättesaadav. Piiranguks on seatud, et Microsoft Express tüüpi arendustarkvara, mis on vajalik Windows Phone mobiilirakenduste arendamiseks on ainult võimalik installida Windows 8 operatsioonisüsteemiga töölaua arvutile. [10]

Kuigi Windows Phone platvorm pole nii laialdaselt levinud, kui iOS ja Android on antud operatsioonisüsteem ikkagi kasutusel miljonitel mobiilsetel seadmetel. Windows Phone 8 operatsioonisüsteem on üritatud luua Windows 8 sarnasena ja põhineb samal põhjal. Tarkvara arendajad peavad arvestama, et mobiilirakenduste arendamise tarkvara nõuab Windows 8 operatsioonisüsteemi olemasolu. Arendustarkvara on tasuta kättesaadav, aga selle kasutamiseks nõutav Windows 8 on tasuline [11].

1.2 Mobiilsete seadmete mitmekülgsus

Arendades mobiilirakendusi, mis peavad lihtsa vaevaga olema kohandatavad mitmele erinevale platvormile on peale operatsioonisüsteemi oluline arvestada ka mobiilsete seadmete eripäraga. Mobiilseid seadmeid eristavad üksteisest paljud olulised tehnilised näitajad, millega peavad mobiilirakenduste arendajad arvestama. Seades eesmärgiks, et rakendus töötaks veatult kuni üheksakümnel protsendil turul olevatest seadmetest peab mobiilirakendus olema kohandatav ligikaudu kolmesaja kolmekümne ühele erinevale mobiilsele seadmele. Mobiilsete seadmete hulka kuuluvad nii mobiiltelefonid kui ka tahvelarvutid. [12]

Alates 2008 aastast tootma hakatud mobiiltelefonide tehnilised näitajad võivad varieeruda väga palju. Resolutsioon varieerub 320 x 240-st 1280 x 720-ni. Ekraani suurus varieerub 2.55 tollist kuni 6.4 tollini. RAM-i hulk varieerub 128-st 3072 MB-ni. Ainuke tehniline erisus, mis väga ei varieeru on mobiiliaku kestvus. Mobiiliaku kestvust lühendavad kõige rohkem võrgu sirvimine ja video pildi vaatamine. [13] [14] [15] [16]

Võrreldes alates 2012. aastast tootma hakatud tahvelarvuteid võib näha samuti palju varieeruvust tehnilistes näitajates. Tahvelarvuti ekraanisuurus varieerub 14.5 cm-st kuni 34 cm-ni. Ekraani resolutsioon varieerub 800 x 600-st kuni 2560 x 1600-st. RAM-i hulk varieerub 0.5 GB-st 8 GB-ni ja aku kestvus varieerub neljalt tunnilt kaheksateistkümneni. [17]

Tahvelarvuteid eristab mobiiltelefonidest põhiliselt suurus. Suuremate mõõtmete tõttu erinevad tahvelarvutitel mitmed tehnilised näitajad mobiiltelefonidest, kuigi paremad mobiiltelefonid ei pruugi tahvelarvutitelegi alla jääda.. Sõltumata laiast mobiilsete seadmete valikutest ja tehniliste näitajate variatsioonist on oluline mobiilirakenduse arendajal teha valik ja seadistada eesmärk, millistel ja kui paljudel erinevatel mobiilsetel seadmetel peaks mobiilirakendus töötama.

1.3 Mobiilirakenduste põhitüübid

Ennem mobiilirakenduse arendamise algust seisab arendaja valiku ees, millist põhitüüpi mobiilirakendust ta arendab. Põhilisi mobiilirakenduste tüüpe on kaks – põlisrakendus ja veebipõhine mobiilirakendus, mis on ligipääsetav läbi mobiilse seadme brauseri. Mõlemal

mobiilirakenduse põhitüübil on omad eelised ja puudujäägid. Mobiilirakenduse arendaja peab tegema valiku, mis sõltub arendatavast mobiilirakendusest.

1.3.1 Põlisrakendus

Põlisrakendus (inglise k. *Native App*) on kõige levinum mobiilirakenduse tüüp, mida on harjutud mobiilsetes seadmetes kohtama. Põlisrakenduse peab standartselt mobiilirakenduste poest alla laadima ja mobiilsele seadmele installima. Pärast installimist on antud mobiilirakendus mobiilses seadmes vabalt kasutatav.

Põlisrakendusi arendatakse mobiilse seadme operatsioonisüsteemile ette nähtud programmeerimiskeeles ja arenduskeskkonnas. Lisaks on nad kättesaadavad spetsiaalselt ettenähtud mobiilirakenduste poest ja neil on võimalik kasutada mobiilsele seadmele omaseid riistvaralisi võimalusi ja API-si terves ulatuses. [18]

Põlisrakendusel on mitmeid eeliseid. Arendaja poolseks eeliseks on võimalus kasutada soovitud mobiilse seadme tehnilisi võimalusi ja mobiilsele seadmele omaseid API-sid. Kasutajad saavad sellisest võimalusest mitmekülgsemate võimalustega mobiilirakenduse. Samuti kasutades mobiilsele seadmele omaseid tehnoloogiaid saab veenduda, et arendatud mobiilirakendus töötab mobiilses seadmes kiiresti. [19]

Lisaks on põlisrakenduse levitamine lihtsam. Kasutajad saavad otsida mobiilirakendusi spetsiaalsest mobiilirakenduste poest ja selle vahendusel soovitud mobiilirakendus alla laadida. Kuna antud tüüpi mobiilirakendus peab läbima eelneva kontrolli ennem mobiilirakenduste poodi saamist saavad kasutajad olla kindlad mobiilirakenduse ohutuses ja turvalisuses. [19]

Antud mobiilirakenduse tüübil eksisteerib ka puudusi. Arendajal on kulukas tagada põlisrakenduse kompetents erinevate mobiilsete seadmetega. Sellise kompetentsi tagamiseks peab arendaja kohandama koodi kõigi erinevate mobiilsete seadmete jaoks. Soov antud kompetentsi järgi tõstab ka mobiilirakenduse tööshoidmise kulutusi. [19]

Kuna antud tüüpi mobiilirakendusi peab edastama kasutajateni läbi mobiilirakenduste poe tekitab antud kohustus mitmeid komplikatsioone arendajate jaoks. Mobiilirakenduste vastuvõtmine mobiilirakenduste poodi võib kujuneda pikaks ja aeganõudvaks protsessiks arendaja jaoks, mis ei pruugi iga kord esimesel katsel õnnestuda. Samuti ei garanteeri mobiilirakenduse saadavus mobiilirakenduste poes selle menu kasutajate seas. Põlisrakenduse

uuendamine ei ole kohustuslik kasutajatele, seega arendajad ei saa mobiilirakendust uuendades kindlad olla, kas arendatud uuendus jõuab mobiilirakenduse kasutajani. [19]

1.3.2 Veebipõhine mobiilirakendus

Veebipõhine mobiilirakendus (inglise k. *Mobile Web App*) on teist põhitüüpi arendatav mobiilnerakendus. Need mobiilsed rakendused ei ole installeeritud otse mobiilsesse seadmesse, vaid on ligipääsetavad läbi mobiilse seadme brauseri. Veebipõhistel mobiilirakendustel on võrreldes põlisrakendustega mitmeid erinevusi.

Veebipõhine mobiilirakendus on ligipääsetav kasutades mobiilse seadme brauserit. Ligipääsuks läbi mobiilse seadme brauseri on vaja eelnevalt teada veebipõhise mobiilirakenduse asukohta. Antud mobiilirakendused asuvad serveris ja on arendatud mitmetele operatsioonisüsteemidele korraga. Veebipõhiseid mobiilirakendusi arendatakse kasutades HTML, Javascript ja CSS programmeerimiskeeli. [18]

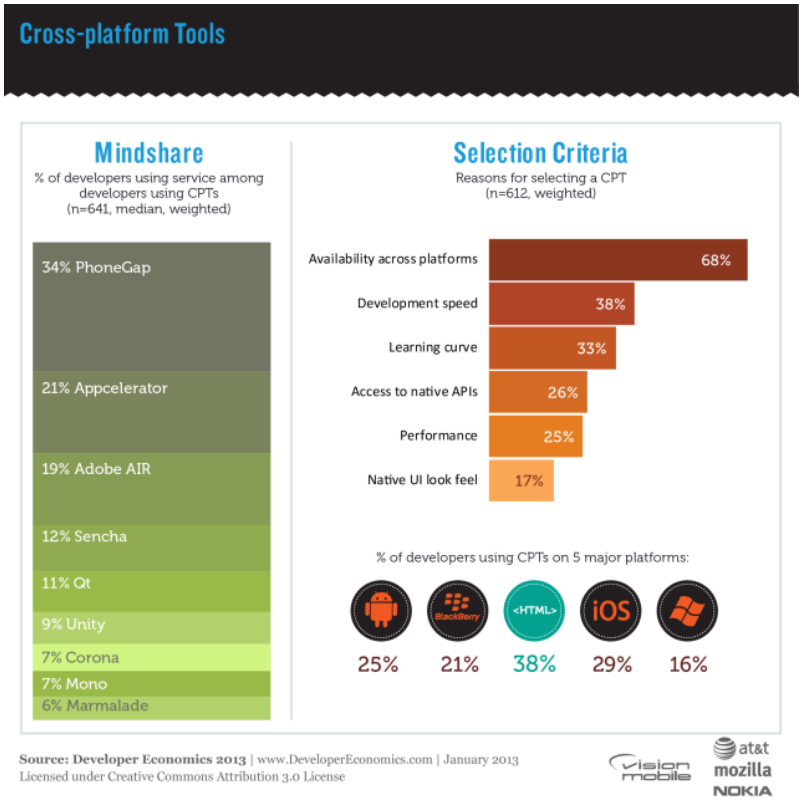
Veebipõhise mobiilirakenduse arendamisel on mitmeid plusse. Üheks eeliseks on mobiilirakenduse asukoht serveris. Serveris asuv koodibaas annab mobiilirakenduse erineva versioonide suhtes eelise. Serveril läbiviidud mobiilirakenduse muudatus jõuab otseselt kõigini mobiilirakenduse kasutajateni, võrreldes põlisrakendustega, kus uue versiooni saamiseks on vaja kasutajal teha oma mobiilses seadmes mobiilirakenduse uuendus. Veebipõhist mobiilirakendust ei ole vajalik lasta kontrollida mobiilirakenduste poel. Kontrolli puudumatus annab arendajale vabaduse mobiilirakenduse levitamise üle. [19]

Võrreldes põlisrakendusega eksisteerib ka veebipõhise mobiilirakenduse arendamisel puudujääke. Veebipõhiste mobiilirakenduste kohandamine mitmele erinevale brauserile on kulukas. Mitme erineva brauseri toetamine suurendab arendamise ja mobiilirakenduse töö säilitamise kulutusi. Veebilehtedele omane kasutajatepõhine käitumismalli jälgimine on raskendatud, sest kasutajad võivad veebipõhise mobiilirakenduse poole pöördumiseks kasutada mitmeid erinevaid brausereid. Veebipõhiste mobiilirakenduste üles leidmine kasutajate poolt on ebatraditsiooniline, sest need pole süstemaatiliselt mobiilirakenduste poes esindatud. Samuti on puudulik perioodiline kontroll veebipõhiste mobiilirakenduste üle. Kontrolli puudumise tõttu pole kasutajatele tagatud mobiilirakenduse ohutus ja turvalisus. Lisaks eelnevale on veebipõiste mobiilirakendustel piiratud mobiilse seadme riistavara ja API-de kasutus. Antud piiranguga peavad arendajad arvestama veebipõhist mobiilirakendust arendades. [19]

1.4 Põlisrakenduse arendus mitmele platvormile

Traditsiooniliselt arendatakse põlisrakendusi operatsioonisüsteemile ettenähtud arenduskeskkonnas ja vastavas programmeerimiskeeles. See lähenemine teeb arendajatel raskeks oma mobiilirakenduse tööle saamine mõnes teises mobiilses seadmes, mis töötab teise operatsioonisüsteemiga. Arendajate elu lihtsustamiseks on arendatud arenduskeskkondi, mille eesmärgiks on teha võimalikuks ühe koodibaasiga põlisrakenduse tööle saamine mitmes teises mobiilses seadmes. [20]

Mitmeplatvormiliste mobiilirakenduste arendajate seas, kes kasutavad arendamiseks mitmeplatvormilisi arenduskeskkondi, läbi viidud uuring toob välja enimkasutatavad mitmeplatvormilised arenduskeskkonnad (vt. Joonis 6 - Mitmeplatvormiliste arenduskeskkondade kasutus). Antud nimekirjast on mõeldud põlisrakenduste loomiseks arenduskeskkonnad nagu Appcelerator, Qt, Unity, Corona, Mono ja Marmalade. [21] Lisaks neile nimetatule on levinud ka MoSync, Rhomobile ja Xamarin. [22] Antud nimekirjast pööraks tähelepanu keskkondadele, mis ei ole mõeldud peamiselt mobiilimängude arendamiseks ega veebipõhiste mobiilirakenduste arendamiseks.



Joonis 6 - Mitmeplatvormiliste arenduskeskkondade kasutus

1.4.1 Appcelerator

Appcelerator toodab arendusplatvormi, milleks on Titanium. Titanium pakub võimalust arendada ühe koodibaasiga mobiilirakendusi mitmele erinevale mobiilsele seadmele. Arendamise programmeerimiskeeleks on Titaniumis Javascript ja arenduskeskkonnaks on Eclipse-i tüüpi rakendus. [23]

Titaniumi kasutamisel on arendajad toonud välja mitmeid eeliseid ja puudujääke. Välja on toodud, et Titanium pakub arendajatele võimalusi kasutada üle kolmesaja erineva mobiilsele seadmele iseloomuliku API. Laialdase API valikuga on võimalik arendada täielikult riistvarapõhiseid mobiilirakendusi. Titanium pakub ka andmete talletamise võimalusi andmepilves või sihtseadmes endas. [24]

Eelisteks loetakse veel, et Titaniumiga on esimeste töötavate prototüüpide valmis saamine kiire protsess. Tulenevalt kasutatavast programmeerimiskeelest Javascript on lihtne arendada mobiilirakenduses suhtlust veebiteenuste ja JSON andmete formaadis. Lisaks võimaldab Titanium suure ühise koodibaasiga saada mobiilirakenduse tööle nii Android-i ja iOS-i mobiilsetel seadmetel. Üheks suurimateks eelisteks loetakse veel Titaniumi suurenevat kasutatavust arendajate seas. Samuti võimaldab Titanium arendajatel müüa oma arendatud mobiilirakenduse erinevaid valmis osi Titaniumi poolt pakutavas veebipoes. [25]

Eeliste kõrval on välja toodud ka puudujääke. Arendajad on märganud, et lihtsalt on tõesti prototüüp mobiilirakendusest valmis saada, aga mobiilirakenduse tõusva keerukuse korral tõuseb arendamise keerukus proportsionaalselt kiiremini. Titaniumi poolt pakutav API mobiilirakenduse sisesteks transaktsioonideks on ebastabiilne. API ebastabiilsus sunnib arendajad kasutama teistsugust mobiilirakenduse teenimismudelit. [25]

Arenduskeskkonna miinusteks on veel väike arenduskeskkonna sisseehitatud programmeerimise abi. Arendaja peab lihtsamate küsimuste korral pöörduma dokumentatsiooni poole. Lisaks on vigade avastamine (inglise k. debug) ainult arenduskeskkonna tasulise versiooni osa. [26] Samuti on Titaniumi arenduskeskkonnal tarkvara kompileerides tendents muudatusi koodibaasis ignoreerida. Lisaks on võimalik, et Titanium lõpetab konsooli info kirjutamise. [25]

Vaatamata puudujääkidele kogub Titanium arenduskeskkonnana populaarust. Arenduskeskkonna valikul tuleb tasulise ja tasuta Titaniumi arenduskeskkonna

funktsionaalsuse erinevust arvesse võtta. Titanium-i valik arenduskeskkonnaks sõltub arendatavast mobiilirakendusest ja arendaja eelistustest.

1.4.2 MoSync

MoSync on avatud lähtekoodika arenduskeskkond, mis pakub võimalust ühe koodibaasiga arendada mitmeplatvormilist mobiilirakendust. MoSync toetab kompileerimist üheksasse erinevasse platvormi. Arenduskeelte valikuteks on C ja C++ või HTML5 ja Javascript. Antud programmeerimiskeeli saab kasutada paaris või hübriid tüüpi mobiilirakenduse korral kõiki korraga. Arenduskeskkonnaks on MoSync-il Eclipse-i taoline rakendus. [27]

Eelisteks loetakse MoSync platvormi kasutamise juures suurt mobiilsetes seadmete tuge, mille hulka kuuluvad iOS, Android ja Windows Phone. Mitme erineva programmeerimiskeele valik annab arendajatele võimaluse arendada mitmeid tüüpe mobiilirakendusi. Kasutades kõiki eelnevalt nimetatud programmeerimiskeeli on võimalik luua veebipõhiseid mobiilirakendusi ja põlisrakendusi. MoSynci-s on võimalik luua mobiilirakenduse kasutajaliides kasutades veebipõhised programmeerimiskeeli HTML5, Javascript ja CSS, ning antud kasutajaliidesele funktsionaalsus C ja C++ programmerimiskeeltes. [28]

Arendajad on välja toonud, et põhilistest C ja C++ programmeerimiskeelte teekidest, mida MoSync sisaldab jääb väheseks. Tulemusena on arendajad sunnitud ise puuduolevad alamprogrammid koostama. Samuti Javascriptiga mobiilse seadme tehnilise funktsionaalsuse ja API-de kasutamise tehnoloogia on ebastabiilne. Miinuseks loetakse veel MoSync-i litsentseeritust. Arendatud mobiilirakenduste kommerts – ja vabakasutuse jaoks on erinevad litsentsid. [28]

Kuigi MoSync ei ole kõige populaarsem mitmeplatvormiline arendusplatvorm on ta üks eelistatumatest. Eelistuse põhjusteks on olulisemate mobiili operatsioonisüsteemide tugi ja mugav ja mitmekülgne Eclipse-taoline arenduskeskkond. Litsentseeritust on puudujäägiks, kuid arenduskeskkonna vabakasutus on võimalik mitte kommertslikel eesmärkidel.

1.4.3 RhoMobile

RhoMobile pakub Rhodes arenduskeskkonda mitmeplatvormiliseks mobiilirakenduste arendamiseks. Rhodes on avatud lähtekoodiga arendusraamistik, milles arendamiseks

kasutatakse Ruby programmeerimiskeelt. Rhodes-i arenduskeskkond pakub tuge peamistele mobiilsetele operatsioonisüsteemidele iOS, Android ja Windows Phone. [22]

Rhodes-i arenduskeskkonna kasutamisel on mitmeid plusse. Rhomobile pakub arendajatele erinevaid rakendusi, mis pakuvad lisavõimalusi mobiilirakenduste arendamiseks. Nende lisade alla kuuluvad RhoHub, Rhom data mapper, Rhoelements ja Rhoconnect. Antud lisad võimaldavad kompileerida mobiilirakendusi andmepilves, mis eemaldab vajaduse SDK-de järele kohalikus arvutis. Veel lihtsustavad eelnimetatud lisad andmete talletamist, andmete kättesaamist ja rohkemate platvormide toe tagamist. Samuti on kasutades Rhodes arenduskeskkonda võimalik kasu haarata sihtmobiilse seadme tehnilistest lahendustest ja API-dest. [29]

Rhodes-i arenduskeskkonnas loodud mobiilirakenduse iOS ja Androidi koodibaas ühtib hinnanguliselt üheksakümne viie protsendi ulatuses, aga see näitaja on palju väiksem Windows Phone ja Blackberry korral. Samuti on suureks miinuseks lokaalse kompileerimise süsteemi keerukas ülesseadmine. Kohalike SDK-de ülesseadmine on keeruline protsess, mille tõttu soovitatakse kasutada Rhodes-i andmepilves asuvat kompileerimisrakendust. Viimaseks puuduseks on suured litsentsi maksud arendatud mobiilirakenduste kommertskasutamiseks. [29]

Rhodes-i arenduskeskkond on hea mitmeplatvormiline mobiiliarenduse keskkond, kuigi kõikide mugavuste ja võimaluste kasutamiseks peab üles seadma ja kasutama kõiki pakutavaid rakendusi. Unikaalne on Rhodes-i juures Ruby arenduskeel ja kasutades RhoHub-i kohalikku SDK vajalikkuse puudumine. Kommertslitsents on Rhodes-il arendatud mobiilirakenduste kommertskasutamiseks kallis.

1.4.4 Xamarin

Xamarin-i arenduskeskkonnas arendatakse mobiilirakendusi C# programmeerimiskeeles ja antud koodibaas kompileeritakse eelistatud platvormile. Arendamistarkvaradeks kasutatakse, kas Xamarin Studio-t või Visual Studio-t. Arendatavad mobiilirakendusi on võimalik kompileerida nii iOS, Android ja Windows Phone seadmetele. [30]

Arenduskeskkonna eeliseks on mobiilirakenduse äriloogika koodibaasi jagatavus erinevate platvormide vahel. Koodibaasi jagatavuse tagab ühtse programmeerimiskeele kasutus erinevatele platvormidele mobiilirakendusi arendades. Oluliseks eeliseks on veel C#

programmeerimiskeel. Arendajad loevad C# programmeerimiskeelt väga sarnaseks Java-ga, kuid C# programmeerimiskeelel on arendajate silmis eelistatud. Samuti loetakse C# programmeerimiskeele õppimisperioodi lühemaks võrreldes Objective C programmeerimiskeelega. [31]

Xamarin-i arenduskeskkonna puuduseks on kasutajaliidese koodibaasi halb skaleeruvus erinevate mobiilsete operatsioonisüsteemide vahel. See tuleneb erinevatest kasutatavatest kasutajaliideste API-dest erinevatele mobiilsetele operatsioonisüsteemidele arendamise korral. Probleemseks kohaks on arendatud koodibaasi halb jagatavus. Nimelt Xamaranis arendatud koodibaasi ei ole kasutatav teistes arenduskeskkondades. [32]

Arenduskeskkond Xamarin paistab välja võimalusega arendada C# programmeerimiskeeles. Individuaalkasutuse litsentsi on võimalik soetada tasuta, kuid organisatsioonisisene odavaim litsents maksab tuhat dollarit aastas. [30] Litsentsi hind seab arendajatele piire ja arenduskeskkonna valik kasutatavuse tõttu sõltub arendatavast mobiilirakendusest.

1.5 Mitmeplatvormiliste mobiilimängude arendus

Eelnevalt täpsemalt kirjeldatud arenduskeskkondi on võimalik kasutada põlisrakenduste loomiseks. Neid keskkondi on võimalik kasutada mobiilirakenduste arendamiseks sõltumata rakenduse enda iseloomust. Teist tüüpi mobiilirakendusteks loetakse mobiilimänge, mis on väga populaarsed mobiilsete seadmete kasutajate seas. Mobiilimängude arendamise lihtsustamiseks on välja töötatud ka mitmeplatvormilisi arenduskeskkondi, mille fookuseks on mobiilimängude arendamine. Eelnevas peatükis välja toodud nimekirjast on sellisteks arenduskeskkondadeks Unity, Corona ja Marmalade. [21]

1.5.1 Unity

Unity on arenduskeskkond, milles on võimalik arendada kolme ja kahe dimensioonilisi mänge. Arenduseks kasutatakse Unity arenduskeskkonna graafilist liidest ja programmeerimiskeelte valikusse kuuluvad C#, Javascript ja Boo. Arendatud mängud on võimalik kompileerida paljudele platvormidele. Platvormide valikusse kuuluvad levinumad mobiilsed platvormid Android, iOS, Windows Phone ja lisaks kuuluvad valikkuse ka mitte mobiilsetest platvormidest OS X, Windows, Nintento Wii, Xbox 360 ja PS3. [33]

Unity kasutamisel on mitmeid eeliseid. Võrreldes teiste arenduskeskkondadega pakub Unity väga suurt platvormide valikut ja ei piirdu selle juures ainult mobiilsete ja töölaua arvutite platvormidega tuues juurde konsoolid. Samuti on arendatud koodil väga hea skaleeruvus platvormide vahel. Unity pakub palju võimalusi graafiliste efektide lisamiseks pakkudes suurt tuge varjudega töötlemiseks. Veel üheks suureks plussiks loetakse Unity arenduskeskkonna graafilist kasutajaliidest. Antud liidesega on võimalik lihtsalt ja loogiliselt ümber käia.[33]

Arendajad on märganud ka mitmeid puudujääke antud arenduskeskkonnas. Kui on soov arendada mängu mitme arendajaga peab tegema lisakulutusi, sest Unity ei toeta väliseid koodihaldureid. Lisakulutuste alla kuuluvad kohustuslik Unity Pro litsents ja lisatarkvara Unity Asset Server. [33] Samuti lisafunktsionaalsused on Unity-l kõik eraldi maksustatud isegi eraldi Android-i ja iOS-i jaoks. [34]

Unity on mitmeplatvormiline arenduskeskkond, mis sobib hästi üksikarendajale. Arendaja, kes otsib väga spetsiifilisi tarkvarakasutuse võimalusi, peab soetama nende kasutusõiguseks lisa litsentse. Litsentside vajalikkus kaasneb ka mitmekesi arendades, sest väliste koodihaldurite kasutus pole toetatud. Arenduskeskkonnana on Unity lihtsalt kasutatav ja kiirelt arusaadav. Samuti toetavad arendatud projektid lihtsa vaevaga paljusi erinevaid platvorme.

1.5.2 Corona

Corona on mitmeplatvormiline tehnoloogia, mis kasutab arendamiseks Lua skriptimise keelt. Corona-ga on võimalik arendatud projekti eelnevalt testida arenduskeskkonna sisesel simulaatoril ja järgnevalt kompileerida soovitud platvormile. Corona toetab iOS ja Android-i platvormi ja arendada saab ainult kahe dimensioonilisi mobiilimänge. [35]

Arendamisel Corona arenduskeskkonnal on mitu eelist. Corona raamistik võimaldab programmeerida mitu korda kiiremini võrreldes teiste programmeerimiskeeltega. Erinevate raamistike ja API-de kasutamine on tehtud lihtsaks ja simulaatoril on võimalik tulemusi koheselt näha. Samuti pakub arenduskeskkond lihtsat tuge mobiilirakenduste sisesteks ostudeks ja reklaamide kuvamiseks. Kergesti on võimalik muuta ka arenduskeskkonnas kasutava simulaatorit esindamaks mitmeid erinevaid mobiilseid seadmeid. [35]

Arenduskeskkonnal on arendajate sõnul ka puudujääke. Corona projektidest on raske vigadega tekkimise korral neid üles leida, ka lihtsamate vigade parandamine võib kaua aega võtta. Kuna Lua skriptimise keelel on vaba süntaks peab tegema suuri pingutusi, et kirjutatud koodibaas loetav ja arusaadav välja näeks. Mõnedel standardsetel kasutajaliideste elementidel on vähene tugi Corona arenduskeskkonnas. Nende elementide vähene tugi võib mobiilirakenduse kasutajaliidese muuta konarlikuks. [35]

Corona on mobiilimängude arendamisele suunatud arendamiskeskkond. Arendamiseks on kasutusel Lua skriptimise keel. Mitmeplatvormilise arenduskeskkonnana toetab Corona vähe platvorme ja mobiilimängude arendus on piiratud kahele dimensioonile. Arenduskeele populaarsus on tingitud lihtsast kiirest võimalikust mobiilimängu arendamisest.

1.5.3 Marmalade

Marmalade-i platvorm lubab arendajatel oma C++ programmeerimiskeeles kirjutatud Xcode-i või Visual Studio projekti oma keskkonnas avada ja kompileerida projekt mitmele soovitud platvormile. Toetatud platvormide hulka kuuluvad iOS, Android, Windows Phone ja ka lauaarvutite operatsioonisüsteemidest Windows ja OS X. Samuti võib oma projekti ka algusest peale Marmalade-i arenduskeskkonnas kirjutada. [36]

Arenduskeskkonna kasutamise eeliseks on oma mobiilimängu lihtne mitmeplatvormilisuse tagamine. Ainult siis peab koodibaasis muudatusi tegema, kui on soov toetada väga platvormi spetsiifilisi tehnilisi võimalusi. Seega suurem osa koodibaasist on kasutatav üle kõikide soovitud platvormide. Arendamine C++ keeles võimaldabki rakenduse lihtsa skaleeruvuse mitme erineva platvormi vahel. Samuti lubab Marmalade lihtsalt kasutada väliseid C++ teeke ja ka Objective C projekti kompileerida teistele platvormidele. [37]

Miinusteks arenduskeskkonna juures on see, et seal esineb kohati vigu, ning mõned vead ei leia Marmalade tootjate poolt kiiret lahendust. Samuti, kui soovitud platvormidele lisatakse mõningaid uusi tehnilisi uuendusi või aplikatsioone, siis nendele toe saamine läbi antud keskkonna võib kaua aega minna. Arendamine Marmalade keskkonnas eeldab ka eelnevat mängude loomise oskust C++ programmeerimiskeeles. [34]

Marmalade arenduskeskkonnal on palju tugevamaid eeliseid võrreldes puudujääkidega. Kogenud C++ arendajale on sellega hea töötada. Võrreldes teiste mitmeplatvormiliste

arenduskeskkondadega on ka litsetnside tasu odavam. Marmalade arenduskeskkond on hea valik mänguarenduse kogemusega arendajatele.

2. Kahemõõtmelise mobiilimängu loomine Unity mängumootori abil

Eelnevalt on kirjeldatud mitut erinevat mitmeplatvormilist arenduskeskkonda, mis on fokuseeritud mobiilimängude arendusele. Igal arenduskeskkonnal on oma plussid ja miinused erinevate arendajate jaoks ja erinevate mobiilimängude arendamise jaoks. Peatükis on kirjeldatud töös valminud mobiilimängu arendamiseks valitud arenduskeskkonda ja selle kasutust.

2.1 Unity mängumootori valik

Mobiilimängude loomine on võimalik kõikides arenduskeskkondades, mida kasutatakse põlisrakenduste arendamiseks. Mobiilimängu arendamiseks tuleb teha valik paljude arenduskeskkondade vahel. Igal arenduskeskkonnal on omad eelised ja puuduseid. Mängumootor Unity osutus valituks mobiilimängu loomisel mitmel põhjusel.

Unity toetab suurt hulka erinevaid platvorme. Platvormide alla ei kuulu ainult mobiilsed operatsioonisüsteemid, vaid sinna kuuluvad ka lauaarvutite operatsioonisüsteemid, konsoolid ja võimalused ka brauseris mängitavaks rakenduseks projekt kompileerida. [33]

Unity arenduskeskkond on tasuta kättesaadav. Tingimus tasuta kehtib täpselt nii kaua kuni ei teki vajadust lisa funktsionaalsuse järgi. Kommertskasutuseks ei oma Unity eraldi litsentsi, vaid antud litsents kuulub Unity Pro versiooni alla. Unity Pro versioon muutub kohustuslikuks kui firma sissetulek, kus Unity arenduskeskkonnaga arendatud mängud on tuluallikaks, ületab sadant tuhandet dollarit aastas. [36]

Unity pakub arenduskeskkonna rohkem tundma õppimiseks nii algajatele kui kogenumatele palju võimalusi. Võimaluste hulka kuuluvad video- ja tekstiõpetused, skriptimise keelte dokumentatsioon, foorumit küsimustele vastuste saamiseks ja otseülekandena toimuvaid videoloenguid. [38]

Unity arenduskeskkond võimaldab graafilisi objekte luua läbi graafilise kasutajaliidese. Graafiliste objektide tekitamine on lihtne ja visuaalselt kohesetl jälgitav. Antud võimalus

kiirendab mängu arenduskäiku ja võimaldab algajatel mänguarendajatel lihtsamalt ümber käia graafiliste objektidega. [33]

Unity mängumootori valik põhines kolmel kriteeriumil – mitmeplatvormilisus, teatud firma sissetulekuni tasuta litsents, algaja mänguarendaja sõbralikkus. Häid arenduskeskkondi mitmeplatvormiliste mobiilimängude arendamiseks on teisigi. Arendajate eelistused erinevad ja iga arendaja teeb valiku oma kriteeriumite järgi.

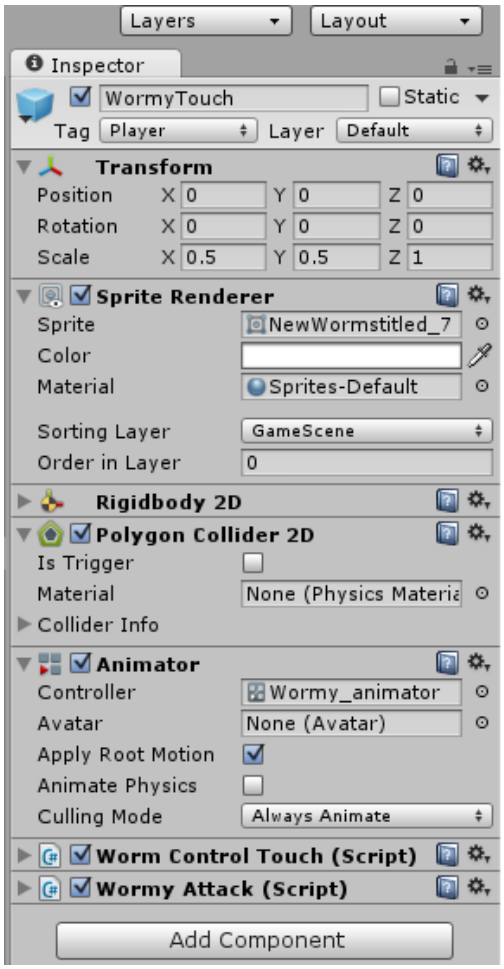
2.2 Tähtsaimad mänguarenduse osad Unity mängumootorist

Unity mängumootoris on palju erinevaid võimalusi ja API-sid, mida arendajad saavad kasutada. Erinevate API-de kasutamine sõltub suuresti arendatavast mobiilimängust. Mobiilimängude arendamisel on aga teatud komponendid, mida kasutatakse kindlasti Unity mängumootorist.

2.2.1 GameObject

Mänguobjekt (inglise k. *GameObject*) on Unity mänguraamistiku põhiliseim objekt. Iseseisvalt on mänguobjekt tühi objekt, aga mänguobjekt töötab konteinerina teistele komponentidele, mis implementeerivad reaalset funktsionaalsust. [39]

Komponentide valik, mida mänguobjekti külge saab rakendada on palju. Samuti saab rakendada mänguobjekti külge enam kui ühe valikus olevatest komponenditest. Vastavalt komponentide valikule võib mänguobjekt esindada mobiilimängus näiteks mängutausta, peategelast ja valgust. Selleks, et aru saada, mida mänguobjektidega saavutada saab peab teadma nende erinevate komponentide kasutusvõimalusi. [39] Näite mänguobjekti ülesehitus, mis esindab peategelast antud töös valminud mobiilimängus, võib näha jooniselt (vt. Joonis 7).



Joonis 7 - Unity mänguobjekti näite ülesehitus

Mänguobjekt on üks objekt, mille kasutamist ei saa mobiilimängu loomise protsessis vältida. Antud objekti iseloomu määravad selle külge rakendatud komponendid. Komponentide kombinatsioonidena on võimalik mänguobjektid panna esindama palju erinevaid mängumaailma erinevaid osi.

2.2.2 Skriptimine

Eelnevalt mainitud mänguobjekti üheks komponendiks saab olla skript. Unity võimaldab skripte kirjutada C#, Javascript ja Boo skriptimise keeltes. Skriptimise arenduskeskkonnaks on Unity mängumootoril arenduskeskkond nagu MonoDevelop. [40]

Unity skriptidel on mitmeid Unity poolt defineeritud funktsioone, mida täidetakse teatud olukordade tekkimisel. Kõige enam kasutust leidvad funktsioonid neist on *Update* – täidetakse ennem, kui järgmine kaader kuvatakse ja *FixedUpdate* – täidetakse iga kord, kui

arvutatakse mänguobjektide füüsikaline käitumine. Veel oluliste funktsioonide alla kuuluvad funktsioonid, mis täidetakse kokkupuudetel teiste mänguobjektidega. [41]

Samuti nagu teistes programmeerimiskeeltes ja arenduskeskkondades on ka Unity arenduskeskkonnas skriptimisel head tavad, mida järgida. Iga skriptimise komponent, mis on mänguobjekti külge rakendatud saab sisaldada ainult ühte skripti. Üks skript vastab objektorienteeritud programmeerimise mõistes ühte klassi, mis hea tavana täidab ühte eesmärki. Kui sellest vaatevinklist läheneda Unity skriptimisele, siis on võimalik ka teisi objektorienteeritud programmeerimise põhimõtteid rakendada. [42]

Unity arenduskeskkonnas on võimalik kirjutada skripte kolmes erinevas skriptimise keeles – C#, Javascript ja Boo. Igal loodavad skriptil on võimalik kasutada Unity poolt defineerituid funktsioone, mis kutsutakse esile teatud olukordade puhul. Kasutades häid programmerimise tavasid on võimalik skriptide uuesti kasutamise võimalust ja refaktoreerimise tõhusust tõsta. Skriptimisega on võimalik Unity mängumootoris panna mänguobjektid ja nende komponendid käituma vastavalt arendaja soovile.

2.2.3 Rigidbody

Mänguobjektile on võimalik lisada komponent nimega *Rigidbody*. Antud komponent võimaldab ära määrata mänguobjekti füüsikalised omadused. Kuna Unity mängumootoris on sisseehitatud füüsika, siis *Rigidbody* komponendi kasutamine võimaldab sisseehitatud füüsikat kasutada. [43]

Kui komponent *Rigidbody* on määratud mänguobjektile, siis on võimalik mänguobjekti liikumist ja pöörlemist mõjutada läbi skriptimise lisades antud objektile jõudusid. Skriptimises on soovitatav mänguobjekti füüsikaline käitumine ära määrata *FixedUpdate* funktsiooni sees, sest füüsikaliste jõudude arvutamise aeg ei kattu alati kaadrite vahetumise ajaga. Mänguobjekt, millel on *Rigidbody* komponent allub gravitatsioonile vastavalt Unity mängumootoris sätestatud gravitatsioonijõu suurusega. Mänguobjekti individuaalse mõjutatus gravitatsioonijõust sõltub sellele sätestatud massist ja konstandist, mis määrab gravitatsioonijõu mõjumise koefitsiendi. [44]

Selleks, et mänguobjekt oleks mõjutatav füüsikalistest jõududest ja gravitatsioonijõust on vajalik sellele lisada *Rigidbody* komponent. Antud komponent määrab ära mänguobjektile

omased füüsikalised suurused. Skriptimisega on võimalik antud komponendi olemasolul rakendada mänguobjektile füüsikalisi jõudusid. *Rigidbody* komponendi kasutamisega on võimalik jäljendada reaalse maailma füüsikalist olukorda ja teha mobiilimängu käitumine loomulikuks.

2.2.4 Collider

Collider (edaspidi kokkupuute jälgija) on komponent, mis jälgib kokkupuuteid teiste mänguobjektidega, millel on ka lisatud kokkupuute jälgija tüüpi komponent. Kokkupuute jälgija lisatakse enamasti mänguobjektile koos *rigidbody* komponendiga, mis võimaldab kokkupuute jälgijal liikuda koos mänguobjektiga. [44]

Vastavalt mänguobjekti visuaalsele esitusele on võimalik lisada erinevaid tüüpi kokkupuute jälgijaid. Kokkupuute jälgijate tüüpe on viis – kast, sfäär, kapsel, *mesh*, ratas ja hulknurk. Kokkupuute jälgija tüüp tuleb valida vastavalt soovile, kuidas mänguobjekt teiste mänguobjektidega kokku puutub. Kui mänguobjekti visuaalset, esitust on keeruline ära katta kindlat tüübi kokkupuute jälgijaga on soovitatav jagada oma keerulisem objekt väiksemateks, millele eraldi on võimalik antud kokkupuute jälgija lisada. Kui mänguobjekt omab mobiilimängus liikumist peab olema lisatud ka *rigidbody* komponent mänguobjektile lisaks kokkupuute jälgijale. [44]

Kokkupuute jälgija on komponent, mis ei oma visuaalset keha, vaid on mänguobjekti komponent jälgimaks mänguobjekti enda kokkupuuteid teiste mänguobjektidega. Kokkupuute jälgija kasutamine on asendatmatu, kui on tarvis jälgida mistahes kokkupuuteid mänguobjektide vahel. Kokkupuute jälgijaga on võimalik jälgida kollisioone kokkupuute jälgijate vahel ja vastavalt soovile realiseerida tagajärg.

2.2.5 Animation Controller

Animatsiooni juhtija (inglise k. *Animation Controller*) on mänguobjekti animatsioonide juhtimiskomponent. Animatsiooni juhtija käib mänguobjekti *Animator* komponendi külge ja kontrollib mänguobjekti animeeritust. Animatsiooni juhtija juurde kuulvad veel animatsioonid ja animatsioonide vahetumistingimused. [45]

Visuaalsetel mänguobjektidel on tavaliselt mitmeid erinevaid animatsioone. Animatsiooni juhtija võimaldab ära määrata, millal peab mänguobjekt teatud animatsiooni mängima.

Animatsiooni juhtijas tuleb ära määrata, millist animatsiooni mängida kui mitte ühtegi teist tingimust ei ole täidetud. Kõige tavalisemaks peamiseks animatsiooniks on peategelase paigalseis. Kui animatsioone on mänguobjektil rohkem kui üks, tuleb ära määrata transaktsioon nende animatsioonide vahel. Transaktsioonide määramiseks kasutatakse animatsiooni juhtijas lisamuutujaid, mille poole pöördutakse läbi skriptide. Transaktsiooni saab ka ära määrata esitamiste kordade järgi, näiteks pärast animatsiooni ühekordset mängimist pöördu teise animatsiooni juurde. [45]

Animatsiooni juhtija on komponent, mis määrab ära, millist animatsiooni teatud hetkel mängida. Mitme animatsiooni korral tuleb animatsioonid oma vahel ühendada transaktsioonidega, mis toimuvad teatud tingimustel. Animatsioonide vahetumist saab määrata skriptidega, mis muudavad animatsiooni juhtija muutujaid. Animatsiooni juhtija teeb võimalikus Unity mängumootoris mänguobjektide animeerimise.

2.3 Mobiilimängu „Harvest Crunch“ arendus

2.3.1 Arenduskäik

1. Tutvumine Unity kodulehel kättesaadavate juhenditega. [38]

   Unity pakub kodulehel mitmeid võimalusi Unity arendustarkvara õppimiseks. Valikute seas on videojuhendid, dokumentatsioon, foorumid ja arhiveeritud videoloengud.

2. Näidismängu loomine. [46]

   Näidismängu loomiseks on Unity kodulehel videojuhendite kogum. Näidismängu jaoks vajalik graafika on Unity poolt tagatud.

3. 2D ja mobiilimängu arenduse videojuhendite vaatamine. [47][48]

   Autor tutvus videojuhenditega kahedimensiooniliste mängude arendamisest ja arhiveeritud videoloenguga mobiilimängu arendusest.

4. Mobiilimängu tausta, kaamera ja peategelase algne loomine – animatsioon.

   Antud arendusjärgus lõi autor mobiilimängu tausta jaoks mänguobjekti, sätestades selle visuaalseks kujutiseks pildi. Unity arenduskeskkonna poolt loodud kaamera

sätestati mobiilimängu taustast lähtuvale suurusele. Loodi peategelane ja lisati peategelasele animatsioon.

5. Peategelase puute- ja kallutustundliku juhtimisvõimaluse loomine.

   Peategelase mänguobjektile lisas autor skripti, mis liigutab peategelast vastavalt kasutaja sisendile. Peategelase liikumist mobiilset seadet kallutades kirjeldav skript on välja toodud joonisel (vt. Joonis 8 - Peategelase liikumist kallutustega kirjeldav skript).

```
void Update () {

    Vector3 dir = new Vector3 (
        Input.acceleration.x,
        Input.acceleration.y,
        0.0f) * speed;

    transform.Translate(dir);

    transform.position =
            new Vector3(
                Mathf.Clamp (transform.position.x, boundary.xMin, boundary.xMax),
                Mathf.Clamp (transform.position.y, boundary.yMin, boundary.yMax),
                0.0f
        );
```

Joonis 8 - Peategelase liikumist kallutustega kirjeldav skript

6. Mängujuhi objekti loomine.

   Autor lõi mängujuhi objekti, millel puudub visuaalne esitus. Mängujuhi objektiga on seotud skript, mis kontrollib rakenduse käiku selle käivitamisest kuni mängu taseme algamiseni.

7. Avalehe ja nuppude loomine – tekkimine, liikumine, vajutusel esinevad sündmused.

   Antud faasis lõi autor avalehe ja nuppude mänguobjektid. Avalehe ja nuppude tekkimine lisati mängujuhi objekti skripti. Nuppudele vajutamisele reageerimise skript lisati nuppudele (vt. Joonis 9 - Nupule vajutamist jälgiv skript).

```
if (Input.touchCount == 1)
{
    Vector3 wp = Camera.main.ScreenToWorldPoint(Input.GetTouch(0).position);
    Vector2 touchPos = new Vector2(wp.x, wp.y);
    if (collider2D == Physics2D.OverlapPoint(touchPos) && Input.GetTouch(0).phase == TouchPhase.Ended)
    {
        gameController.playButtonPressed();
    }
```

Joonis 9 - Nupule vajutamist jälgiv skript

8. Vaenlase loomine – tekkimine, liikumine, kokkupuute registreerimine, animatsioon.

   Elu objekti loomine.

   Autor lõi kaks uut mänguobjekti. Elu mänguobjektile määrati visuaalne pilt. Vaenlase mänguobjektile määrati visuaalne pilt ja animatsioon. Vaenlase mänguobjektile määrati juhuslik liikumissuund lähtuvalt tekkimise positsioonist. Lisaks määrati peategelasele kokkupuudete registreerimise skript.

9. Tasemete kontrollija objekti loomine – elude kontroll, taseme alustamine, mängu lõpetamine, vaenlaste tekkimine.

   Antud faasis lõi autor tasemete kontrollija mänguobjekti, mille ülesandeks on skriptidega kontrollida mängu käiku.

10. Peategelase rünnaku loomine – animatsioon, rünnaku kontroll kokkupuutel, rünnaku esilekutsumine ekraani puudutamisel.

    Autor lõi peategelasele uue animatsiooni ja animatsiooni esile kutsumise tingimused. Lisati peategelase kokkupuute jälgimisele ründefaasi kontroll (vt. Joonis 10 - Peategelase kokkupuute jälgija).

```
void OnCollisionEnter2D(Collision2D collision) {

    if (collision.gameObject.tag == "Opponent") {

        if(isAttacking){
            Destroy(collision.gameObject);
        } // end if
        else if(!takingDamage){
            StartCoroutine(wormyTakingDamage());
            Destroy(collision.gameObject);
        } // end else

    } // end if
}
```

Joonis 10 - Peategelase kokkupuute jälgija

11. Mängu lõpetamise ja uuesti alustamise loomine.

    Tasemete kontrollija objektile lisati elude otsa lõppemisel mängu peatamise loogika. Samuti lisati võimalus tasemete kontrollija skripti mängu uuesti alustamiseks

12. Boonusobjekti lisamine – tekkimistingimused, liikumine, peategelase animatsioon, uued puutetingimused.

    Autor lõi kaks uut mänguobjekti – boonusobjekt ja kokkupuutel tekkiv kilp. Boonusobjekti tekkimistingimuste loogika lisati tasemete kontrollija skripti. Boonusobjektile lisas autor kokkupuute jälgimise peategelasega ja kokkupuutel peategelasele tekkiva kilbi loogika.

2.3.2 Valminud kahemõõtmeline mobiilimäng

Autor arendas mobiilimängu kasutades Unity mängumootorit kahemõõtmelise seadistusega. Kompileerimisplatvormiks oli seatud Android-i mobiilne operatsioonisüsteem. Mobiilimängu arendamise käigus kasutatud graafika autor on Anton Servetnik.

Mobiilimängu Harvest Crunch eesmärk on kontrollides peategelaseks olevat õunaussi jääda ellu võimalikult pikaks ajaks. Mängijat takistavad mängumaailmas teised putukad, kes üritavad õunaussile viga teha. Õunaussil on võimalus ennast putukate eest kaitsta. Putukatest pääsemiseks on nendest võimalik hoiduda, neid rünnata, või nendega kokkupuutudes omada kaitsekilpi. Kokkupuutudes putukatega kaotab õunauss elusid. Kaotades kõik kolm elu saab mäng läbi. Mängu on võimalik uuesti alustada.

Mängu alustamiseks peab vajutama „Play“ nuppu ja seejärel valima eelistatud peategelase kontrollimisviisi. Peategelase kontrollimisviisi valikuks tuleb vastavalt vajutada „Touch“ – puutetundlik juhtimine või „Tilt“ – mobiilse seadme kallutustundlik juhtimine. Valinud kontrollimisviisi algab mäng. Juhtides peategelast on võimalik ka mängumaailmas liikuvaid putukaid rünnata. Ründamiseks peab kasutaja puudutama mobiilse seadme ekraani. Puudutades mobiilse seadme ekraani sooritab õunauss ühe rünnaku. Kui rünnaku vältel puutub õunauss kokku mõne putukaga ei saa õunauss viga ja putukas hävib. Kui õunauss puutub kokku putukaga sellel hetkel, kui õunauss parasjagu ei ründa kaotab õunauss elu ja putukas hävib. Mobiilimängu vältel ilmub mänguväljale ka lehekesi. Puudutades peategelasega lehekest aktiveeritakse õunaussi kaitsekilp. Kaitsekilp kaitseb õunaussi järgmise kokkupuute eest putukaga. Kaitsekilp hävib esimesel kokkupuutel putukaga. Kaotades kõik elud lõpeb mobiilimäng. Mängu uuesti alustamiseks on vajalik ekraani uuesti puudutamine.

2.3.3 Situatsioonid mobiilimängust:

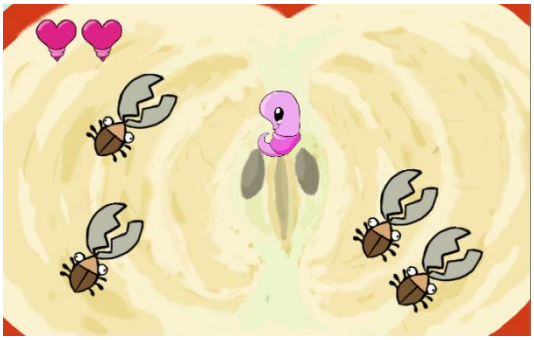
1. Mängu laadimisel avanev aken (vt. Joonis 11 - Harvest Crunch Play Screen)

Joonis 11 - Harvest Crunch Play Screen

2. Vajutades „Play“ järgnev aken (vt. Joonis 12 - Harvest Crunch Control Selection)

Joonis 12 - Harvest Crunch Control Selection

3. Mängu käik (vt. Joonis 13 - Harvest Crunch Gameplay)

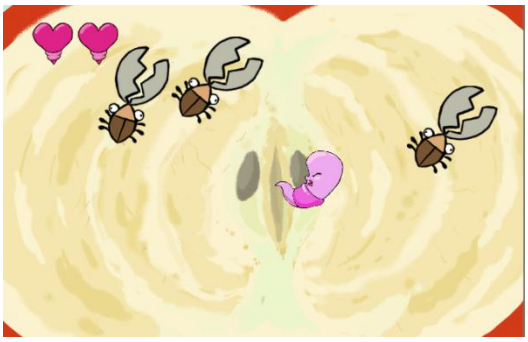


Joonis 13 - Harvest Crunch Gameplay

4. Ründav peategelane (vt. Joonis 14 - Harvest Crunch Wormy Attack)

Joonis 14 - Harvest Crunch Wormy Attack

5. Viga saav peategelane (vt. Joonis 15 - Harvest Crunch Wormy Take Damage)



Joonis 15 - Harvest Crunch Wormy Take Damage

6. Mäng läbi (vt. Joonis 16 - Harvest Crunch Game Over)

Joonis 16 - Harvest Crunch Game Over

7. Kaitsekilp ja kaitsekilpi aktiveeriv leheke (vt. Joonis 17 - Harvest Crunch Shield Active)

Joonis 17 - Harvest Crunch Shield Active

2.4 Hinnang Unity arenduskeskkonnale

Hakates mobiilimängu arendama Unity arenduskeskkonnas pole eelnevatest programmeerimise kogemustest palju abi. Arendaja puutub kokku skriptimises enamasti Unity enda poolt defineeritud objektidega, mille meetodid ja kasutusvõimalused on vaja ennem keerulisemat arendust selgeks teha.

Skriptimise käigus tekkinud vigade leidmine on Unity arenduskeskkonnas keeruline. Kuna vead on kergesti esinema Unity enda poolt defineeritud objektidega ja funktsioonidega, ei ole algajal Unity arendajal lihtne nendest vigadest aru saada ja neid parandada.

Arendades Unity arenduskeskkonnas kindlale platvormile mängu on vajalik lokaalne SDK-de olemasolu. Ennem arendama asumist on vaja viidata sätetes soovitud platvormi SDK asukohale.

Visuaalsete objektide tekitamine ja nende animeerimine on Unity arenduskeskkonnas tehtud lihtsaks. Uurides esmalt Unity kodulehel olevaid õppevideosi on animeerimine võimalik kiiresti selgeks saada.

Vaatamata uutele Unity objektidele on võimalik vähese tutvumise ja katsetamistega saadud kogemustega kergesti leida soovitud abi probleemide lahendamiseks pöördudes dokumentatsiooni või kasutajate foorumi poole. Samuti pakub Unity arenduskeskkonna juurde kuuluv MonoDevelop arendamise abi ja soovitusi viidates funktsioonidele ja skriptide parandamisele.

Unity arenduskeskkonda kasutavate arendajate hulk on pidevas tõusujoones. Tulenevalt sellest avastatakse Unity arenduskeskkonna vead kiiresti ja edastatakse korrektuurideks Unity arenduskeskkonna arendajatele. Samuti suur kasutajaskond hõlbustab tekkinud probleemide lahendamist läbi kasutajate foorumi.

Võttes arvesse eelnevalt käsitletud Unity arenduskeskkonna eeliseid ja puudujääke, ning mobiilimängu arendamise käigus esile kerkinud positiivsetest ja negatiivsetest külgedest võib anda Unity arenduskeskkonnale hinnangu. Unity arenduskeskkonna kasutajate hulk on tõusujoones ja see on põhjendatud. Vaatamata esinevatest negatiivsetest külgedest toetab Unity arenduskeskkonna õppimist mitmed kodulehel kättesaadavad abivahendid. Samuti lihtsustab Unity mitmeid mitmeplatvormiliste mobiilimängude arendamise juures tekkivaid komplikatsioone.

Kokkuvõte

Bakalaureusetöö eesmärkideks oli uurida mobiiliplatvorme, mobiilirakenduste arendamisviise, mitmeplatvormilist arenduskeskkondi ja arendada kahemõõtmeline mobiilimäng.

Autor puutus uurimise käigus kokku mobiiliplatvormidega iOS, Android ja Windows Phone, mobiilirakenduste tüüpidega *Native App* ja *Mobile Web App*, arenduskeskkondadega Appcelerator, Mosync, RhoMobile, Xamarin ja mobiilimängu arenduskeskondadega Unity, Corona ja Marmalade. Autor valis mobiilimängu arenduseks Unity arenduskeskkonna ja kirjeldas arenduseks kasutatud objekte ja arenduskäiku.

Järeldustena tõi autor välja mobiilsete platvormide rohkuse ja kulukuse, arendades mobiilirakendust, mis peab töötama soovitud mobiilsetel platvormidel. Kasutades platvormispetsiifilisi arenduskeskkondi on võimalik arendada ühele soovitud platvormile hästi töötav mobiilirakendus, aga selle laiendamine teistele platvormidele on raskustatud.

Autor mainis veel, et on olemas lai valik mitmeplatvormilisi arenduskeskkondi, millel on kõigil omad eelised ja puudujäägid. Autor järeldas erinevate arenduskeskkondi uurides, et arenduskeskkonna valik sõltub arendaja eelistustest ja arendatavast mobiilirakendusest.

Unity mängumootori valikut mobiilse mängu arendamiseks põhjendas autor arenduskeskkonna vastavusega kolmele kriteeriumile. Kriteeriumiteks olid arenduskeskkonna mitmeplatvormiline tugi, tasuta kommertskasutus litsents teatud piirini ja algaja mänguarendaja sõbralikkus. Samuti tõi autor välja mobiilimängu arenduse käigus olulisemad Unity mängumootori osad.

Vaatamata mitmetele puudustele hindas autor Unity arenduskeskkonda positiivselt. Hinnangu põhjenduseks lisaks välja toodud eelistele olid mobiilimängu arenduse lihtsustamine ja küllane abimaterjalide kogu Unity koduleheküljel.

Uurides eelnimetatud teemasid ja arendades tulemusena mobiilimängu, jõudis autor oma püstitatud probleemide lahenduseni. Bakalaureusetöö võimalikeks edasiarendusteks on arendatud mobiilimängu turustamine, tulu saamine läbi mobiilirakenduse poe, tulu teenimise mudeli toetamine mobiilirakendusega ja selle toe arendus mobiilirakendusele.

Summary

The purpose of the thesis was to research mobile platforms, mobile application development tools, multi-platform mobile application development tools and develop a two dimensional mobile game.

The Author researched mobile platforms like iOS, Android and Windows Phone, types of mobile applications Native App and Mobile Web App, development tools Appecelerator, MoSync, RhoMobile, Xamarin and mobile game development tools Unity, Corona and Marmalade. For developing a mobile game the author chose Unity development tools and explained the development tool choice, explained the most important components from Unity and explained the development process.

As a result the author mentioned the abundance of mobile platforms and to develop a mobile application, which works on every desired mobile platform is costly. Using platform-specific development tools it is possible to develop a well performing mobile application for one desired platform, but to adapt this mobile application to different mobile platforms is difficult.

The author also mentioned that there is a wide range of multi-platform development tools, which all have their own pros and cons. The author concluded by researching development tools that the development tool choice depends on developer's preferences and the mobile game to be developed.

The author explained that the choice of Unity game engine depended on three criteria. The criteria were Unity's multi-platform support, free commerce licence to some extent, beginner mobile game developer friendliness. In additon the author mentioned the key components of the chosen development tools, which the author came across during the mobile game development.

Despite many cons regarding Unity game engine the author's feedback was positive. In addition to previously mentioned pros the author mentioned that Unity's game engine simplifies mobile game development and Unity provides sufficient amount of help for developers in their web page.

By researching previously mentioned topics and developing a mobile game in result the author reached the purposes which the author set. The possibilities for further research would be marketing of the developed mobile game, obtaining revenue through mobile application store, supporting the revenue model with the mobile game and developing the support for the revenue model.

Viidatud allikad

[1] „Mobile operating systems,“ Wikipedia, 20 5 2014. [Võrgumaterjal]. http://en.wikipedia.org/wiki/Mobile_operating_system. [Kasutatud 2014 5 25].

[2] R. Sharma, „Developing for Android - An Introduction,“ [Võrgumaterjal]. http://www.cprogramming.com/android/android_getting_started.html. [Kasutatud 26 5 2014].

[3] Google, [Võrgumaterjal]. http://developer.android.com/tools/index.html. [Kasutatud 26 5 2014].

[4] Apple, [Võrgumaterjal]. https://developer.apple.com/library/ios/documentation/miscellaneous/conceptual/iphoneostechoverview/Introduction/Introduction.html. [Kasutatud 26 5 2014].

[5] „Cocoa Touch Layer,“ Apple, [Võrgumaterjal]. https://developer.apple.com/library/ios/documentation/miscellaneous/conceptual/iphoneostechoverview/iPhoneOSTechnologies/iPhoneOSTechnologies.html#//apple_ref/doc/uid/TP40007898-CH3-SW1. [Kasutatud 26 5 2014].

[6] „Media Layer,“ Apple, [Võrgumaterjal]. https://developer.apple.com/library/ios/documentation/miscellaneous/conceptual/iphoneostechoverview/MediaLayer/MediaLayer.html#//apple_ref/doc/uid/TP40007898-CH9-SW4. [Kasutatud 6 1 2014].

[7] „Core Services Layer,“ Apple, [Võrgumaterjal]. https://developer.apple.com/library/ios/documentation/miscellaneous/conceptual/iphoneostechoverview/CoreServicesLayer/CoreServicesLayer.html#//apple_ref/doc/uid/TP40007898-CH10-SW5. [Kasutatud 1 6 2014].

[8] „Core OS Layer,“ Apple, [Võrgumaterjal]. https://developer.apple.com/library/ios/documentation/miscellaneous/conceptual/iphoneostechoverview/CoreOSLayer/CoreOSLayer.html#//apple_ref/doc/uid/TP40007898-CH11-SW1. [Kasutatud 1 6 2014].

[9] „iOS Developer Tools,“ Apple, [Võrgumaterjal]. https://developer.apple.com/library/ios/documentation/miscellaneous/conceptual/iphoneostechoverview/iPhoneOSDeveloperTools/iPhoneOSDeveloperTools.html#//apple_ref/doc/uid/TP40007898-CH7-SW1. [Kasutatud 1 6 2014].

[10] „Chapter 1. Vision and architetcure,“ O'Reilly Media, Inc., [Võrgumaterjal]. http://chimera.labs.oreilly.com/books/1234000001853/ch01.html. [Kasutatud 05 28 2014].

[11] „Microsoftstore,“ Microsoft, [Võrgumaterjal]. http://www.microsoftstore.com/store/msusa/en_US/cat/Windows/categoryID.62684800. [Kasutatud 1 6 2014].

[12] L. Whitney, „App developers challenged by number of different devices,“ NET, [Võrgumaterjal]. http://www.cnet.com/news/app-developers-challenged-by-number-of-different-devices/. [Kasutatud 2 6 2014].

[13] „List of iOS devices,“ Wikipedia, [Võrgumaterjal]. http://en.wikipedia.org/wiki/List_of_iOS_devices. [Kasutatud 2 6 2014].

[14] „Comparison of Android devices,“ Wikipedia, [Võrgumaterjal]. http://en.wikipedia.org/wiki/Comparison_of_Android_devices. [Kasutatud 2 6 2014].

[15] „List of Windows Phone 8.1 devices,“ Wikipedia, [Võrgumaterjal]. http://en.wikipedia.org/wiki/List_of_Windows_Phone_8.1_devices. [Kasutatud 2 6 2014].

[16] „List of Windows Phone 8 Devices,“ Wikipedia, [Võrgumaterjal]. http://en.wikipedia.org/wiki/List_of_Windows_Phone_8_devices. [Kasutatud 2 6 2014].

[17] „Comparison of tablet computers,“ Wikipedia, [Võrgumaterjal]. http://en.wikipedia.org/wiki/Comparison_of_tablet_computers. [Kasutatud 2 6 2014].

[18] „Designing content for multiple mobile devices,“ Learning Solutions Magazine, [Võrgumaterjal]. http://www.learningsolutionsmag.com/articles/1018/designing-content-for-multiple-mobile-devices. [Kasutatud 2 6 2014].

[19] P. Viswanathan, „The Pros And Cons Of Native Apps And Mobile Web Apps,“ About.com, [Võrgumaterjal]. http://mobiledevices.about.com/od/additionalresources/qt/The-Pros-And-Cons-Of-Native-Apps-And-Mobile-Web-Apps.htm. [Kasutatud 3 6 2014].

[20] P. Viswanathan, „Cross-Platform Tools: Are they Really Worth It?,“ About.com, [Võrgumaterjal]. http://mobiledevices.about.com/od/additionalresources/fl/Cross-Platform-Tools-Are-they-Really-Worth-It.htm. [Kasutatud 2 6 2014].

[21] J. Cowart, „Pros and Cons of the Top 5 Cross-Platform Tools,“ Developer Economics, 12 11 2013. [Võrgumaterjal]. http://www.developereconomics.com/pros-cons-top-5-cross-platform-tools/. [Kasutatud 3 6 2014].

[22] P. Viswanathan, „Top 5 Tools for Multi-Platform Mobile App Development,“ About.com, [Võrgumaterjal]. http://mobiledevices.about.com/od/mobileappbasics/tp/Top-5-Tools-Multi-Platform-Mobile-App-Development.htm. [Kasutatud 3 6 2014].

[23] „Titanium Mobile Development Environment,“ Appcelerator, [Võrgumaterjal]. http://www.appcelerator.com/titanium/. [Kasutatud 4 6 2014].

[24] G. Dhat, „10 Pupolar Cross Platform Mobile App Development Tools,“ Decos, 29 11 2013. [Võrgumaterjal]. http://www.decossoftdev.com/insight/blog/2013/11/29/10-Popular-Cross-Platform-Mobile-App-Development-Tools. [Kasutatud 4 6 2014].

[25] E. Angelini, „5 Pros and Cons of Appcelerator's Titanium,“ Spirales, 28 2 2012. [Võrgumaterjal]. http://enricoangelini.com/2012/5-pros-and-cons-of-appcelerators-titanium/. [Kasutatud 4 6 2014].

[26] Roman, „Appcelerator Titanium evaluation,“ Softaria, 29 7 2011. [Võrgumaterjal]. http://softaria.com/2011/07/29/appcelerator-titanium-evaluation/. [Kasutatud 4 6 2014].

[27] „MoSync SDK,“ MoSync, [Võrgumaterjal]. Available: http://www.mosync.com/sdk. [Kasutatud 4 6 2014].

[28] „Frameworks for cross-platform mobile develpoment,“ Shakuro, 23 10 2012. [Võrgumaterjal]. http://shakuro.com/blog/view/frameworks-for-cross-platform-mobile-development. [Kasutatud 4 6 2014].

[29] R. Jha, „Rhodes Architecture and Requirements,“ 13 1 2013. [Võrgumaterjal]. http://rajthoughts.wordpress.com/2013/01/13/rhodes-architecture-and-requirements/. [Kasutatud 4 6 2014].

[30] „Xamarin,“ Xamarin, [Võrgumaterjal]. https://xamarin.com/. [Kasutatud 4 6 2014].

[31] „Xamarin Advantages,“ Mobmaxine, [Võrgumaterjal]. http://mobmaxime.com/xamarin-advantages/. [Kasutatud 5 6 2014].

[32] L. Whtiney, „Why I Don't REcommend Xamarin for Mobile Development,“ Whitneyland, 30 5 2013. [Võrgumaterjal]. http://www.whitneyland.com/2013/05/why-i-dont-recommend-xamarin-for-mobile-development.html. [Kasutatud 5 6 2014].

[33] A. Doulin, „Unity 3 Review,“ 16 12 2010. [Võrgumaterjal]. http://www.doolwind.com/blog/unity-3-review/. [Kasutatud 5 6 2014].

[34] M. Sawitus, „iOS and Android game development on Windows,“ 1 2 2012. [Võrgumaterjal]. http://gamedevcoder.wordpress.com/2012/02/01/ios-and-android-game-development-on-windows/. [Kasutatud 4 6 2014].

[35] W. J. Francis, „Cross-platform vs. native development: Corona SDK,“ Software Engineer, 3 7 2013. [Võrgumaterjal]. http://www.techrepublic.com/blog/software-engineer/cross-platform-vs-native-development-corona-sdk/. [Kasutatud 5 6 2014].

[36] A. Lee, „15 essential mobile game development tools,“ Develop-online, 15 10 2013. [Võrgumaterjal]. http://www.develop-online.net/tools-and-tech/15-essential-mobile-game-development-tools/0184480. [Kasutatud 5 6 2014].

[37] „How Marmalade works,“ Marmalade, [Võrgumaterjal]. https://www.madewithmarmalade.com/marmalade/how-marmalade-works. [Kasutatud 5 6 2014].

[38] „Learn with Unity,“ Unity 3D, [Võrgumaterjal]. http://unity3d.com/learn. [Kasutatud 6 6 2014].

[39] „GameObjects,“ Unity, [Võrgumaterjal]. http://docs.unity3d.com/Manual/GameObjects.html. [Kasutatud 7 6 2014].

[40] „Welcome to the Unity Scripting Reference,“ Unity, [Võrgumaterjal]. http://docs.unity3d.com/ScriptReference/. [Kasutatud 7 6 2014].

[41] „Scripting Overview,“ Unity, [Võrgumaterjal]. http://docs.unity3d.com/412/Documentation/ScriptReference/index.html. [Kasutatud 7 6 2014].

[42] „Good Coding Practices in Unity,“ Unity, [Võrgumaterjal]. http://unity3d.com/learn/tutorials/modules/intermediate/scripting/coding-practices. [Kasutatud 7 6 2014].

[43] „Rigidbody,“ Unity, [Võrgumaterjal]. http://docs.unity3d.com/ScriptReference/Rigidbody.html. [Kasutatud 7 6 2014].

[44] „Rigidbody,“ Unity, [Võrgumaterjal]. http://docs.unity3d.com/Manual/class-Rigidbody.html. [Kasutatud 7 6 2014].

[45] „Animator and Animation Controller,“ Unity, [Võrgumaterjal]. http://docs.unity3d.com/Manual/Animator.html. [Kasutatud 7 6 2014].

[46] „Project: Space Shooter“ , Unity, [Võrgumaterjal]. http://unity3d.com/learn/tutorials/projects/space-shooter. [Kasutatud 9 6 2014]

[47] „Mobile Development“, Unity, [Võrgumaterjal]. http://unity3d.com/learn/tutorials/modules/beginner/live-training-archive/mobile-development. [Kasutatud 9 6 2014].

[48] „2D“, Unity, [Võrgumaterjal]. http://unity3d.com/learn/tutorials/modules/beginner/2d. [Kasutatud 9 6 2014].