Heikki Väyrynen

Monialustaisen mobiilisovelluksen kehittäminen ja julkaisu

Opinnäytetyö
Tieto- ja viestintätekniikka / Peliohjelmointi

2020

Kaakkois-Suomen ammattikorkeakoulu

Tekijä/Tekijät	Tutkinto	Aika
Heikki Väyrynen	Insinööri (AMK)	Toukokuu 2020
Opinnäytetyön nimi Monialustaisen mobiilisovelluksen kehittäminen ja julkaisu		47 sivua
Toimeksiantaja Gamelab / Niina Mässeli		
Ohjaaja Marko Oras		

Tiivistelmä

Tämä opinnäytetyö dokumentoi Android- ja iOS-käyttöjärjestelmiin toteutetun sovelluksen suunnittelun, kehittämisen ja julkaisun. Sovellus sisältää kolme peliä, joiden tarkoituksena on näyttää erilaisia keinoja ottaa vastaan kontrolleja mobiililaitteen kosketusnäytöltä.

Kyseessä oli kehittämistyö, jonka tarkoituksena oli kartoittaa suunnittelussa ja kehityksessä huomioon otettavia asioita mobiilikehityksen kannalta ja selventää, miten julkaisuprosessi molemmille alustoille tapahtuu.

Työ toteutettiin käyttäen Unity-pelimoottoria yhteistyössä Android Studio -ohjelmointiympäristön ja Xcode-ohjelmointiympäristön kanssa. Ohjelmointikielenä käytettiin C#-ohjelmointikieltä.

Lopullisessa tuotoksessa sovellus saatiin julkaisuvalmiiksi molemmille alustoille, mutta varsinainen julkaisu onnistuttiin tekemään ainoastaan Android-käyttöjärjestelmälle. Apple-kehittäjätilissä ei ollut vaadittavia oikeuksia julkaisua varten. Julkaisua varten molemmille alustoille tehtiin kattavat ohjeistukset.

Asiasanat

peliohjelmointi, Android, iOS, mobiilikehitys, mobiilisovellus, julkaisu

South-Eastern Finland
University of Applied Sciences

Author (authors)	Degree	Time
Heikki Väyrynen	Bachelor of Engineering	May 2020

Thesis title

Development and publishing of
a multiplatform mobile application

47 pages

Commissioned by

Gamelab / Niina Mässeli

Supervisor

Marko Oras

Abstract

This thesis examines the planning, development, and publishing of an application for Android and iOS operating systems. The application consists of three games with the purpose of showcasing different methods of receiving input from the mobile touch screen.

This thesis studies the factors to be considered when planning and developing an application for mobile devices and describes the publishing process for both aforementioned operating systems.

The thesis study was conducted using Unity game engine in collaboration with Android Studio IDE and Xcode IDE. C#-programming language was used for programming.

The application was completed and could have been published on both operating systems. But since the Apple developer account lacked the required permissions for publishing, the actual publishing was completed only for Android. However, the thesis gives clear instructions for publishing applications on both operating systems.

Keywords

game programming, Android, iOS, mobile development, mobile application, publishing

SISÄLLYS

KUVALUETTELO

KÄSITTEIDEN MÄÄRITTELY

Android SDK	Software Developement Kit eli Android-käyttöjärjestelmän ohjelmisto kehitykseen vaadittavat työkalut.
Android NDK	Native Development Kit eli Android-käyttöjärjestelmän työkalut käytettäessä C++- ja C-ohjelmointikieliä.
APK	Android Application Package on Android-sovellusten tiedostotyyppi.
App Bundle	Android-alustan uusi virallinen sovelluksen julkaisumuoto.
AR	Augmented Reality eli lisätty todellisuus. Näkymä, johon on tuotettu tietokonegrafiikalla elementtejä.
ARM	Advanced RISC Machines eli 32-bittinen mikroprosessoriarkkitehtuuri, joita käytetään matkapuhelimien suorittimissa.
ARM64	64-bittinen versio ARM-mikroprosessoriarkkitehtuurista.
ARMv7	32-bittinen versio ARM-mikroprosessoriarkkitehtuurista.
Boolean	Muuttujatyyppi, jolla on kaksi mahdollista arvoa: tosi tai epätosi.
Endless runner	Pelityyppi, jossa pelaajan on tarkoituksena selvitä mahdollisimman pitkään ja kerätä siten pisteitä. Peleissä ei ole loppua, vaan ne jatkuvat niin kauan kuin pelaaja selviää.
IL2CPP-kääntäjä	Unity-pelimoottorista löytyvä kääntäjä, joka mahdollistaa sovelluksen kääntämisen C++-ohjelmointikielelle ennen sovelluksen rakentamista.
IntelliJ IDEA	JetBrains-yhtiön kehittämä ohjelmointiympäristö Java-pohjaiseen ohjelmistokehitykseen.
JDK	Java Development Kit eli Java-ohjelmistojen kehittämiseen suunniteltu kehittämisympäristö.
Ortografinen kamera	Kamera, joka ei kuvaa perspektiiviä vaan kaikki esineet näyttävät olevan samassa tasossa.
Pikselitaide	Piirtämistapa, jossa yksittäisiä pikseleitä värjäämällä luodaan kuva.
Swift	Apple-yhtiön kehittämä ohjelmointikieli.
UDID-numero	Unique Device ID eli Apple-laitteista löytyvä 40-numeron sarja, jota käytetään laitteiden tunnistamiseen.
UI	User Interface eli käyttöliittymä.

1 JOHDANTO

Opinnäytetyössä selvitetään pelin kehittämistä mobiilialustoille aina suunnittelusta julkaisuun asti. Työssä keskitytään pääasiallisesti mobiilialustoille tärkeisiin osuuksiin niin ohjelmoinnin kuin käyttöliittymän kannalta ja testaukseen ja julkaisuun vaadittaviin toimiin. Tavoitteena kehittää toimiva sovellus, jonka voi julkaista Android- ja iOS-käyttöjärjestelmille.

Toimeksiantajana toimii Kaakkois-Suomen ammattikorkeakoulun Gamelab. Gamelab on peliohjelmoinnin insinöörikoulutuksen laboratorio. Se toimii yhteistyössä pelialan yritysten ja ammattilaisten kanssa, antaen opiskelijoille mahdollisuuden saada kattavan ammattitaidon pelialalle ja ohjelmistokehitykseen. (Gamelab.fi 2019.)

Opinnäytetyössä toteutettiin Unity-pelimoottorilla sovellus, joka sisälsi kolme peliä, joiden suunnittelussa keskityttiin erityyppisten kontrollien luontiin. Kaikki pelien kontrollit toteutettiin käyttäen mobiililaitteen kosketusnäyttöä. Sovelluksen julkaisu molemmille alustoille ei toteutunut Apple-kehittäjätilin oikeuksien puuttumisen takia. Android-käyttöjärjestelmälle julkaisu saatiin tehtyä, mutta opinnäytetyön valmistuessa sovellus ei ollut vielä läpäissyt Googlen tarkistusta.

2 TEORIA

Mobiilikehityksessä on otettava huomioon laitteiden erilaiset vaatimukset ja käyttöjärjestelmät. Yleisimmät käyttöjärjestelmät mobiililaitteille ovat Android ja iOS. Nämä kaksi käyttöjärjestelmää vastaavat lähes 100 prosentin markkinaosuudesta mobiililaitemarkkinoilla (Mobile Operating System Market Share Worldwide 2020.)

2.1 Android

Android on alun perin vuonna 2003 perustetun Android Inc. -yhtiön suunnittelema ohjelmisto kameroihin, mutta yhtiö myytiin vuonna 2005 Googlelle ja Android OS -käyttöjärjestelmän kehitys puhelimille alkoi. Google muodosti ryhmän nimeltä Open Handset Alliance, joka nykyäänkin vastaa Android OS -

käyttöjärjestelmän kehityksestä. Open Handset Allianceen kuuluu monia teknologia-alan yrityksiä monilta mobiilialan osa-alueelta aina matkapuhelimien valmistajista teleoperaattorihin. (Callaham 2019.)

Androidin lähdekoodi on vapaa ja siitä voi siis kehittää omia versioitaan halutessaan. Kuitenkin yleisimmät versiot Androidista sisältävät Googlen omia ohjelmistoja, joten nämä versiot ovat vain osin vapaita. (Hoffman 2017.)

Android antaa paljon vapauksia erilaisten sovellusten käyttöön. Androidissa on mahdollista suorittaa ohjelmia, joita ei ole ladattu Google Play Store -sovelluskaupan kautta. Android antaa sovelluksille paljon vapauksia erilaisiin toimintoihin, joilla voi muokata käyttöjärjestelmän ulkonäköä ja toimintoja. (Hoffman 2017.)

Kehittäjiä varten Androidilta löytyy valtava määrä ohjeita ja suunnittelusuosituksia. Suunnitteluohjeistus jakautuu 6 osa-alueeseen: tyyli, käytettävyys, komponentit, asettelu, kuviot ja animaatiot. Jokainen osa-alue sisältää tarkempia ohjeita ja neuvoja. Suosituksia löytyy myös eri laitteita varten. (Android Developers 2019).

Kehitystä varten on myös luotu Android Studio, joka on Android-käyttöjärjestelmää varten suunniteltu ohjelmointiympäristö. Sen pohjana on IntelliJ IDEA ja se tarjoaa ohjelmointialustan ja kehittämistyökaluja. (Friesen 2020.) Android Studio käyttää ohjelmoinnissa Java-ohjelmointikieltä, mutta myös C#- ja C++-ohjelmointikielet ovat tuettuja (Sonmez 2016).

2.2 iOS

iOS on Applen luoma käyttöjärjestelmä mobiililaitteille. Se julkaistiin ensimmäisen iPhone-mallin kanssa vuonna 2007. Markkinoille tullessa käyttöjärjestelmästä puuttui monia perustoimintoja, kuten 3G-tuki, se ei tukenut ulkopuolisia sovelluksia, ominaisuus kopioida ja liittää tekstiä puuttui, kotinäkymää ei kyennyt muokkaamaan ja se ei tukenut multimediaviestejä. Apple keskittyi saamaan käyttökokemuksen hyväksi ja nopeaksi. Toki iOS sisälsi myös massiivisia innovaatioita, jotka muokkasivat mobiilimarkkinoita aina nykypäivään asti. (Bohn, Souppouris & Seifert 2013.)

iOS on suljetumpi kokonaisuus kuin Android, mutta se on alkanut sallia suurempia oikeuksia sovelluksille. Esimerkiksi sovellusten välinen tiedon jakaminen on nykyään sallittua ja sovelluksilla on lupa suorittaa taustalla. Itse lähdekoodi on kuitenkin suljettu. (Hoffman 2017.)

Apple myös tarjoaa kehittäjille opastusta ja suosituksia suunnittelun ja toiminnallisuuden kannalta. Suunnittelussa keskitytään selkeyteen, kunnioitukseen ja syvällisyyteen. Selkeydellä haetaan sovelluksesta mahdollisimman helppokäyttöistä ja kaikkien osien tulisi olla visuaalisesti helposti nähtävissä aina teksteistä kuviin. Kunnioituksella tarkoitetaan sovelluksen pitämistä mahdollisimman yksinkertaisena ja helposti lähestyttävänä, jossa tärkeät asiat ovat helposti huomattavissa. Syvyydellä luodaan käyttäjälle parempaa tuntumaa sovelluksen käytöstä ja saadaan sovellus tuntumaan laajalta kokonaisuudelta. (Apple Developer s.a.)

Apple on luonut kehittämistä varten Xcode-ohjelmointiympäristön. Xcode tarjoaa kehittäjiä varten paljon erilaisia työkaluja, kuten ohjelmointialustan, graafisen käyttöliittymän sovelluksen ulkonäön rakentamista varten ja versionhallinnan. Ohjelmointikielenä käytössä on Swift, mutta myös C-ohjelmointikielet ovat tuettuja. (Arvidsson 2018.)

2.3 Yleisesti

Molemmille alustoille on mahdollista kehittää sovelluksia ohjelmistojen omalla ohjelmointikielellä. Mutta tämä tarkoittaa, että koko sovelluksen joutuisi luomaa teoriassa kahdesti, koska käyttöjärjestelmät eivät ole yhteen sopivia. Myös kaikki kehitystyökalut ovat erilaisia. Siksi monesti kehityksessä käytetään järjestelmästä riippumattomia ohjelmia, jotka osaavat kääntää koodin molemmille alustoille. Pelien kohdalla pelimoottorit toimivat juuri tämän kaltaisina alustoina kääntäen pelin koodin toimimaan suoraan molemmille käyttöjärjestelmille. (Sonmez 2016.)

3 SUUNNITTELU

Tarkoituksena opinnäytetyössä oli luoda mobiilialustalla toimiva sovellus, joka julkaistaan molemmille Android- ja iOS -käyttöjärjestelmille. Sovellukseen luotiin kolme yksinkertaista peliä, joissa esitellään erilaisia keinoja luoda pelejä mobiilialustoille.

Suunnittelu on aina tärkeä osa projektia ja laatimalla tarkka suunnitelma vältetään ongelmia kehitysvaiheessa. Tämän projektin suunnittelussa valittiin käytettävät ohjelmistot ja mietittiin pelien sisältöä ja käyttöliittymää. Kaikki pelit suunniteltiin olevan 2D-näkymään, jotta sovelluksen sisältö pysyisi temaattisesti samankaltaisena.

3.1 Ohjelmat

Kehittämiseen valittiin Unity-pelimoottori, joka on yksi maailman käytetyimmistä pelimoottoreista. Yli 50 prosenttia uusista mobiilipeleistä on kehitetty Unity-pelimoottorilla ja sitä voi hyödyntää sekä 3D- että 2D-kehittämisessä. Myös useat isommat pelitalot hyödyntävät pelimoottoria kehityksessään. Unity tarjoaa monia kehittämistä helpottavia ominaisuuksia ja tukee monia eri alustoja, joihin lukeutuvat myös Android ja iOS. (Create and Monetize with Unity Mobile Games Development s.a.) Tässä opinnäytetyössä pelimoottorista käytetään versiota Unity 2019.2.15f1, joka on uusin julkaistu versio opinnäytetyön pelien kehityksen aloituksen aikaan.

Ohjelmointiin kieleksi valittiin C#, joka on Unity-pelimoottorissa tuettu kieli. C# mahdollistaa olio-ohjelmoinnin. (Coding in C# in Unity for beginners s.a.) Olio-ohjelmoinnissa rakennetaan yhteenkuuluvista ohjelmointikomponenteista eli luokista kokoelma. Ohjelmoinnista saadaan näin yksinkertaisempaa, kun koko ohjelmaan ei tarvitse rakentaa suoraan toiminnalliseksi, vaan voidaan rakentaa toimivia luokkia, joista kootaan koko ohjelma. (Hietanen 2003, 6–7.)

2D-kuvakulman takia pelien grafiikka piirretään. Sitä varten piirtotyyliksi valittiin pikselitaide. Pikselitaide näyttää hyvältä ja se on helpompaa oppia varsinkin ohjelmoijalle. Pikselitaiteella tarkoitetaan piirtotyyliä, jossa painotetaan jokaisen pikselin merkitystä kuvassa. Piirtäessä värjätään yksittäisiä pikseleitä ja näin muodostetaan valmis kuva. (Kotaki 2012.)

Piirtämiseen valittiin selaimessa toimiva ilmainen sivusto pixelart.com. Palvelu antaa hyvän pohjan luoda piirroksia ja sisältää monia hyödyllisiä ominaisuuksia, kuten erilisiä tasoja ja työkaluja. Vaikka palvelu rajoittaa piirtämisen aikana piirroksen koon 700x700 pikseliin, saa ohjelmasta kuvan kuitenkin ulos maksimissaan 40 kertaa suurempana ja myös koko piirroksen projektitiedoston voi ladata.

Pelin rakentamiseen tarvitaan molemmille alustoille omat ohjelmansa. Android-käyttöjärjestelmää varten ohjelma on Android Studio. Unity ei kuitenkaan vaadi koko Android Studio -ohjelmointiympäristön asentamista, vaan Unity voi ladata automaattisesti sovelluksen rakentamista varten tarvittavat ohjelmistot. Tämän jälkeen Unity voi suoraan rakentaa projektin APK-tiedoston, joko tietokoneelle tai suoraan Android-laitteeseen. (Android environment setup 2020.)

iOS-käyttöjärjestelmälle rakentamiseen vaaditaan Xcode. Xcode on saatavissa Applen tietokoneille. Applen tietokone on siis välttämätön kehitettäessä iOS-käyttöjärjestelmälle. Unity ei linkity samalla tavalla Xcode-ohjelmointiympäristön kanssa, vaan Unity rakentaa Xcode-projektitiedoston, jonka avulla Xcode osaa rakentaa pelin mobiilialustalle.

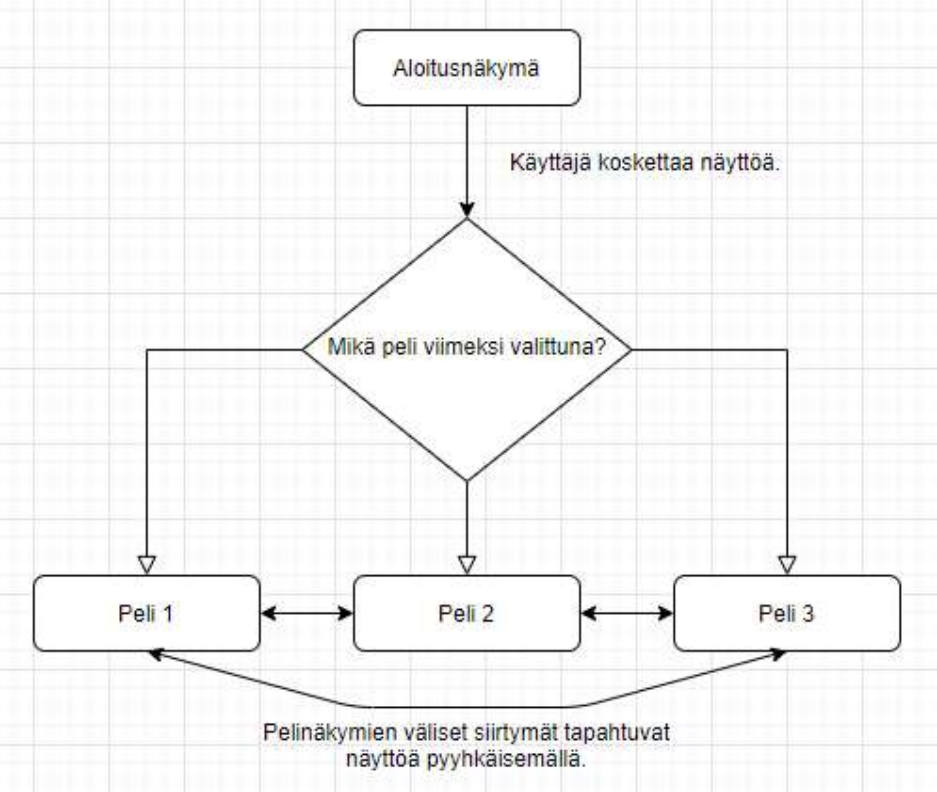
3.2 Kehitysvaiheen suunnittelu

Opinnäytetyön lähtökohtana oli kolmen pelin kehittäminen. Peleissä tultaisiin hyödyntämään aiemmista projekteista tuttuja asioita, jotta kehitysvaiheen toteutuksesta tulisi sujuvampaa. Tarkoituksena peleissä olisi tuoda esiin erilaisia kontrollimenetelmiä. Opinnäytetyössä kaikkien pelien kontrollit luetaan koskетusnäytöltä, mutta tarkoituksena on esitellä erilaisia tapoja hyödyntää koskетusnäytön ominaisuutta.

3.2.1 Valikot

Sovellus avautuu pelin alkunäkymään, joka toimii valikkona eri pelien välillä. Valikon toiminnan haluttiin olevan mahdollisimman yksinkertainen, joka otettiin huomioon sitä suunniteltaessa. Alkuvalikon toimintaan haluttiin luoda selkeä ilme ja nopea toiminnallisuus.

Valikon toiminta on selkeä (kuva 1). Käyttäjälle ensin avautuu aloitusnäkymä, jossa on sovelluksen nimi ja kuva, joka kuvaa sovelluksen sisältöä. Tästä näkymästä pääsee näyttöä koskettamalla pelinvalintaan. Jokaiselle kolmesta pelistä on oma valintanäkymä ja sovellus muistaa, mitä peliä käyttäjä on viimeiseksi pelannut ja avaa kyseisen valintanäkymän. Valintanäkymä tulee sisältämään kuvauksen ja kuvan pelistä ja painikkeen, jota painamalla pääsee itse peliin. Pelinäkymästä toiseen pääsee näyttöä pyyhkäisemällä.



Kuva 1. Valikon toiminnan kuvaus

Pelien sisäisen valikon käyttämisen halutaan myös olevan mahdollisimman yksinkertaista. Käyttäjälle avautuu näkymä, jossa on helposti nähtävissä pelin parhaat tulokset ja niiden alla kaksi painiketta, joista toinen aloittaa pelin ja toinen palauttaa pelaajan takaisin alkuvalikkoon. Ruudun oikeasta ylälaidasta löytyy myös painike, josta käyttäjä saa ohjeet pelin pelaamista varten.

3.2.2 Peli 1

Ensimmäisenä pelinä oli tarkoitus luoda endless runner, jossa tarkoituksena on selvitä mahdollisimman pitkään ja samalla kerätä pisteitä. Tämän tyyppiset pelit ovat yleisiä mobiilipelimarkkinoilla.

Pelissä pelaaja yrittää selvitä mahdollisimman pitkään väistäen esteitä. Mekaniikkana tähän valittiin painovoiman vaihtaminen. Pelaaja voi siis koskettamalla näyttöä vaihtaa painovoiman suunnan ja näin väistää esteitä. Pelihahmo pystyy liikkumaan ruudun ylä- ja alalaidassa. Pelin pisteytys toimii perustuen aikaan, jonka pelaaja onnistuu pelissä selviämään. Tämä aika kerrotaan kymmenellä ja näin saadaan varsinaiset pisteet.

3.2.3 Peli 2

Toiseksi peliksi valittiin myös samaan tapaan yleinen mobiilipelityyppi yhdistä 3. Kyseisessä pelissä pelaajalla on ruudukko täynnä erilaisia muotoja ja pelaajan tarkoituksena on muodostaa vähintään kolmen suoria näistä muodoista. Pelaaja ansaitsee pisteitä jokaisesta muodosta, joka poistuu ruudukosta.

Peliin päätettiin myös lisätä aikaraja, jotta siinä voi helposti vertailla tuloksia. Aikarajaksi valittiin minuutti. Pisteytystä muokattiin myös siten, että pelaaja ansaitsee kertoimia pisteiden saantiin pelaamalla mahdollisimman nopeasti. Tämä aiheuttaa enemmän jakaumaa lopullisissa pisteissä.

Peliin suunniteltiin kaksi tapaa kontrolloida peliä. Ensimmäisenä ideana oli tehdä kosketuksella toimiva malli. Tässä pelaaja koskettaa tiettyä muotoa ja sen jälkeen koskettaa tämän muodon viereistä muotoa ja niiden paikkoja vaihdetaan. Toinen ratkaisu soveltaa aiempaa ja siinä pelaaja koskettaa haluamaansa muotoa ja pyyhkäisee haluamaansa suuntaan ja muoto vaihtaa paikkaa siinä suunnassa olevan muodon kanssa.

3.2.4 Peli 3

Kolmannen pelin suunnittelussa haluttiin luoda peli, joka tarjoaa erilaisen kontrollin pelaajalle. Peli kuitenkin haluttiin pitää yksinkertaisena ja siksi se muistuttaa ensimmäistä peliä, sillä siinä ideana on selvitä mahdollisimman pitkään. Pelissä pelaaja ohjaa alusta ja ampuu vastustajia. Pisteiden keräys tapahtuu tuhoamalla muita aluksia.

Pelin suurin ero Peliin 1 on kontrollit, jotka suunniteltiin toimivan sauvaohjaimen tapaisella toiminnolla, jossa ruudulle piirretään ohjain pelaajan koskettamaan kohtaan. Pelaajan ampuminen taas toimii näyttöä oikeasta kohtaa koskettamalla.

4 KEHITYS

Sovelluksen kehityksessä päätettiin valikon toimivan pystytasossa. Pelien kohdalla sovellus vaihtaa vaakatason ja pystytason välillä riippuen pelistä. Toiminta ei ole käyttäjäystävällinen ja normaalisti vaihtoa kuvan tason välillä kannattaa välttää käyttäjäkokemuksen parantamiseksi, mutta kyseessä on kokoelma pelejä, jolloin valikko on vain keino päästä peleihin.

Valikkoon ja peleihin luotiin koodi, joka katsoo, painetaanko Escape-painiketta. Escape-painiketta vastaa Android-laitteissa Back-painike. Alkuvalikossa toiminto sulkee sovelluksen ja pelien sisällä se palauttaa pelin alkunäkymään. iOS-laiteissa vastaava toiminto on Home-painikkeessa ja se on toiminnassa automaattisesti, jollei sitä sovelluksen puolelta estä. Painike tosin vie käyttäjän aina pois sovelluksesta.

4.1 Käyttöliittymän kehitys

Käyttöliittymän kehitys on yksi vaativimmista osa-alueista mobiilialustat huomioon ottaen. Mobiilille kehittäessä on otettava huomioon erilaiset laitteet ja varsinkin niiden näytöt. Tablettien ja älypuhelimien näyttöjen resoluutiot ovat hyvin erilaiset ja näin ollen sovelluksen pitää muokata omaa tyyliään eri laitteiden näyttöihin sopivaksi (Kuva 2).

Apple Products

	Pixel Size	Viewport
iPhone		
iPhone XR	828 x 1792	414 x 896
iPhone XS	1125 x 2436	375 x 812
iPhone XS Max	1242 x 2688	414 x 896
iPhone X	1125 x 2436	375 x 812
iPhone 8 Plus	1080 x 1920	414 x 736
iPhone 8	750 x 1334	375 x 667
iPhone 7 Plus	1080 x 1920	414 x 736
iPhone 7	750 x 1334	375 x 667
iPhone 6 Plus/6S Plus	1080 x 1920	414 x 736
iPhone 6/6S	750 x 1334	375 x 667
iPhone 5	640 x 1136	320 x 568
iPod		
iPod Touch	640 x 1136	320 x 568
iPad		
iPad Pro	2048 x 2732	1024 x 1366
iPad Third & Fourth Generation	1536 x 2048	768 x 1024
iPad Air 1 & 2	1536 x 2048	768 x 1024
iPad Mini 2 & 3	1536 x 2048	768 x 1024
iPad Mini	768 x 1024	768 x 1024

Android Devices

	Pixel Size	Viewport
Phones		
Nexus 6P	1440 x 2560	412 x 732
Nexus 5X	1080 x 1920	412 x 732
Google Pixel 3 XL	1440 x 2960	412 x 847
Google Pixel 3	1080 x 2160	412 x 824
Google Pixel 2 XL	1440 x 2560	412 x 732
Google Pixel XL	1440 x 2560	412 x 732
Google Pixel	1080 x 1920	412 x 732
Samsung Galaxy Note 9	1440 x 2960	360 x 740
Samsung Galaxy Note 5	1440 x 2560	480 x 853
LG G5	1440 x 2560	480 x 853
One Plus 3	1080 x 1920	480 x 853
Samsung Galaxy S9+	1440 x 2960	360 x 740
Samsung Galaxy S9	1440 x 2960	360 x 740
Samsung Galaxy S8+	1440 x 2960	360 x 740
Samsung Galaxy S8	1440 x 2960	360 x 740
Samsung Galaxy S7 Edge	1440 x 2560	360 x 640
Samsung Galaxy S7	1440 x 2560	360 x 640
Tablets		
Nexus 9	1536 x 2048	768 x 1024
Nexus 7 (2013)	1200 x 1920	600 x 960
Samsung Galaxy Tab 10	800 x 1280	800 x 1280
Chromebook Pixel	2560 x 1700	1280 x 850

Kuva 2. Listaus erilaisista näytön resoluutioista iOS- ja Android-laitteissa (Popular Screen Resolutions: Designing for All 2020).

Opinnäytetyöhön luotiin skaalautuva käyttöliittymä, joka muokkautuu laitteen resoluution mukaan. Skaalauksen kannalta tärkeintä resoluutiossa on kuvasuhde. Nykyaikaisissa älypuhelimissa näyttöjen muotoon on myös tehty muutoksia ja tämän kaltaiset muutokset, kuten kaareva näyttö, aiheuttavat ongelmia sovelluksen skaalaamiseen. Kaarevassa näytössä ruudun reuna on edelleen osa näyttöä, mutta se ei ole järkevästi hyödynnettävissä pelialueena tai käyttöliittymän puolelta.

Unity tarjoaa käyttöliittymän kehitykseen hyviä työkaluja, joista tärkeimpänä on editorissa oleva pelinäkymä. Pelinäkymään voi valita haluamansa resoluution ja siten voidaan nähdä, miltä sovellus näyttää halutussa resoluutiossa.

Valikossa kontrolleina toimivat kosketus ja pyyhkäisy. Kosketus toteutetaan käyttäen Unity-pelimoottorin UI-objektien ominaisuutta, jossa niihin voidaan luoda tapahtumia. Tapahtumia on erilaisia, kuten UI-objektin painaminen, raahaus ja liikkuminen. Tähän tapahtumaan voidaan sitoa suoraan koodin funktio

tai objektin komponentin ominaisuus. Opinnäytetyössä kaikki painikkeet käyttöliittymässä toimivat tämän ominaisuuden kautta. Myös alkunäkymä saadaan suljettua tapahtumien kautta.

Pyyhkäisy on puolestaan rakennettu koodissa. Pyyhkäisyn toiminta mittaa ensimmäisen kosketuksen paikkaa verraten sitä kosketuksen loppumispaikkaan. Ohjelma sitten laskee, oliko pyyhkäisy mihin suuntaan ja oliko se tarpeeksi pitkä (kuvat 3–5).

```
// Mobile Input
if (Input.touches.Length > 0)
{
    if (Input.touches[0].phase == TouchPhase.Began)
    {
        tap = true;
        isDragging = true;
        startTouch = Input.touches[0].position;
    }
    else if (Input.touches[0].phase == TouchPhase.Ended
        || Input.touches[0].phase == TouchPhase.Canceled)
    {
        isDragging = false;
        Reset();
    }
}
```

Kuva 3. Pyyhkäisyn alkaminen koodissa

Koodi ensimmäiseksi tarkistaa kosketuksen aloituksen ja sitten luo vektorin ensimmäisen ja viimeisen kosketuksen välille (Kuva 4). Sen jälkeen tarkastetaan vektorin pituus, jota verrataan arvoon, joka on todettu hyväksi pyyhkäisyn minimiarvoksi.

```
// Calculate distance of start of input and end
swipeDelta = Vector2.zero;
if (isDragging)
{
    if (Input.touches.Length > 0)
    {
        swipeDelta = Input.touches[0].position - startTouch;
    }
    else if (Input.GetMouseButton(0))
    {
        swipeDelta = (Vector2)Input.mousePosition - startTouch;
    }
}
```

Kuva 4. Pyyhkäisyn jatkuessa lasketaan vektorin pituutta

Jos vektori on tarpeeksi pitkä, selvitetään sen suunta. Suunnan selvittämiseksi verrataan x- ja y-akselin itseisarvoa toisiinsa (Kuva 5).

```
// Was it swipe?
if (swipeDelta.magnitude > 150)
{
    float x = swipeDelta.x;
    float y = swipeDelta.y;
    if (Mathf.Abs(x) > Mathf.Abs(y))
    {
        if (x < 0)
        {
            swipeLeft = true;
        }
        else
        {
            swipeRight = true;
        }
    }
    else
    {
        if (y < 0)
        {
            swipeDown = true;
        }
        else
        {
            swipeUp = true;
        }
    }


    Reset();
}
```

Kuva 5. Pyyhkäisyn loppuessa selvitetään sen suunta

X-akselin ollessa suurempi pyyhkäisyn suuntana on oikea, jos arvo positiivinen, ja vasen, jos arvo on negatiivinen. Y-akselilla positiivisella arvolla suunta on ylöspäin ja negatiivisella arvolla suunta on alaspäin. Jokaiselle suunnalle on luotu oma boolean-muuttuja, jonka avulla muut ohjelman koodit saavat tietää pyyhkäisyn tapahtuneen.

4.2 Pelien kehitys

Kuten käyttöliittymän kehityksessä, myös pelien kohdalla näyttöjen koko vaikuttaa pelin pelattavuuteen. Erilaisilla näytöillä pelialueen koko voi vaihdella ja pelaaja ei välttämättä näe kaikkea tarpeellista. Tämä tarkoittaa pelien kohdalle kameran koon skaalaamista näytön mukaan. Tämä aiheuttaa myös ongelmia pelien ulkonäköön, koska joillain näytöillä pelialueen ulkopuolelle näkee paljon

paremmin. Tosin toisin kuin käyttöliittymä, jonka muokkausta pystyy tutkia helpommin editorissa, pelien testaaminen eri näyttöresoluutioissa vaatii paljon enemmän. Koska koon muokkaus tapahtuu koodissa, se tutkii käytettävän laitteen näyttöä ja kehitysvaiheessa se on monesti tietokone, joka ei vastaa mobiililaitetta.

2D-kuvakulmassa kamera asetetaan ortografiseen kuvaan. Ortografinen kamera eroaa normaalista perspektiivikamerasta siten, että kamera ei välitä objektien etäisyydestä toisiinsa tai kameraan, jolloin kaikki näkymässä vaikuttaa olevan samalla etäisyydellä kamerasta (Camera 2020). Kameran ominaisuus orthographicSize määrittää kuvassa näkyvän alueen koon. Ominaisuuden arvo on desimaaliluku, joka on puolet näkymän korkeudesta. Horisontaalinen koko taas riippuu kuvasuhteesta. (Camera.ortographicSize 2020.) Säädettäessä kameran kokoa halutaan määrittää kameran näkymän leveys. Kun leveys on pakotettu, kameran korkeus vaihtelee verrattuna laitteen näytön kuvasuhteeseen.

Opinnäytetyön jokaisesta pelistä löytyy myös aloitusnäkymä, jossa on listattuna pelaajan viisi parasta tulosta pelissä (Kuva 6). Tulokset ovat laitekohtaiset.

Kuva 6. Pelin aloitusnäkymä

Näkymässä on myös painikkeet ohjeita, uudelleen pelaamista ja alkuvalikkoon palaamista varten. Näkymä on samanlainen kaikissa peleissä värin vaihtoa lukuun ottamatta.

4.2.1 Peli 1

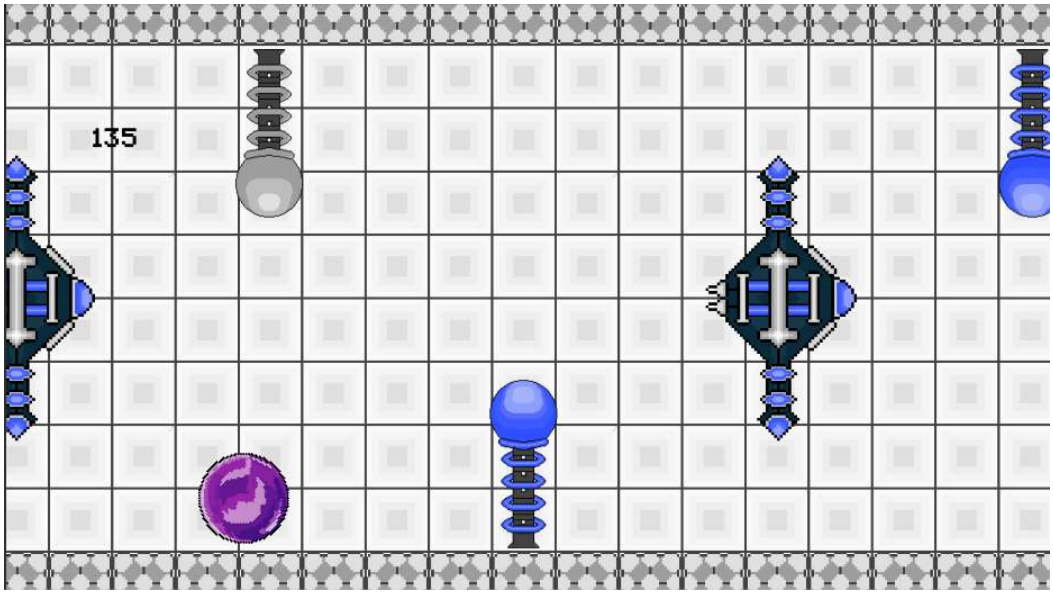
Kyseessä oli siis yksinkertainen peli, jossa pelaaja vaihtaa painovoiman suuntaa ja yrittää selviytyä mahdollisimman pitkään. Pelin kuvakulma on vaakatasossa, jotta pelaajalla on mahdollisuus nähdä mahdollisimman pitkälle pelialueella.

Pelin toiminnasta haluttiin mahdollisimman yksinkertainen ja ainoa kontrolli, joka pelaajalta otettiin, on ruudun kosketus. Kosketus luotiin ohjelmoimalla hiiren vasemman näppäimen painallus (Kuva 7), joka vastaa mobiililaitteella näytön kosketusta. Toiminto valittiin pelin testaamisen helpottamiseksi, koska nyt testaamista voitiin tehdä helposti myös tietokoneella ilman erillisen koodin luontia sitä varten.

```
if (Input.GetButtonDown("Action"))
{
    rb2D.gravityScale = -rb2D.gravityScale;
    rotationSpeed = -rotationSpeed;
}
rb2D.velocity = new Vector2(0, rb2D.velocity.y);
transform.Rotate(new Vector3(0, 0, rotationSpeed));
```

Kuva 7. Pelissä 1 pelaajan gravitaation vaihtaminen

Pelissä pelaaja pystyy kulkemaan joko pelialueen ylä- tai alareunassa. Pelissä pelaajan tielle luodaan esteitä, joiden luominen tapahtuu koodissa siten, että kaksi peräkkäistä estettä ei voi olla samoja. Esteitä on kolme erilaista: alareunan este, yläreunan este ja keskellä oleva este (Kuva 8). Pelaajan on väistettävä esteitä vaihtamalla painovoiman suuntaa.



Kuva 8. Pelin 1 pelikuvaa

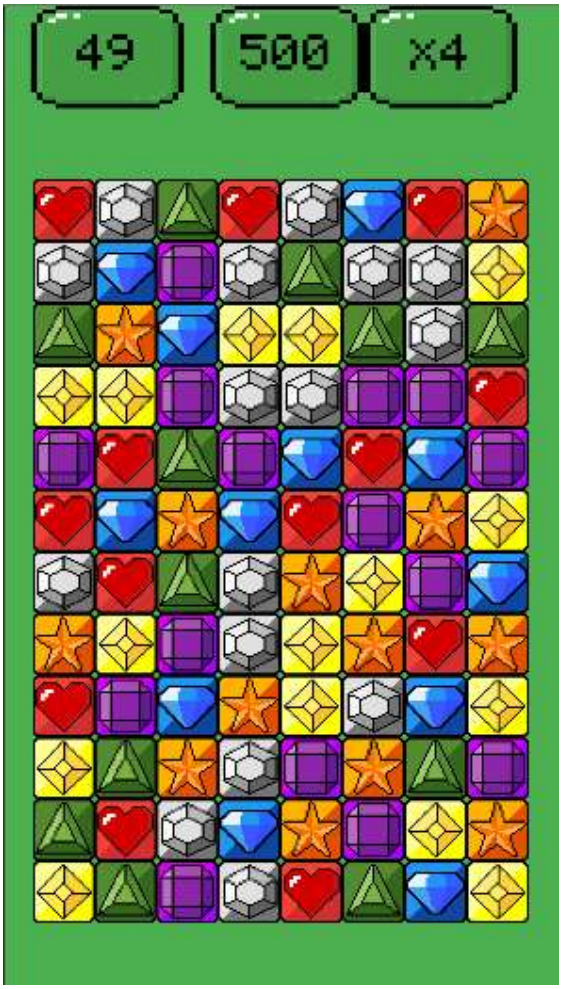
Pelaaja hahmo ei oikeasti liiku eteenpäin kentällä, vaan kenttä liikkuu pelaajaa kohden luoden vaikutelman liikkumisesta. Vaikutelmaa lisää pelaajahahmon pyöriminen, joka tapahtuu koodissa ja pyörimissuuntaa vaihdetaan painovoiman vaihdoksen ohessa. Pelaaja oikeasti liikkuu ainoastaan y-akselilla. Kentän liikkumisnopeus ja pelaajaan vaikuttava painovoima kasvavat pelin edetessä tehden pelaamisesta vaikeampaa.

Pisteytys pelissä tapahtuu verrattuna peliaikaan. Pelaaja kerää pisteitä selvitessään pidempään ja selviämisaika kerrotaan kymmenellä ja pyöristetään tasaluvuksi. Peli loppuu, kun pelaaja osuu esteeseen. Silloin hänet palautetaan pelin aloitusnäkymään.

4.2.2 Peli 2

Pelissä pelaajalle luodaan 8 x 12 -ruudukko, joka muodostuu laatoista, joissa on kuvana eri muotoja. Laattoja siirtämällä pelaaja yrittää muodostaa vähintään kolmen suoria pysty- tai vaakariveille. Peli on ainoa opinnäytetyön peleistä, jonka kuvakulma on pystysuora ja näin ollen se on ainoa peli, jonka kameraan ei ole tehty pakotettua leveyttä. Pystysuorassa ortografinen kamera on helpommin oikeassa koossa, koska sen näkymän koko suoraan vaikuttaa korkeuteen.

Ruudukon jokaisessa ruudussa on laatta, jossa on kuvana jokin 7 erilaisesta muodosta (Kuva 9). Ruudukon luonnin aikana kaikille laatoille arvotaan muoto, mutta ruudukon luonnissa varmistetaan, että ruudukkoon ei muodostu kolmen suoria.



Kuva 9. Pelin 2 pelikuvaa

Kun ruudukko on luotu peli alkaa, kontrolleina pelaajalle tarjotaan kahta vaihtoehtoa: painallukset ja pyyhkäisy. Kummassakin vaihtoehdossa pelaajan on valittava ensimmäisellä painalluksella haluamansa laatta. Jos pelaaja käyttää painalluksia, hänen irrottaessaan kosketuksen näytöltä laatta jää edelleen valituksi. Seuraavaksi hän valitsee seuraavan laatan, jos laatta on aiemmin valitun laatan vieressä pysty- tai vaakarivillä, niiden paikkaa vaihdetaan. Jos valittu laatta ei ole aiemmin valitun vieressä, vaihtuu uusi laatta valituksi. Painallus on toteutettu Unity-pelimoottorin suoralla funktiolla OnMouseDown() (Kuva

10), jota voidaan kutsua objektista, jolla on törmäyksen tunnistukseen käytettävä komponentti, tai se on UI-objekti, ja se tapahtuu, kun objektia painetaan hiirellä tai kosketetaan näytöllä (MonoBehaviour.OnMouseDown() 2020).

```
void OnMouseDown()
{
    // 1
    if (render.sprite == null || BoardManager.instance.IsShifting || !Started)
    {
        return;
    }

    if (isSelected)
    { // 2 Is it already selected?
        Deselect();
    }
    else
    {
        if (previousSelected == null)
        { // 3 Is it the first tile selected?
            Select();
        }
        else
        {
            if (GetAllAdjacentTiles().Contains(previousSelected.gameObject))
            { // 1
                SwapSprite(previousSelected.render); // 2
                previousSelected.ClearAllMatches(false, gameObject);
                temp = previousSelected;
                previousSelected.Deselect();
                ClearAllMatches(true, gameObject);
            }
            else
            { // 3
                previousSelected.GetComponent<ConnectTile>().Deselect();
                Select();
            }
        }
    }
}
```

Kuva 10. OnMouseDown() funktio pelissä 2

Pyyhkäisykontrolli on sama kuin valikossa, mutta pelissä se muuttaa pelitapaa. Kun käytetään pyyhkäisyä, ensimmäistä painallusta näytölle ei vapauteta. Koska laatta valitaan heti ensimmäisestä kosketuksesta, pelaaja voi sitten pyyhkäistä haluamaansa suuntaan ja valittu laatta vaihtaa paikka viereisen laatan kanssa, jos se on mahdollista. Laattojen vaihtaminen ei ole mahdollista, jos valittu laatta on reunimmaisin laatta ruudukossa pyyhkäisyn suuntaan.

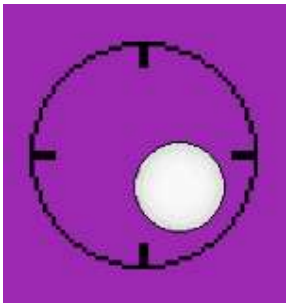
Peliin on rakennettu tekoäly, joka tarkastaa jokaisen siirron jälkeen, onko ruudukossa mahdollisia siirtoja. Jos siirtoja ei ole, ruudukko luodaan uudestaan, mutta tällä kertaa ei välitetä kolmen suorien syntymisestä. Tekoäly käy ruudukkoa läpi 3 x 3 ruudun alueissa, kunnes se löytää siirron. Löytyessään siirto tallennetaan ja jos pelaaja ei ole tehnyt siirtoa kahteen sekuntiin, pelaajalle näytetään vinkkinä siirto, joka löydettiin. Vinkki tapahtuu laatan kuvan värin vaihtamisella nopeasti valkoisen ja harmaan välillä.

Pelin pisteytys tapahtuu siirtojen avulla. Jokaisesta kolmen suorasta saa 20 pistettä ja ylimääräisistä laatoista suorassa tulee 10 pistettä lisää. Pisteytyksessä on mukana myös kerroin, joka kasvaa pelaajan kerätessä pisteitä mahdollisimman nopeasti. Maksimi kerroin on viisi ja kertoimet kasvavat, jos pelaaja saa neljä kolmen suoraa tehtyä kertoimen ajan sisällä. Kertoimien kasvu lasketaan pisteissä, joten suurempien suorien tekeminen kasvattaa sitä nopeammin. Pelissä on aikarajana minuutti ja peli loppuessa pelaaja palautuu pelin alkunäkymään.

4.2.3 Peli 3

Pelissä pelaaja ohjaa avaruusalusta ja hänen tehtävänään on selviytyä mahdollisimman kauan ja tuhota mahdollisimman monta vastustajaa. Pelaaja pystyy liikkumaan ja ampumaan.

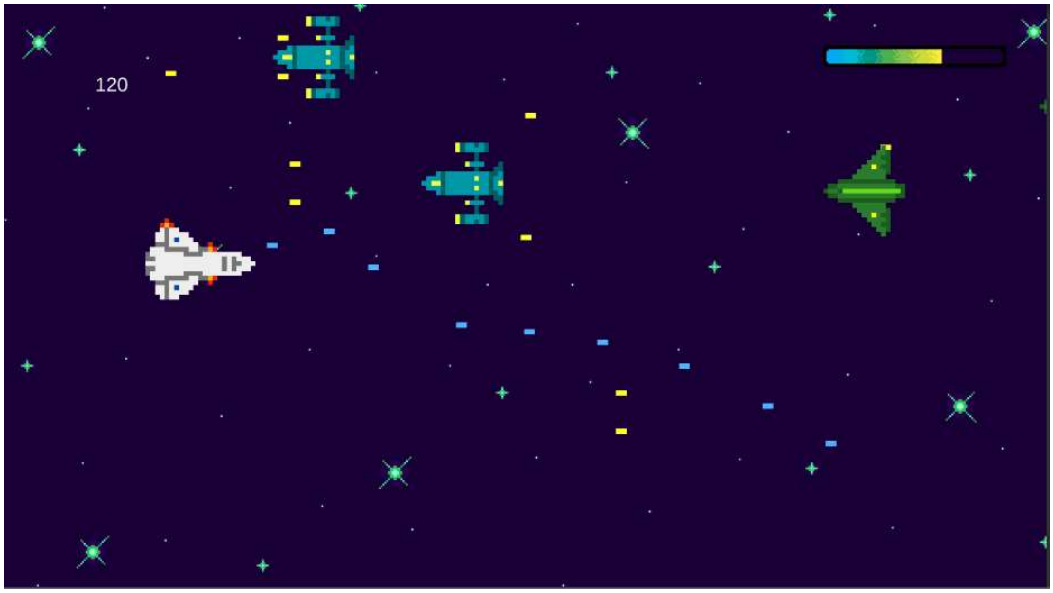
Liikkuminen tapahtuu koskettamalla näytön vasempaan laitaan. Kosketuksesta pelaajalle luodaan näytölle konsolien ohjaimista tuttua analogista sauvaa muistuttava sauvaohjain (Kuva 11). Sauvaohjaimessa on kaksi ympyrää, joista ulompi toimii sauvan keskikohtana ja se luodaan ensimmäisen kosketuksen osoittamaan paikkaan, ja sisempi liikkuu kosketuksen suuntaan, jotta pelaaja näkee, mihin hän on ohjaamassa alusta.



Kuva 11. Ruudulle luotavan sauvaohjaimen kuva.

Sisemmän ympyrän liike on rajoitettu ulomman reunoihin. Liikkumista varten ympyröiden nykyisestä sijainnista muodostetaan vektori ja se muunnetaan yksikkövektoriksi. Tätä vektoria käytetään sitten pelaajan liikkeen suuntana.

Ampuminen tapahtuu UI-objektina olevasta painikkeesta, joka on levitetty kamera näkymän oikeaan reunaan. Painikkeesta painamalla alus ampuu luodin. Painike on tehty näkymättömäksi, jotta se ei häiritse pelaamista. Ampumiseen on luotu myös aseen ylikuumeneminen, joka tarkoittaa, että aseella ei voi ampua loputtomiin, vaan aseen ylikuumetessa sillä ei voi ampua. Ylikuumenemisen huomaa näkymän ylälaidasta löytyvästä mittarista (Kuva 12). Mittari täyttyy aseen kuumetessa.



Kuva 12. Pelin 3 pelikuvaa

Kontrollien kannalta peliin jouduttiin tekemään enemmän töitä. Pelissä toinen kosketus on koko ajan näytöllä ohjaamassa alusta, joten ampumiseen käytettävä kosketus ei saa vaikuttaa aluksen ohjaukseen. Normaalisti näytön koskettaminen kahdella sormella luetaan yhdeksi kosketukseksi, jonka paikaksi lasketaan kosketusten välinen keskikohta. Tämän takia peliin jouduttiin luomaan koodi, joka lukee vain ensimmäistä kosketusta (Kuvat 13–15).

```
if (!IsPointerOverUIObject() && Input.touchCount > 0)
{
    Touch t = Input.GetTouch(0);
    if (t.phase == TouchPhase.Began)
    {
        pointA = Camera.main.ScreenToWorldPoint(
            new Vector3(t.position.x, t.position.y, Camera.main.transform.position.z));

        circle.position = pointA;
        outerCircle.position = pointA;
        circle.GetComponent<SpriteRenderer>().enabled = true;
        outerCircle.GetComponent<SpriteRenderer>().enabled = true;
    }
```

Kuva 13. Kosketuksen aloitus koodissa

Koodi tutkii kosketuksen aloitus kohtaa muuttaen sen koordinaateiksi pelimaailmassa. Samalla sauvaohjain liikutetaan pisteeseen ja laitetaan näkyväksi pelaajalle.

```
else if (t.phase == TouchPhase.Ended)
{
    touchStart = false;
    ShutEngines();
    circle.GetComponent<SpriteRenderer>().enabled = false;
    outerCircle.GetComponent<SpriteRenderer>().enabled = false;
}
else if (t.phase == TouchPhase.Moved)
{
    touchStart = true;
    pointB = Camera.main.ScreenToWorldPoint(
        new Vector3(t.position.x, t.position.y, Camera.main.transform.position.z));
}
```

Kuva 14. Kosketuksen lopetus ja kosketuksen liikkuminen koodissa.

Kosketuksen jatkuessa toiseen muuttujaan tallennetaan nykyisen sijainnin koordinaatti, jota käytetään liikkumisen suunnan määritykseen. Kun kosketus irrotetaan näytöltä, sauvaohjain piilotetaan.

Varmistuksena, että ampumisen kosketus ei vaikuttaisi ohjaamiseen, luotiin myös koodiin funktio, joka estää ohjaamiseen tarkoitetun koodin aktivoinnin, jos kosketus tapahtuu UI-objektille (Kuva 15).

```
private bool IsPointerOverUIObject()
{
    PointerEventData eventDataCurrentPosition = new PointerEventData(EventSystem.current);
    eventDataCurrentPosition.position = new Vector2(Input.mousePosition.x, Input.mousePosition.y);
    List<RaycastResult> results = new List<RaycastResult>();
    EventSystem.current.RaycastAll(eventDataCurrentPosition, results);
    return results.Count > 0;
}
```

Kuva 15. Funktio, joka tutkii, onko kosketus UI-objektin päällä

Funktiossa katsotaan, sijaitseeko kosketus UI-objektin päällä. Funktio palauttaa boolean-muuttujan, jota käytetään sitten ehtona ohjaamisen kosketukseen (Kuva 13).

Pisteytys pelissä on yksinkertainen, kun pelaaja ampuu vastustajan, hän saa 10 pistettä. Vastustaja aluksia on kahta eri tyyppiä. Toinen lentää suoraan vaakatasossa vasemmalle ja toinen liikkuu vasemmalle ja samalla näytön ylä- ja alareunan välillä. Pelissä on myös asteroidi, jota pelaaja ei pysty tuhoamaan. Se luotiin pakottamaan pelaaja liikkumaan pelialueella. Asteroidi liikkuu suoraan vasemmalle vaakatasossa. Peli loppuu pelaajan osuessa vastustajan luoteihin, vastustajaan tai asteroidiin. Pelaaja palautetaan pelin alkunäkymään pelin loppuessa.

4.3 Testaus

Testauksella tarkastetaan sovelluksen toimivuutta, mutta testaamista käytetään myös pienien osien toiminnan tarkastamiseen. Kaikki mitä sovellukseen tehdään, tulisi testata. Jokaisen osan testaaminen heti estää ongelmien tuomista jo toimivaan sovellukseen. Mitä enemmän testaamista toteutetaan, sitä enemmän ongelmia, saadaan huomioitua. (Blundell & Milano 2015, 29–31.)

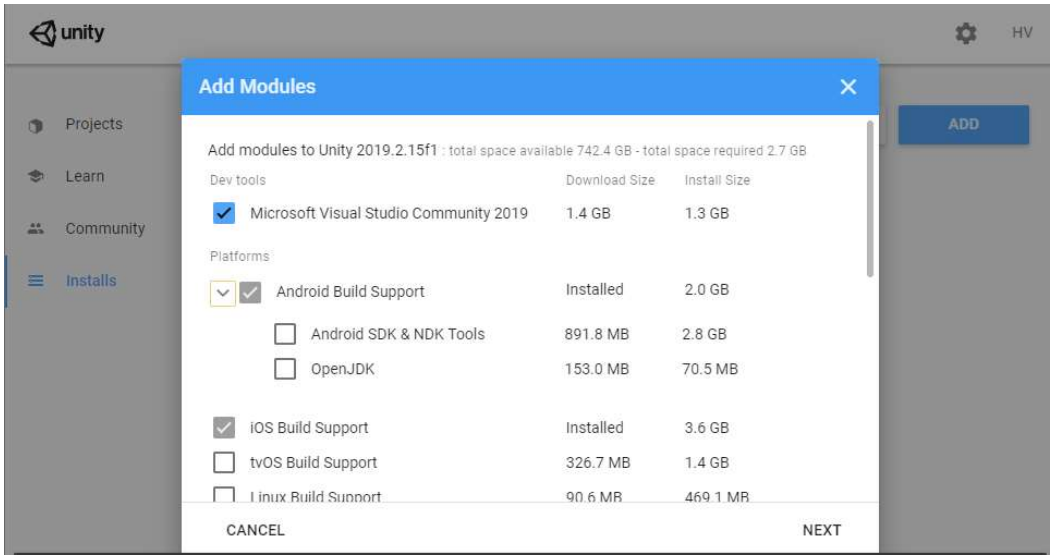
Unity tarjoaa helpon tavan testata varsinaista koodin toiminnallisuutta omassa editorissaan. Editori voi emuloida pelejä suoraan Unity-pelimoottorin ikkunassa ja näin ollen kaikkea testaamista varten sovellusta ei tarvitse rakentaa.

Tärkeä osa testaamisessa mobiilialustalla on erilaisten laitteiden kanssa sovelluksen testaus. Eri laitteiden kanssa halutaan nähdä käyttöliittymään ja peleihin tehtyjen skaalausten toimivuus. Pääasiallisina testilaitteina opinnäytetyössä olivat Android-alustalla Sony Xperia XZs ja iOS-alustalla iPad Air 2. Testilaitteissa on suurena hyötynä näyttöjen resoluution suuri ero, koska iPad on tabletti ja Xperia on kännykkä. Opinnäytetyössä pääasiallisena testausalustana on toiminut Android, koska ohjelmointiin käytettiin Windows-tietokonetta ja sillä ei voida rakentaa suoraan iOS-laitteelle.

4.3.1 Android

Androidin testaaminen on tehty yksinkertaiseksi Unity-pelimoottorissa. Unity pystyy linkittämään itsensä Android Studio -ohjelmointiympäristön tarvittavien osien kanssa. Testausta varten tarvitaan vähintään JDK- ja Android SDK -työkalut. Aiemmin Android NDK -työkalut olivat vaihtoehtoisia, mutta 1.8.2019 jälkeen julkaisua varten tarvitaan 64-bittinen versio sovelluksesta ja Android NDK -työkalut ovat siis nykyään julkaisua varten pakollisia (Leonard 2019).

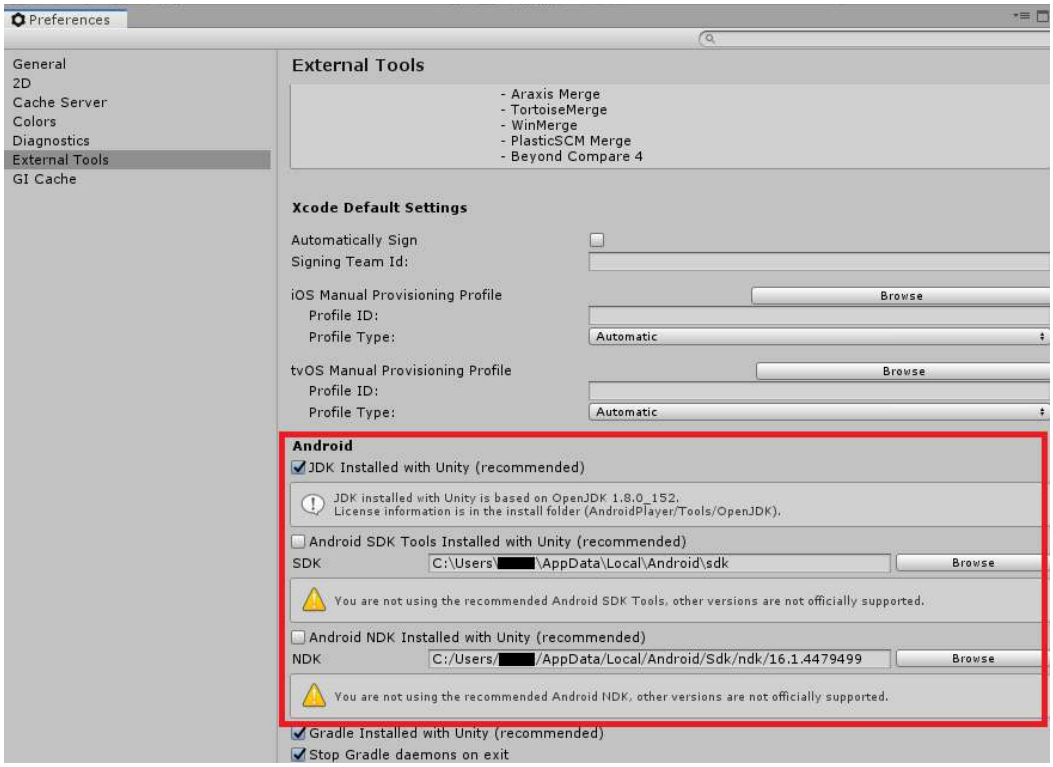
Unity-pelimoottoria asentaessa voi ladata mukaan Android-kehitystyökalut. Nämä pitää erikseen valita normaalin Android-kehittämistuen lisäksi (Kuva 16). Erittely on tehty, koska jokaiseen Unity-pelimoottoriversioon ei tarvitse ladata Android-työkaluja uudelleen. Tietokoneella voi myös olla nämä työkalut valmiina, jos siinä on käytetty Android Studio -ohjelmointiympäristöä. Työkalut voidaan myös ladata myöhemmin pelimoottoriin, kun niitä tarvitaan.



Kuva 16. Android-kehittämistä varten Unity-pelimoottorista löytyvät työkalut

Työkalujen lisääminen tapahtuu Unity Hub -ohjelmasta, joka hallinnoi Unity-pelimoottori versioita. Ohjelmasta löytyy latauksien hallinta, jossa näkyy koneella olevat pelimoottori versiot. Jokaiseen versioon voidaan täältä lisätä uusia työkaluja ja tätä kautta voidaan ladata kehittämistuet Android- ja iOS-käyttöjärjestelmille.

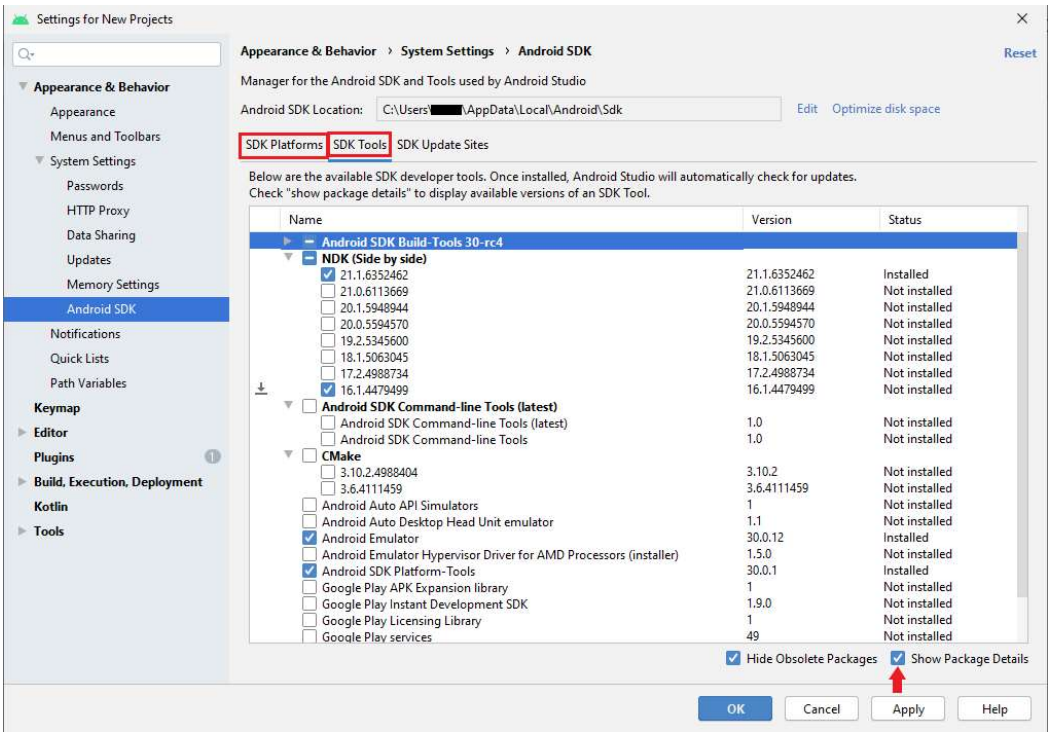
Työkalujen linkittäminen tapahtuu Unity-pelimoottorissa ylälaidassa löytyvästä Edit-kohdasta, jonka alta valitaan Preferences. Sieltä valitaan External Tools -osio, jossa linkitys tapahtuu (Kuva 17).



Kuva 17. External Tools -asetukset Unity-pelimoottorissa

Android SDK löytyy normaalisti tiedostopolusta: C:\Users\<käyttäjänimi>\AppData\Android\Sdk. NDK on tämän kansion sisällä olevan ndk-nimisen kansion alla. Unity tarvitsee tietyn version NDK-työkaluista ja se riippuu Unity-pelimoottorin versiosta. Opinnäytetyössä käytetty NDK-versio oli 16.1.4479499. Android Studio -ohjelmointiympäristöä käytettäessä tietokoneella voi olla monia NDK-versioita. Tiedostopolut ovat samat ladattaessa Unity-pelimoottorin tai Android Studio -ohjelmointiympäristön kautta, jollei käyttäjä niitä itse muuta. Linkitys tarvitsee tehdä vain kerran Unity-version sisällä, koska samaa versiota käyttäessä Unity muistaa SDK- ja NDK-työkalujen sijainnin.

Tarvittavien SDK- ja NDK-versioiden lataaminen Android Studio -ohjelmointiympäristössä käy SDK hallinnan kautta (Kuva 18). SDK-hallinnasta löytyy Android Studio -ohjelman alkuruudun alalaidasta löytyvästä konfigurointi valikosta.



Kuva 18. SDK hallinta Android Studio -ohjelmassa

SDK-hallinnassa löytyy kolme välilehteä. Opinnäytetyön kannalta olennaisia on kaksi ensimmäistä. Ensimmäinen on Android-versio kohtaisten SDK-versioiden lataamista varten ja toinen on lisätyökalujen lataamista varten. NDK-työkalut saa siis ladattua toisesta välilehdestä, mutta SDK-hallinnan alalaidasta pitää asettaa Pakettien tarkemmat tiedot -asetus päälle, jolloin voi valita ladattavan NDK-version. Samasta valikosta löytyy myös USB-ajurien lataaminen, joka on pakko tehdä Windows-tietokoneilla, jos haluaa rakentaa sovelluksen suoraan testilaitteelle.

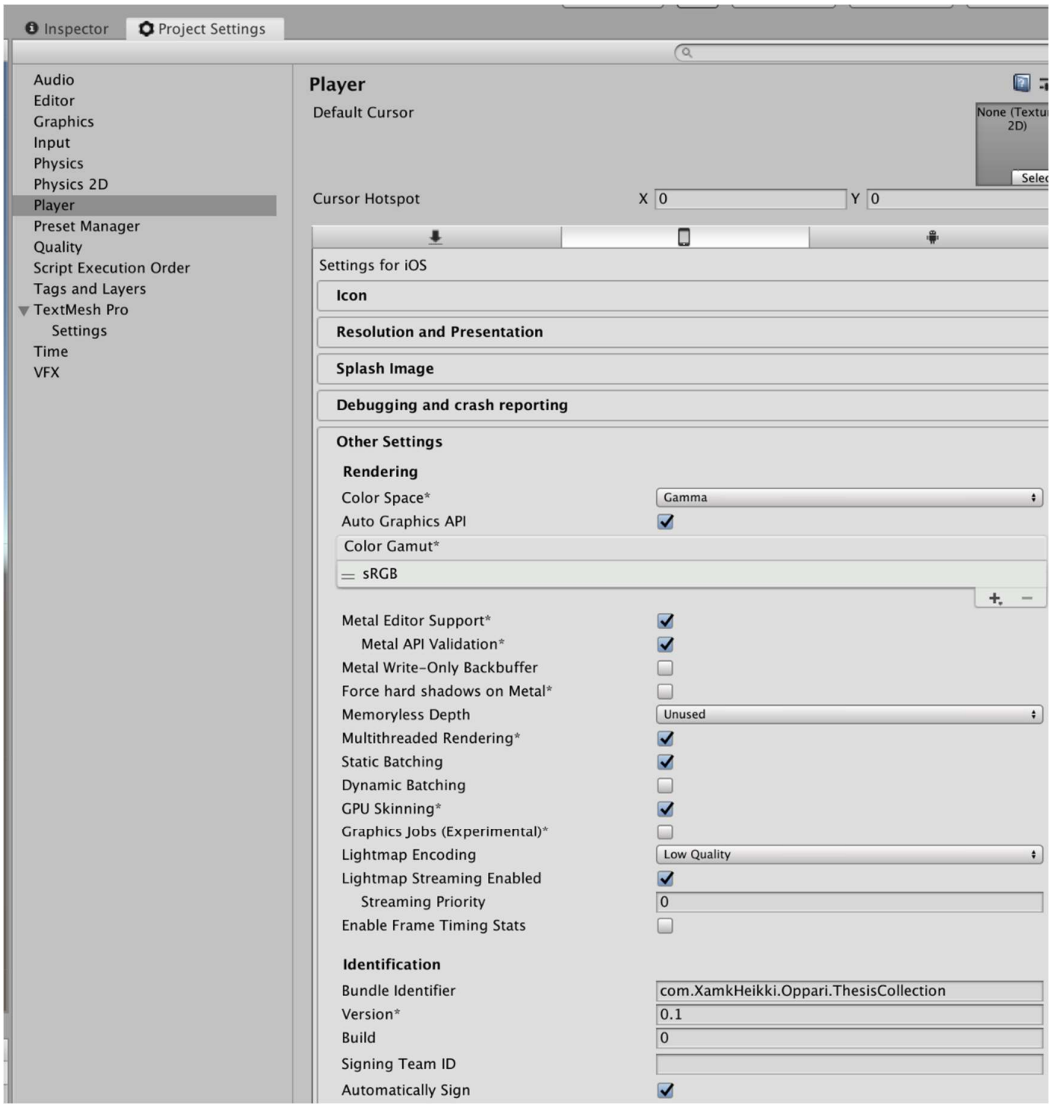
Kun Unity-pelimoottori on linkitetty tarvittaviin työkaluihin, tarvitsee testausta varten vielä säätää Android-laite kehittäjätilaan. Se tapahtuu mobiililaitteen asetuksista. Asetuksissa löytyy kohta Tietoja puhelimesta, joka sisältää tietoja laitteesta ja Android-versiosta. Siellä on kohta Ohjelmistoversion numero, jota painamalla monta kertaa, käyttäjälle aukeaa vaihtoehto tulla kehittäjäksi. Kehittäjäasetukset ilmestyvät asetusvalikkoon. Kehittäjäasetuksista käyttäjän pitää käydä laittamassa USB-vianetsintätila päälle. Kun asetus on päällä, Unity pystyy rakentamaan USB-johdon välityksellä testattavan version suoraan mobiililaitteeseen.

Testausversion jakaminen Androidilla on yksinkertaista, koska Unity rakentaa APK-tiedoston, jonka voi jakaa testaajille. Tämä keino voi kuitenkin olla turvallisuusriski, koska kuka tahansa voi suorittaa APK-tiedoston. Jos kehitysvaiheessa käyttäjällä on jo kehittäjätili Google Play Store -sovelluskauppaan, voi sen kautta myös jakaa testattavan version testaajille. Tämä ratkaisu on turvallisempi ja Googlen palvelut antavat lisätietoja sovelluksen käytöstä kehittäjälle. Testiversion jakaminen sovelluskaupan kautta käy samalla tapaa kuin sovelluksen julkaiseminen, joten se selitetään luvussa 5. Julkaisu. Testaajat rekisteröidään heidän sähköpostiosoitetta käyttäen ja näin he pääsevät käsiksi testiversioon.

4.3.2 iOS

iOS-alustalle testiversion rakentaminen ei ole yhtä nopeaa, mutta sen alustaminen Unity-pelimoottorin puolella vaatii vain Unity-version ohessa ladatut iOS-kehitystyökalut. Jos niitä ei ole aiemmin ladattu, ne löytyvät samalla tapaa Unity Hub -ohjelmasta kuin Android-työkalut.

Sovellukselle pitää tässä vaiheessa myös antaa pakettitunniste. Tätä tunnistetta käytetään sovelluksen tunnistamisessa ja se on muodossa com.OrganisaationNimi.SovelluksenNimi. Kun tunniste on rekisteröity kehittämiseen Xcode-ohjelmointiympäristöön, kukaan ei voi enää käyttää sitä tunnistetta. Testausta varten kannattaa siis luoda oma pakettitunniste lisäämällä tunnisteen perään ”testi”, koska julkaisemista varten tarvitsee uuden pakettitunnisteen. Tunniste luodaan Player-asetuksista (Kuva 19).



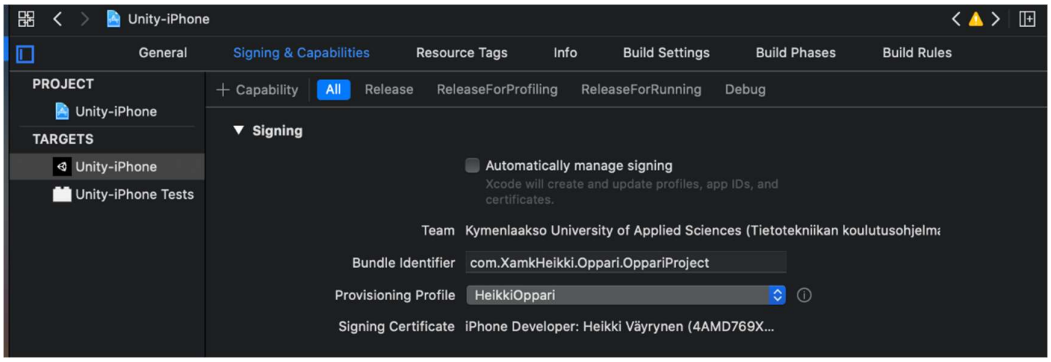
Kuva 19. Player-asetukset iOS-alustalle rakentamiseen

Player-asetuksissa pakettitunniste löytyy Other Settings -kohdan alta.

Unity-pelimoottorissa pitää myös käydä rakentamisasetuksissa vaihtamassa rakennuskohteeksi iOS-käyttöjärjestelmä. Sen jälkeen Unity lataa projektin sisällön uudestaan. Kun rakennuskohde on vaihdettu, Unity pystyy rakentamaan kansion, joka sisältää kaiken projektille tarpeellisen. Kansion sisällä on Xcode-projektitiedosto, joka on muotoa .xcodeproj.

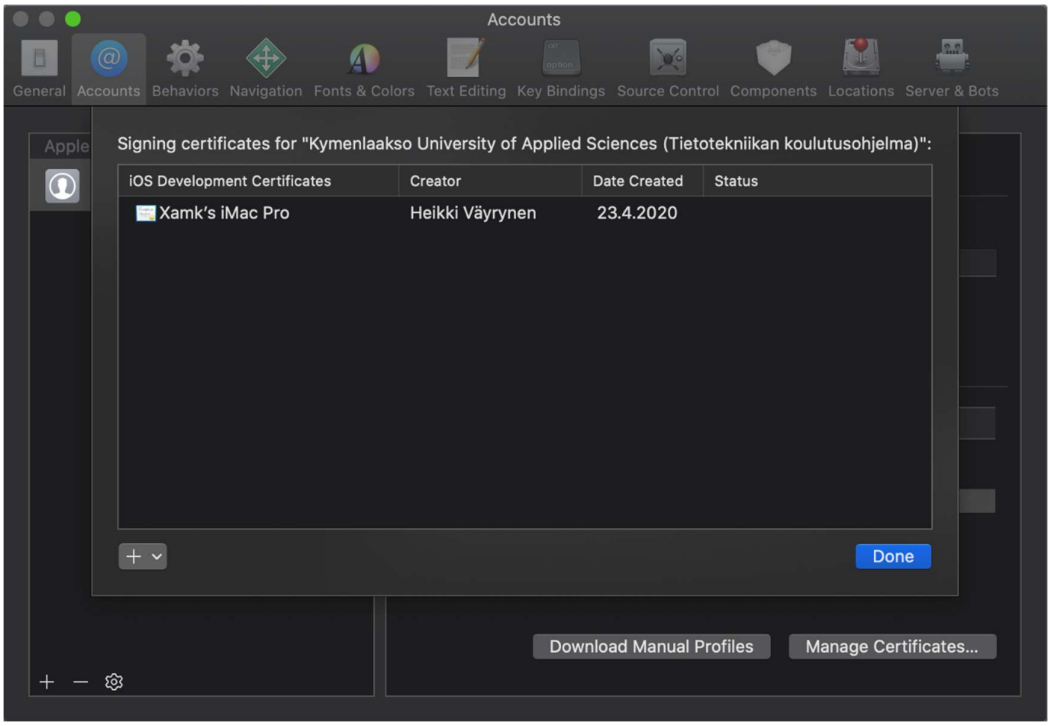
Rakentamista varten kehittäjän pitää konfiguroida Xcode valmiiksi. Ensimmäinen askel on hankkia Apple ID, jota käytetään kehittämiseen. Apple ID lisätään Xcode-ohjelmaan Preferences-asetuksista ja siellä käyttäjäasetuksista. Jos Apple ID ei kuulu valmiiseen kehitystiimiin, sille luodaan oma kehitystiimi.

Seuraavaksi avataan Xcode-projektitiedosto. Xcode tulee ilmoittamaan virheistä kohdassa Identity. Jos käytössä on ilmainen kehitystiimi, voi painaa virheen alla olevaa korjaa virheet painiketta ja Xcode luo suoraan tarvittavat sertifikaatit ja profiilit. Kehittäjä vaatii Provisioning Profile -nimisen profiilin. Jos käytössä on maksullinen kehitystiimi, pitää profiili käydä luomassa Apple kehittäjätilin kautta (Kuva 20).



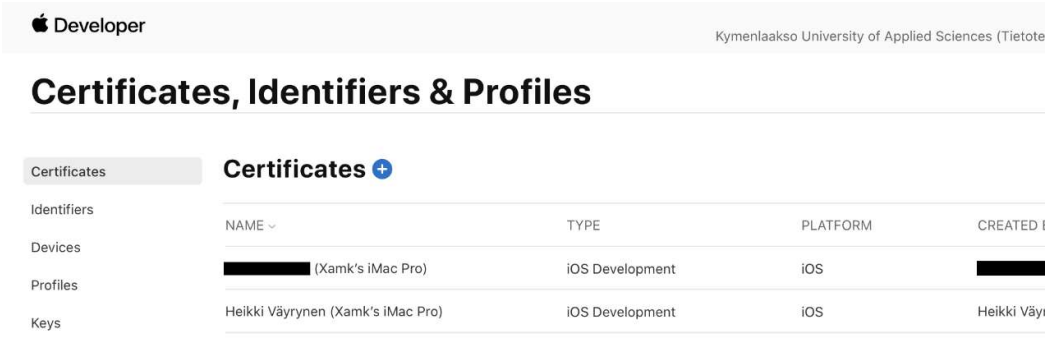
Kuva 20. Xcode Provisioning Profile

Profiilia varten aluksi tarvitsee sertifikaatin käytettävästä tietokoneesta. Sen saa Xcode-ohjelman Preferences-asetuksista samojen käyttäjäasetusten alta, mistä aiemmin lisättiin Apple ID. Siellä on painike Manage Certificates, joka avaa ikkunan, josta voi luoda uuden sertifikaatin (Kuva 21).



Kuva 21. Manage Certificates -asetukset

Sertifikaatin pitäisi nyt olla näkyvissä kehittäjätilillä. Seuraavaksi luodaan sovellukselle tunniste. Tämän luonnissa käytetään samaa pakettitunnistetta, joka luotiin Unity-pelimoottorissa. Seuraavaksi rekisteröidään iOS-testilaite kehittäjätilille. Sen rekisteröintiin tarvitsee laite kohtaisen UDID-numeron. Kun kaikki muu on tehty, voidaan luoda profiili. Näiden luominen tapahtuu Apple-kehittäjätilin hallinnassa (Kuva 22).



Kuva 22. Apple-kehittäjätilin hallinta

Profiilin voi ladata koneelle tai ladata verkosta tarvittaessa. Xcode-ohjelmassa voi nyt valita oikean profiilin, joka ilmaisella käyttäjällä luotiin suoraan. Nyt

Xcode pystyy rakentaa testiversion sovelluksesta testilaitteeseen, joka on kiinni tietokoneessa. Jos laite kieltäytyy ajamasta sovellusta, asetuksista löytyy asetus sovellukselle, jossa voi valita sovelluksen luotettavaksi.

Yleiseen testaamiseen tarvitsee luoda eri versio sovelluksesta, joka on valmis julkaisua varten. Tämä jaetaan käyttäjille App Store Connect -nimisen kehittäjätyökalun kautta. Tämä toiminta on samanlainen kuin julkaisussa ja se selvennetään kappaleessa 5. Julkaisu.

5 JULKAISU

Sovelluksen ollessa valmis se halutaan julkaista molemmille mobiilialustoille. Molemmille alustoille löytyy julkaisua varten oma sovelluskauppa. Android-käyttöjärjestelmälle se on Google Play Store ja iOS-käyttöjärjestelmälle se on App Store.

5.1 Android julkaisu

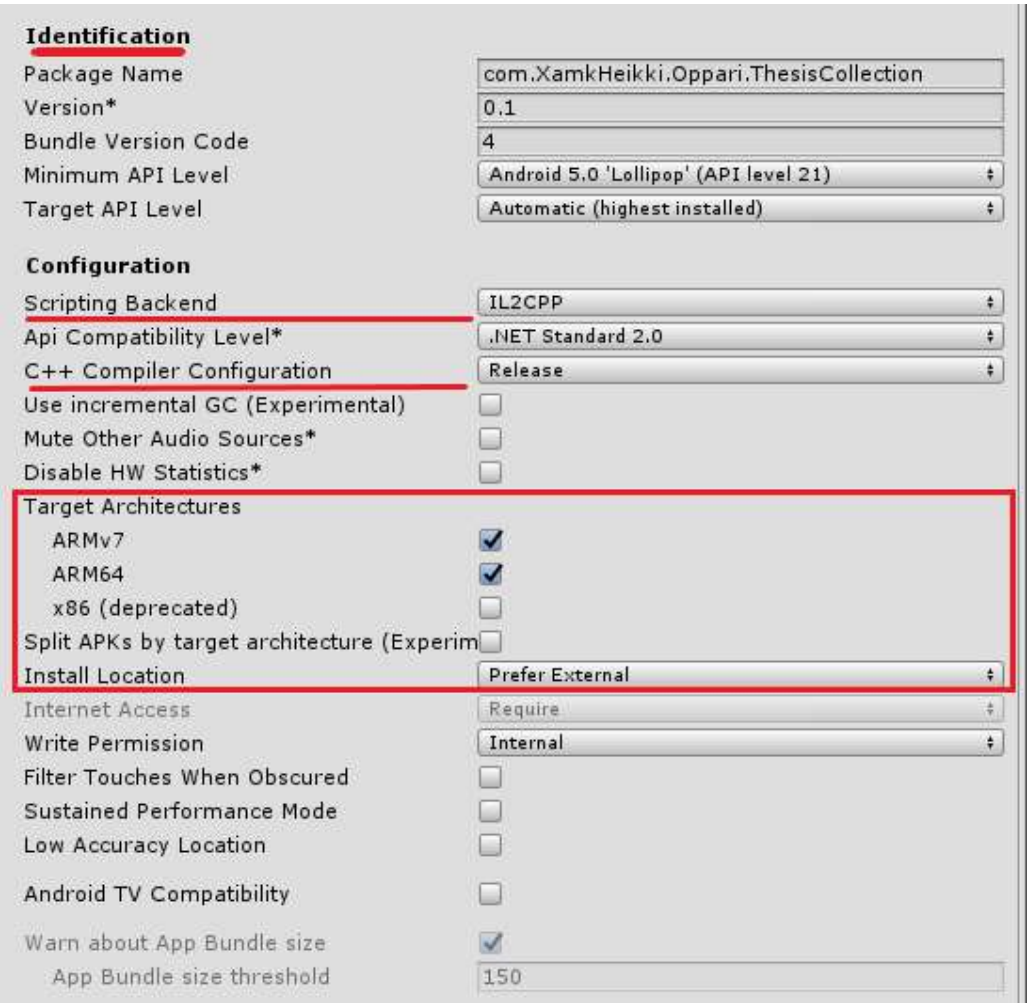
Android-alustalle pääasiallinen julkaisuväylä on Google Play Store. Play Store vaatii kehittäjätilin luomisen. Tilin luominen maksaa 25€, mutta se on kertamaksu, jonka jälkeen tiliä voi käyttää useiden sovellusten julkaisemiseen.

5.1.1 Android-version rakentaminen

Ennen julkaisua kannattaa varmistaa, että sovellus on rakennettu oikeassa muodossa Unity-pelimoottorissa. Tämä tapahtuu muokkaamalla Unity-pelimoottorissa olevia Player-asetuksia, jotka löytyvät Project-asetusten alta. Player-asetuksissa on erilaiset asetukset kaikille sovelluksen rakennusalustoille. Kaikista näistä alustakohtaisista asetuksista löytyy eri kategorioita. Android-alustalle nämä ovat ikoni, resoluutio ja ulkonäkö, sovelluksen avausanimaatio, muut asetukset, julkaisu asetukset ja XR-asetukset. Ikoni-kohdassa sovellukseen asetetaan kuvat, jota Android-laite käyttää sovelluksen ikonina Android-laitteen valikoissa. Resoluution ja ulkonäön asetuksissa kehittäjä määrittää sovelluksen oletuskuvakulman ja resoluutioon liittyviä toimintoja. Sovelluksen avausanimaatio sisältää aina Unity-pelimoottorin logon, mutta kehittäjä pääsee täältä muokkaamaan sen muuta ulkonäköä. Muissa

asetuksissa ja julkaisuasetuksissa tehdään julkaisun kannalta tärkeimmät asetukset. XR-asetukset liittyvät sovelluksiin, jotka käyttävät AR-toiminnallisuutta.

Muissa asetuksissa julkaisun kannalta tärkeinä kohtina identifikaatio ja konfiguraatio (Kuva 23). Identifikaatiossa säädetään sovellukselle pakettitunniste, joka on samalla tavalla muodostettu kuin iOS-alustalle. Android-alustalla pakettitunniste ei ole samalla sidottu jokaiseen versioon, joten testatessa voi käyttää samaa tunnistetta. Identifikaatiossa myös määritetään sovelluksen nykyiset versionumerot ja sovelluksen kohde Android-versiot.



Kuva 23. Muut asetukset Unity-pelimoottorissa Android-alustaan liittyen

Konfiguraatiossa säädetään julkaisua varten sovelluksen rakentamiseen vaikuttavia asioita. Scripting Backend -asetuksesta kannattaa valita IL2CPP-kääntäjä. Se parantaa eri alustoille rakennettaessa sovelluksen suorituskykyä,

turvallisuutta ja yhteensopivuutta alustan kanssa. Se kääntää käytetyn ohjelmointikielen C++-ohjelmointikieleksi ennen alustakohtaisen tiedoston luomista. (IL2CPP 2020.) Sitten C++-kääntäjän konfigurointi kohdasta vaihdetaan rakennettavan sovelluksen version tyypiksi julkaisu. Kohde arkkitehtuureista valitaan ARMv7 ja ARM64 eli 32- ja 64-bittinen versio. Jos julkaisualustana käyttää muuta kuin Google Play Store -sovelluskauppaa, kannattaa valita APK-tiedostojen erittely rakennusvaiheessa. Sovelluskauppaa varten on varsinaisissa rakentamisasetuksissa eri vaihtoehto tälle toiminnolle.

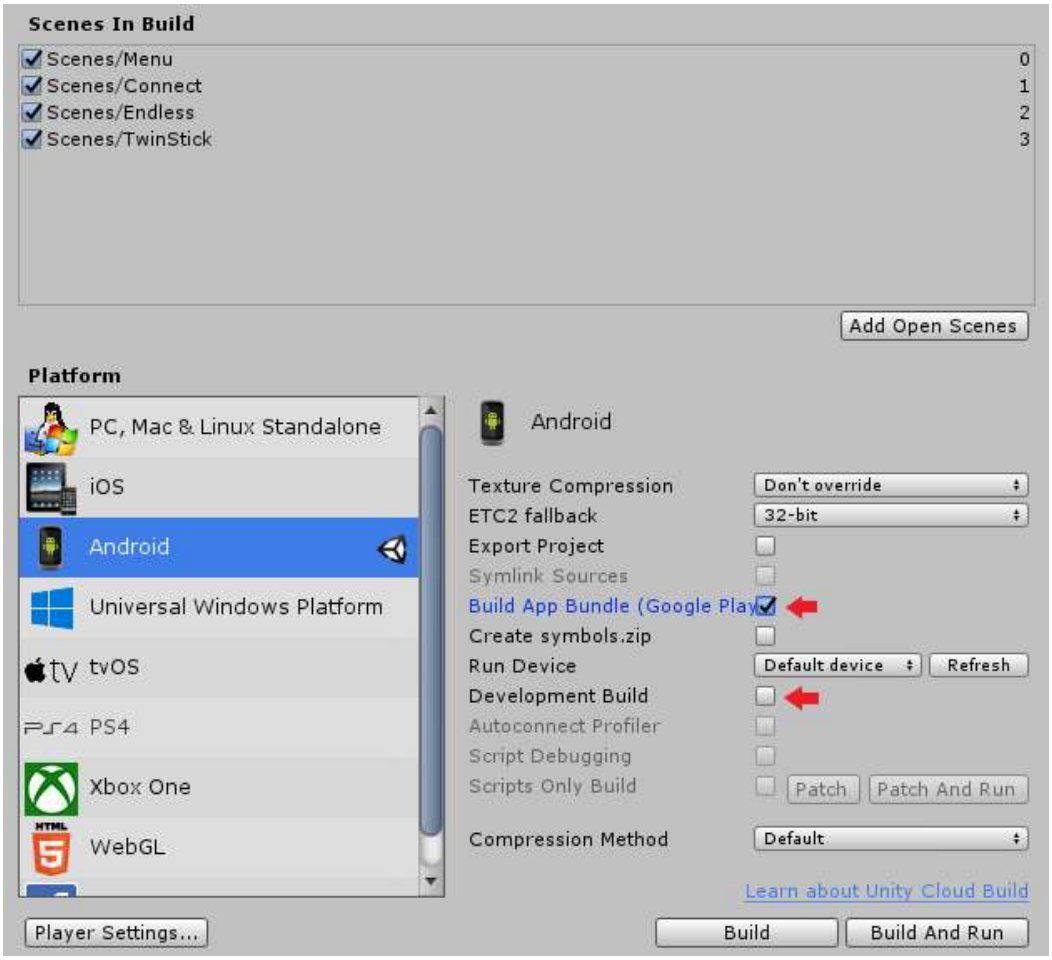
Julkaisuasetuksissa käyttäjä luo Keystore-tunnuksen sovellukselle (Kuva 24). Tätä tunnistinta käytetään tunnistamaan kehittäjä esimerkiksi sovellusta päivittäessä (Android keystore system 2019).

Kuva 24. Keystore-tunnuksen luonti

Keystore-tunnuksen luonnissa aluksi asetetaan koko Keystore-tunnuksen salasana. Keystore voi olla yrityksen tai organisaation käytössä jatkuvasti ja sen alle voidaan luoda projektitunnuksia, jotka erittelevät projektit ja niitä tarvitaan

kyseisen projektin kehittämiseen. Projektitunnukselle annetaan nimi ja salasana. Sille myös määritetään voimassaoloaika. Voimassaoloaika on tärkeä, jos sovellus sisältää ulkopuolisen omistamaa sisältöä, johon on ostettu oikeus vain tietyksi aikaa. Loput tiedot antavat täsmennystä kehityspaikasta ja kehittäjästä. Nämä tiedot eivät ole välttämättömiä, mutta ne kannattaa täyttää.

Kun kaikki asetukset ovat valmiit, voidaan siirtyä rakentamaan sovellus Unity-pelimoottorissa. Rakentamisasetuksissa valitaan projektista rakennettavat osat (Kuva 25).



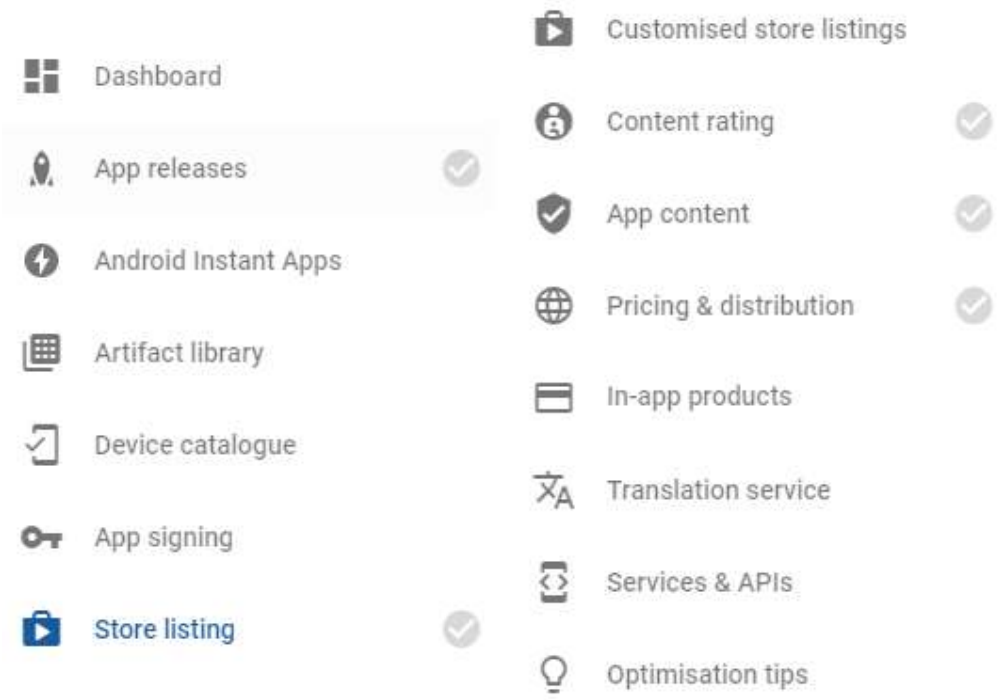
Kuva 25. Rakentamisasetukset

Rakennettaessa Google Play Store -sovelluskauppaan kannattaa valita rakennustavaksi App Bundle. Se jakaa sovelluksen eri bittiset versiot ja pienentää sovelluksen kokoa, kun se ladataan. Kannattaa myös muistaa, että asetuksissa ei ole vahingossa valittuna kehitysversio. Kaikki muu voi olla oletusarvoissaan ja sovelluksen voi nyt rakentaa ja se on sitten valmis julkaistavaksi.

5.1.2 Google Play Store

Google Play Store -kehittäjätilin luonti on yksinkertainen prosessi. Siihen tarvitsee Google-tilin, joka voi olla olemassa oleva tai kehittäjätiliä varten luotu uusi tili. Sen jälkeen hoidetaan maksu ja käyttäjä pääsee kehittäjäkonsoliin. Kaikki toiminnot kehittäjäkonsolissa on hyvin selkeästi ohjeistettu ja konsoli estää käyttäjää tekemästä virheitä.

Konsolista aukeaa ensimmäiseksi näkymä, jossa voidaan luoda uusi sovellus. Sovelluksen luonnissa asetetaan sovellukselle nimi, jonka vaihtaminen on vaikeaa, joten on tärkeää nimetä sovellus sopivalla tapaa. Sovellukselle annetaan heti myös kuvaus. Konsolissa on sivupalkki, jossa näkee ennen julkaisua määritettävät asiat (Kuva 26).



Kuva 26. Google Play -konsolin sivupalkki

Sivupalkissa pakollisien osioiden sivussa on oikeinmerkki, joka muuttuu vihreäksi, kun asetukset on hoidettu kuntoon. Pakollisia osioita ovat sovellusjulkaisut, sovelluksen tietosivu, sisällön ikärajoitus, sovelluksen sisältö ja hinnoittelu ja jakelu.

Sovellusjulkaisuissa voi julkaista sovelluksen testausta ja varsinaista kauppaa varten. Sisäiseen testaukseen ei tarvitse hoitaa muita pakollisia asetuksia, mutta muihin versio julkaisuihin ne on käytävä läpi. Mahdolliset julkaisukanavat ovat tuotanto, beta, alfa ja sisäinen testi. Tuotanto on varsinainen julkaisuversio. Beta on avoin testiversio ja alfa on suljettu testiversio.

Sovellusjulkaisuissa kaikki kanavat noudattavat samaa kaavaa. Konsoliin ladataan APK-tiedosto tai App Bundle, joista nykyisin kannattaa valita App Bundle. Kun versio on ladattu, konsoli tarkistaa tiedoston virheiden varalta, jos tiedostossa ei ole ongelmia, nimetään julkaisu ja kerrotaan, mitä uutta tässä versiossa on. Tämän jälkeen, jos kaikki muut asetukset ovat kohdallaan, voi sovelluksen tarkistaa ja käynnistää julkaisuprosessin. Prosessissa Google vielä tarkastaa tarkemmin sovelluksen tiedot ja toimivuuden. Tämä saattaa kestää monta päivää. Opinnäytetyön aikaan prosessin kestoksi ilmoitettiin seitsemän päivää tai enemmän.

Sovelluksen tietosivulla kehittäjä voi muokata kuvausta, asettaa sovelluksen kauppaikonit, säätää mainosvideon, määrittää sovelluksen tyypin ja antaa sovellukselle yhteystiedot. Sisällön ikärajoituksessa kehittäjä vastaa kyselyyn, jolla määritettään sovelluksen ikäsuositus.

Sovelluksen sisällössä käydään läpi tietosuojakäytäntö, mainokset, sovelluksen käyttöoikeudet ja kohderyhmä ja sisältö. Tietosuojakäytäntö on hyvä olla, mutta se ei ole tarpeellinen, jos sovellus ei pyydä mitään oikeuksia mobiililaitteelta. Tietosuojakäytännön on oltava saatavilla internetissä. Mainoksissa määritetään, sisältääkö sovellus mainoksia. Sovelluksen käyttöoikeuksissa kerrotaan sovelluksen tarvitsemista oikeuksista mobiililaitteen toimintoihin. Kohderyhmässä ja sisällössä määritetään, kenelle sovellus on suunnattu.

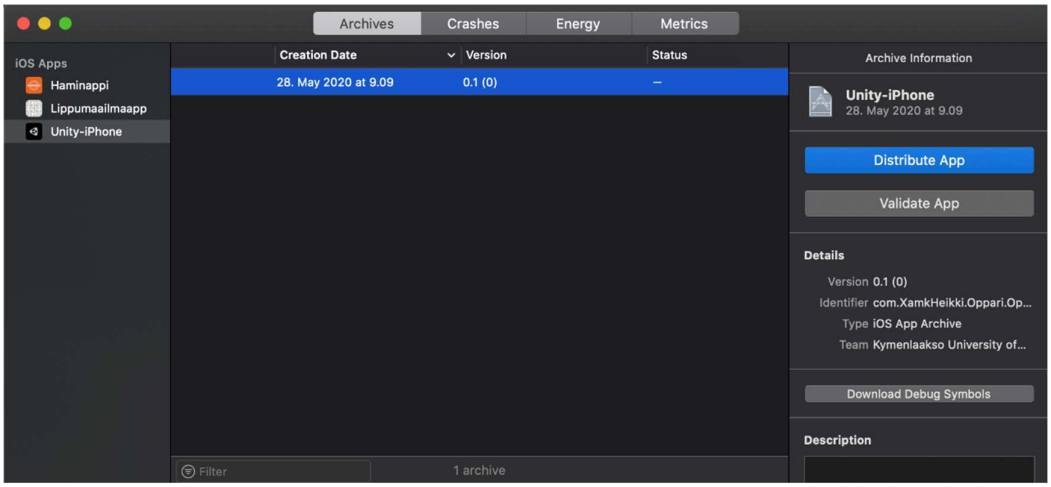
Hinnoittelussa valitaan sovelluksen hinta. Jos haluaa luoda maksullisen sovelluksen, pitää kehittäjän luoda kauppiastili. Jakelu-osiossa valitaan julkaisuun sisältyvät valtiot ja laitteet. Osiosta löytyy myös lupa-asioita, joista tärkeimmät ovat Android-sisältösäännöt ja sovelluksen kuuluminen Yhdysvaltain vientilainalaisuuteen.

5.2 App Store

Julkaisua varten iOS-käyttöjärjestelmällä käytetään App Store -sovelluskauppaa. Julkaisua varten kehittäjä tarvitsee maksullisen kehittäjätilin, joka maksaa noin 100 € vuodessa. Jakelua varten pitää luoda jakeluprofiili, joka tapahtuu samalla tapaa kehittäjätilin kautta kuin aiemman kehitystä varten tarvittu profiili. Kehittäjän pitää ensin hakea Xcode-ohjelman kautta jakelusertifikaatti.

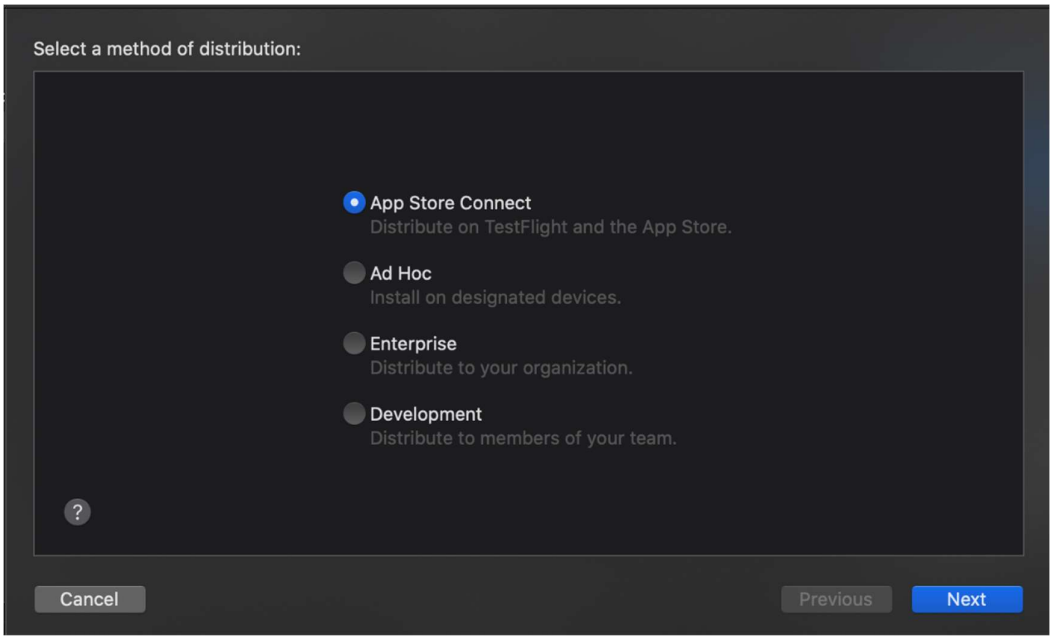
Unity-pelimoottorin puolella kannattaa käydä läpi Player-asetukset samaan tapaan kuin Android-alustalle julkaisua varten. iOS-alustalle asetuksia on vähemmän ja oletusasetukset käyvät pääasiallisesti projektiin. Ikonit ja avausanimaatio pitää muokata erikseen molemmille alustoille.

Sitten Xcode-ohjelmassa luodaan Archive-versio sovelluksesta. Tämä vastaa Android-alustalle luotua App Bundle -versiota. Kun versio on luotu, aukeaa näkymä (Kuva 27), jossa päästään käsiksi sovellukseen ja voidaan käynnistää sen jakelu.



Kuva 27. Xcode Archive-näkymä

Jakeluun on erilaisia vaihtoehtoja: App Store Connect, Ad Hoc, yhtiö ja kehitys. App Store Connect on avointa jakelua ja julkaisua varten. Muut vaihtoehdot ovat eri asteisia keinoja jakaa sovellus testausta varten aina kehitystiimistä kaikkiin haluttuihin laitteisiin (Kuva 28).

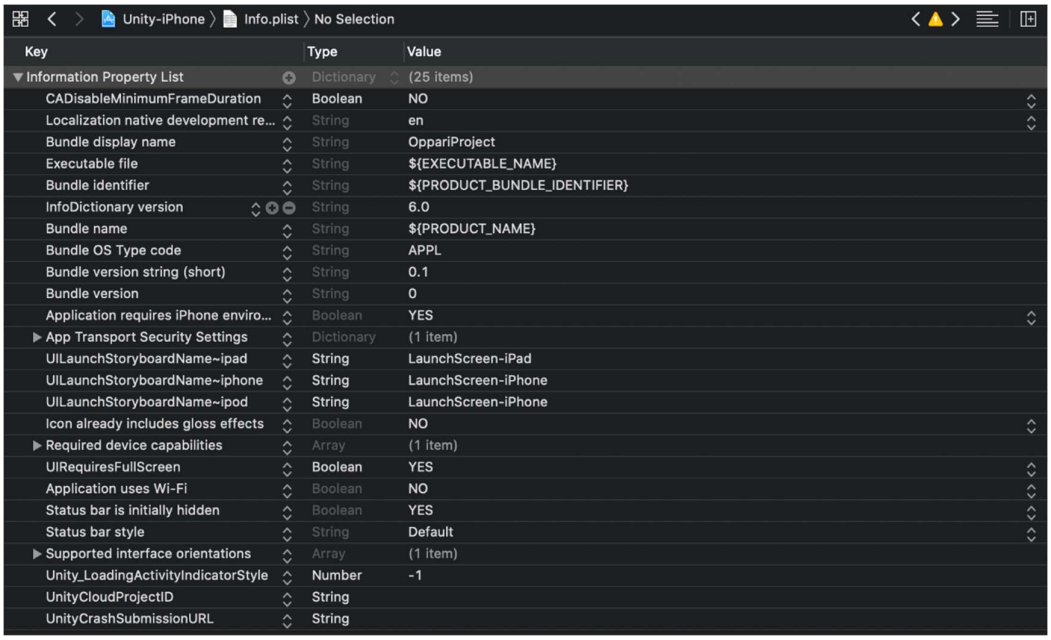


Kuva 28. Jakelun vaihtoehdot iOS-alustalle.

Tässä vaiheessa tarvitaan jakeluprofiili. Opinnäytetyössä käytettyyn kehittäjätiliin ei saatu vaadittavia oikeuksia jakeluprofiilin luomiseen, joten iOS-käyttöjärjestelmään julkaisu käydään läpi teoreettisesti ja lainaten aiemmin julkaisuvaiheeseen vietyä projektia.

App Store Connect -sivusto on iOS-alustalle julkaisuun käytettävä nettisivu. Kun jakelu on Xcode-ohjelman kautta aloitettu, pääsee julkaisua jatkamaan täältä. Sivulla on My Apps -osio, jonka alta pääsee käsiksi sovelluksen julkaisuun ja jakeluun testaajille. Sivulla on hyvin selkeät ohjeet julkaisua varten ja siellä täytetään samanlaiset tiedot kuin Google Play Store -sovelluskaupan puolella.

Apple on tarkka salaustekniikoiden käytöstä sovelluksissa. Näihin luetaan esimerkiksi yhteys tietokantaan. Xcode-projektitiedostossa on valmis Info.plist-tiedosto, johon merkitään salaustekniikoiden käyttö (Kuva 29).



Kuva 29. Info.plist-tiedosto Xcode-ohjelmassa

Apple ei vaadi itselleen aina selvitystä tämän kaltaisista alustan sisäänrakennetuista suojauksista, mutta niistä saatetaan Yhdysvaltain lainalaisuudessa vaatia selvitys. Jos sovellus käyttää muunlaisia salaustekniikoita, niistä lähetetään selvitys ja Apple palauttaa kehittäjälle koodin, joka syötetään sovelluksen tietoihin. (Complying with Encryption Export Regulations 2020.)

6 YHTEENVETO

Kehitettäessä molemmille Android- ja iOS-käyttöjärjestelmille kannattaa käyttää ulkopuolista ohjelmaa. Molempien alustojen omat ohjelmointiympäristöt ovat hyviä oman alustansa kohdalla, mutta eivät tue toiselle kehittämistä. Jos kyseessä on peli, kannattaa valita pelimoottori kuten Unity. Unity tarjoaa tuen molempien alustojen kehitykseen ja pelimoottorina hoitaa hyvin vaatimukset pelikehitykseen.

iOS-käyttöjärjestelmä on vaativampi ja kalliimpi kehityksen kannalta. Xcode-ohjelmointiympäristö on saatavilla ainoastaan Apple-tuotteisiin ja kehittäjätilillä on vuosimaksu, joka on neljä kertaa isompi kuin Android-alustan vastaava, joka on kertamaksu. Molemmille alustoille kehitettäessä kannattaa kuitenkin hankkia Mac-tietokone, koska sillä pystyy kehittämään molemmille alustoille.

Android on testaukseen Unity-pelimoottorin kautta parempi, koska Unity hoitaa rakentamisen suoraan, kun iOS-alustalle joudutaan käyttämään Xcode-ohjelmaa välissä. Kuitenkin testausta kannattaa aina tehdä molemmilla alustoilla kehitysvaiheen aikana. Unity kuitenkin kääntää koodin hyvin molemmille alustoille, jolloin sovelluksen toiminta on samanlaista molemmissa.

Testausta varten kannattaa käyttää mahdollisimman montaa erilaista laitetta. Tärkeimpänä on erilaisten näyttökokojen saaminen testilaitteisiin. Opinnäytetyössä käytössä oli älypuhelin ja tabletti ja näin saatiin nähtyä hyvin erilaisien näyttöjen toiminta pelien ja käyttöliittymän kannalta. Kuitenkin oli selkeää, että joidenkin näyttöjen resoluutioiden kohdalla ei ole selvää, miltä sovellus näyttää.

Julkaisussa molemmissa alustoissa on ongelmansa. Android-alustalle sovelluksen rakentaminen julkaisua varten on paljon hankalampaa. Kun sovellus on valmis julkaistavaksi, sen saaminen sovelluskauppaan on yksinkertaista. iOS-alustalla sovelluksen julkaisuversion rakentaminen on helppoa, koska Xcode hoitaa rakentamisen. Sovelluksen julkaisemiseen on vielä matkaa, koska kehittäjän pitää hankkia erillinen julkaisuprofiili ja Apple vaatii tiukat selvitykset erilaisista salaustekniikoista, joita sovelluksessa on käytetty. Molempien alustojen julkaisuun käytettävät nettisivut ovat erittäin hyvin ohjeistettuja.

Projektin kehityksessä ei ollut suuria ongelmia pelien yksinkertaisuuden vuoksi. Peleistä monimutkaisin oli Peli 2, mutta se oli aiemmin luotu toiseen projektiin, joten sen koodi oli valmis jo ennen projektin aloitusta. Ohjelmoinnin puolella uusina asioina opittiin yksittäisten kosketusten seuraamisesta, vaikka toimintoa ei sellaisenaan tuotu projektiin.

Jatkokehityksen kannalta pelit kannattaa jakaa omiin sovelluksiin, milloin yksittäisten pelien kehitykseen voi panostaa paremmin. Peleissä on myös paljon parannettavaa pelattavuuden ja toimintojen puolella. Peleissä 1 ja 3 sisällössä ei ole vaihtuvuutta ja Peli 2 on heikompi muihin samanlaisiin peleihin verrattuna. Jokaisen pelin kohdalla jatkokehitys on suotavaa. Sovelluksen kuvakulman vaihtaminen kesken suorittamisen on huono toiminto ja se pitäisi saada pois sovelluksesta.

LÄHTEET

Android Developers. 2019. Design for Android. WWW-dokumetti. Päivitetty 27.12.2019. Saatavissa: https://developer.android.com/design [viitattu 13.5.2020].

Android environment setup. 2020. Unity Documentation. WWW-dokumentti. Saatavissa: https://docs.unity3d.com/Manual/android-sdksetup.html [viitattu 10.2.2020].

Android keystore system. 2019. Developers.android.com. WWW-dokumentti. Päivitetty: 27.12.2019. Saatavissa: https://developer.android.com/training/articles/keystore.html [viitattu 27.5.2020].

Apple Developer s.a. Human Interface Guidelines. WWW-dokumetti. Saatavissa: https://developer.apple.com/design/human-interface-guidelines/ios/overview/themes/ [viitattu 13.5.2020].

Arvidsson, N. 2018. A Truly Excellent Beginners Guide To Xcode. WWW-dokumentti. Päivitetty: 9.2.2018. Saatavissa: https://careerfoundry.com/en/blog/ios-development/what-is-xcode/ [viitattu 21.5.2020].

Bohn, D., Souppouris, A. & Seifert, D. 2013. iOS: A visual history. WWW-dokumentti. 16.9.2013. Saatavissa: https://www.theverge.com/2011/12/13/2612736/ios-history-iphone-ipad [viitattu 6.5.2020].

Blundell, P. & Milano, D. 2015. Learning Android Application Testing. E-kirja. Birmingham: Packt Publishing. Saatavissa: https://kaakkuri.finna.fi/ [viitattu: 24.5.2020].

Camera. 2020. Unity Documentation. WWW-dokumentti. Saatavissa: https://docs.unity3d.com/Manual/class-Camera.html [viitattu 22.5.2020].

Camera.ortographicSize. 2020. Unity Documentation. WWW-dokumentti. Saatavissa: https://docs.unity3d.com/ScriptReference/Camera-orthographicSize.html [viitattu 22.5.2020].

Callaham, J. 2019. The history of Android OS: its name, origin and more. Päivitetty: 18.8.2019. Saatavissa: https://www.androidauthority.com/history-android-os-name-789433/ [viitattu 6.5.2020].

Coding in C# in Unity for beginners s.a. Unity3d.com. WWW-dokumentti. Saatavissa: https://unity3d.com/learning-c-sharp-in-unity-for-beginners [viitattu 6.2.2020].

Complying with Encryption Export Regulations. 2020. Developer.apple.com. WWW-dokumentti. Saatavissa: https://developer.apple.com/documentation/security/complying_with_encryption_export_regulations [viitattu 28.5.2020].

Create and Monetize with Unity Mobile Games Development s.a. Unity3d.com. WWW-dokumentti. Saatavissa: https://unity.com/solutions/mobile [viitattu 6.2.2020].

Friesen, J. 2020. Android Studio for beginners, Part 1: Installation and setup. WWW-dokumentti. Päivitetty: 17.1.2020. Saatavissa: https://www.javaworld.com/article/3095406/android-studio-for-beginners-part-1-installation-and-setup.html [viitattu 10.2.2020].

Gamelab.fi. 2019. WWW-dokumentti. Saatavissa: https://www.gamelab.fi/#about [viitattu 20.5.2020].

Hietanen, P. 2003. C++ ja olio-ohjelmointi. E-kirja. Jyväskylä: Docendo. Saatavissa: https://kaakkuri.finna.fi/ [viitattu 21.5.2020].

Hoffman, C. 2017. Android Is “Open” and iOS Is “Closed” — But What Does That Mean to You? WWW-dokumentti. Päivitetty 20.6.2017. Saatavissa: https://www.howtogeek.com/217593/android-is-open-and-ios-is-closed-but-what-does-that-mean-to-you/ [viitattu 13.5.2020].

IL2CPP. 2020. Unity Documentation. WWW-dokumentti. Saatavissa: https://docs.unity3d.com/Manual/IL2CPP.html [viitattu 26.5.2020].

Kotaki, G. 2012. Introduction to Pixel Art for Games. WWW-dokumentti. Saatavissa: https://www.raywenderlich.com/2888-introduction-to-pixel-art-for-games [viitattu 10.2.2020].

Leonard, L. 2019. Preparing your Android Unity game for Google’s 64-bit requirements. WWW-dokumentti. Saatavissa: https://medium.com/@leonard.lin.2003/preparing-your-android-unity-game-for-googles-64-bit-requirements-bb6ece4e57e6 [viitattu 26.5.2020].

Mobile Operating System Market Share Worldwide. 2020. StatCounter. WWW-dokumentti. Saatavissa: https://gs.statcounter.com/os-market-share/mobile/worldwide [viitattu 30.04.2020].

MonoBehaviour.OnMouseDown(). 2020. Unity Documentation. WWW-dokumentti. Saatavissa: https://docs.unity3d.com/ScriptReference/MonoBehaviour.OnMouseDown.html [viitattu 23.5.2020].

Popular Screen Resolutions: Designing for All. 2020. WWW-dokumentti. Saatavissa: https://mediag.com/blog/popular-screen-resolutions-designing-for-all/ [viitattu 20.5.2020]

Sonmez, J. 2016. What is Mobile Development? WWW-dokumentti. 5.12.2016. Saatavissa: https://simpleprogrammer.com/what-is-mobile-development/ [viitattu 13.5.2020].

KUVALUETTELO