Roni Leinonen

Mobiilisovelluksen kehittäminen Android-puhelimelle

Opinnäytetyö

Tietotekniikka / Tietotekniikan koulutusohjelma

Toukokuu 2016

KYAMK
University of Applied Sciences

KYAMK
University of Applied Sciences

Tekijä/Tekijät	Tutkinto	Aika
Roni Leinonen	Insinööri	Toukokuu 2016

Opinnäytetyön nimi

Mobiilisovelluksen kehittäminen Android-puhelimelle

45 sivua

Toimeksiantaja

Marko Oras

Ohjaaja

Niina Salmi

Tiivistelmä

Opinnäytetyön tarkoituksena on antaa kuva Android-mobiililaitteelle kehitettävästä sovelluksesta ja sen työvaiheista suunnittelusta julkaisuun. Työssä esitellään sovelluksen graafisia- ja ohjelmointiratkaisuja sekä lyhyesti niiden luomisessa käytetyt työkalut. Tavoitteena oli saada aikaan julkaisukelpoinen sovellus.

Tuloksena syntyi yksinpelattava kalastuspeli ajanviettoon. Peliä pelataan pelaajan kosketuksen ja puhelimen liikkeen avulla. Simulaatiomainen peli on suunniteltu pelattavaksi ainoastaan puhelimella sen käytettävyyden takia. Laitetta pidellään yhdessä kädessä ja tabletilla se on hankalampaa.

Työ onnistui ja antoi kokemusta sovelluskehityksen eri osa-alueista. Työ auttoi kehittymään ohjelmoinnissa, grafiikan luomisessa ja äänien tekemisessä. Tavoitteeseen päästiin aikataulussa ja syntyi hyvä pohja sovelluksen jatkokehitykselle.

Asiasanat

3D-mallinnus, Android, mobiilisovellus, ohjelmointi, Unity 3D

KYAMK
University of Applied Sciences

Author (authors)	Degree	Time
Roni Leinonen	Bachelor of Engineering	May 2016

Thesis Title

Mobile Application Development for Android Phone

45 pages

Commissioned by

Marko Oras

Supervisor

Niina Salmi

Abstract

The purpose of the thesis was to create an overview of an application development for Android mobile device from design to publishing. This thesis presents graphic and programming solutions of the application and briefly introduces the programs used to make them. The goal was to make an application that can be published.

The result was a single-player fishing game for entertainment. The game is played by touching and tilting the phone. The game is designed to be played only with a phone because of its playing mechanism. The device is held in one hand and with tablet it is difficult.

The thesis was successful and gave more experience in various areas of application development. The work helped to develop skills in programming, creating graphics and making sounds. The objective was achieved as planned and created good basis for further development of the application.

Keywords

3D modeling, Android, mobile application, programming, Unity 3D

SISÄLLYS

TERMIT JA LYHENTEET

3D-malli	Tietokoneella luotu matemaattinen esitys kolmiulotteisesta mallista.
Ohjelmointikieli	Ohjelmointikieli on tietokoneen ohjelmointiin suunniteltu kieli. Kieliä on erilaisia ja kullakin on omat sanastot ja säännöt.
Pelimoottori	Ohjelmistoympäristö, joka on suunniteltu pelien tekemiseen ja kehittämiseen.
Placeholder-grafiikka	Nopeasti tuotettu grafiikka, joka antaa ohjelmoinnin toteuttamiseen vaadittavat graafiset apuvälineet, jotta pelin toimintaa voidaan testata ja kehittää jo ennen viimeisteltyä grafiikkaa.
Polygoni	Monikulmio, 3D-mallissa olevat polygonit ovat kolmi- tai nelikulmaisia.
Prefab	Unityssa luotu valmis peliobjekti, joka säilyttää jo valmiiksi määritetyt komponentit ja asetukset.
Rendering	Renderöinnillä tarkoitetaan tietokoneohjelman avulla luotavaa kuvaa mallista. 3D-mallinnuksessa renderöinti on prosessi, joka antaa malleille viimeisen ulkoasun yhdistämällä tekstuurin, geometrian, katselukulman, valaistuksen ja varjostukset.
Skripti	Kooditiedosto eli tekstitiedosto, jossa on yksi tai useampia rivejä komentoja, joilla voidaan määrittää ominaisuuksia ja toimintoja esimerkiksi peliobjektille.
UV map	3D-mallinnuksessa luotu tekstuuri, joka näyttää 3D-mallin pinnan avattuna kaksiulotteisena tasaisena pintana.

1 JOHDANTO

Opinnäytetyön tarkoituksena on kehittää mobiilisovellus, joka toimii yleisimmissä Android-käyttöjärjestelmän omaavissa puhelimissa. Tavoitteena on saada aikaan julkaisukelpoinen sovellus, joka on ladattavissa täysin ilmaiseksi.

Työn päätavoitteena ei ole saada sovellusta väkisin valmiiksi vaan oppia enemmän pelien tekemisen eri osa-alueista. Työssä pyritään oppimaan niin ohjelmointia kuin kehityksen graafistakin puolta. Kaikki lähtee liikkeelle sovelluksen suunnittelusta, jota seuraa toteutus 3d-mallinnuksesta ohjelmointiin ja äänien luomiseen.

Tässä opinnäytetyössä esitellään alkuun lyhyesti sovelluksen kehityksessä käytetyt työkalut ja miksi niihin päädyttiin. Tämän jälkeen siirrytään toteutukseen, jossa käydään läpi eri työvaiheita. Aloitetaan ideasta ja suunnittelusta, josta siirrytään 3d-mallinnukseen ja myöhemmin syvennytään erityisesti tärkeimpiin ohjelmointiratkaisuihin.

Työn lopussa ohjeistetaan yleisesti Google Play-kaupan julkaisuun vaadittavat asiat. Suunnitellaan sovelluksen jatkokehitystä sekä käydään läpi yhteenveto koko projektista.

2 TYÖKALUT

Sovelluksen kehittämisessä käytettiin pelimoottoria, 3d-mallinnusohjelmaa, kuvankäsittelyohjelmaa ja äänenkäsittelyohjelmaa. Työkalujen valintaan vaikutti aikaisempi kokemus.

2.1 Unity 3D ja MonoDevelop

Unity 3D on Unity Technologiesin kehittämä monialustainen pelimoottori. Unity 3D tukee useita eri julkaisualustoja ja niiden määrä on kasvussa. Unity 3D:stä on saatavilla ilmainen sekä maksullinen versio. Maksullinen versio on ominaisuuksiltaan hieman kattavampi, mutta ilmaisversiollakin pystyy kehittämään laadukkaita pelejä. (Unity Technologies 2016a; Unity Technologies 2016b.)

MonoDevelop on ilmainen avoimen lähdekoodin sovelluskehitysympäristö, jolla voidaan luoda Unity 3D -projektissa käytettävät skriptit. MonoDevelop asentuu oletuksena Unityn mukana. (Unity Technologies 2016c.)

Unity 3D:n valinta pelimoottoriksi syntyi nopeasti sen käyttäjäystävällisyyden ja aikaisemman kokemuksen takia, ja koska se on saatavilla ilmaiseksi. Valintaa helpotti myös mukana tuleva ohjelmointiympäristö MonoDevelop sekä Unityn kattava online-dokumentaatio Unity Manual (Unity Technologies 2016d).

2.2 3ds Max

3ds Max on Autodeskin kehittämä 3d-mallinnus-, animointi ja renderointiohjelmisto. Ohjelmisto on suunnattu ammattilaiskäyttöön ja siitä on saatavilla 30 päivän ilmaiskokeilu. Lisäksi Autodesk tarjoaa opiskelijalisenssin opiskelukäyttöön 3 vuodeksi. Opiskelijalisenssi on tarkoitettu ohjelman opetteluun. (Autodesk 2016.)

Opiskelijalisenssin ansiosta syntyi mahdollisuus tutustua ammattilaiskäyttöön suunnattuun ohjelmaan. Lisäksi 3ds Maxia pääsi käyttämään koululla koulun lisenssillä.

2.3 Audacity

Audacity on äänenkäsittelyohjelma, jolla voi tallentaa ja muokata ääntä. Ohjelma on täysin ilmainen ja toimii useilla alustoilla. (Audacity 2016.)

Audacity on ladattavissa ilmaiseksi ja se on riittävän monipuolinen tämän sovelluksen ääniefektien tekemiseen.

3 TOTEUTUS

Sovelluksen toteutus lähti liikkeelle ideoinnista ja suunnittelusta. Suunnittelun jälkeen 3d-mallinnettiin virveli, jonka jälkeen siirryttiin ohjelmointiin. Peliä ohjelmoitiin ensin käyttäen placeholder-grafiikkaa. Pelin ollessa pelattavassa muodossa päivitettiin grafiikka näyttävämmäksi ja lopuksi peliin lisättiin äänet.

3.1 Idea

Tarkoituksena on tehdä simulaatiomainen kalastuspeli puhelimelle. Idea syntyi oman harrastuksen parissa. Syntyi halu simuloida kalastuksen tuottama

rentoutunut ja rauhallinen olo. Pelin päätavoitteena ei ole saada mahdollisimman paljon kalaa, vaan enemmänkin luoda pelaajalle todentuntuinen illuusio, että hän on kalastamassa.

Pelaajalla on pelissä virveli, jota operoidaan puhelimen kosketuksen ja liikkeen avulla. Perusnäkymä näyttää vain osan virvelin vavasta ja siimasta sekä kelan kokonaan. Virveli toimii samalla tavalla kuin oikea virveli. Pelaaja voi heittää uistimen veteen, kelata sekä avata ja sulkea lukon. Pelaajalle tulee selkeä mielikuva, että hänellä on toisessa kädessä virveli ja toisella tartutaan kahvaan.

Kalan saadessaan voi kalan päästää irti tai ottaa talteen. Vain talteen otetut kalat ja niiden tiedot taulukoidaan pelaajalle nähtäväksi. Lisäksi pelissä voi vaihtaa uistinta, joita avautuu lisää kalamäärän kasvaessa.

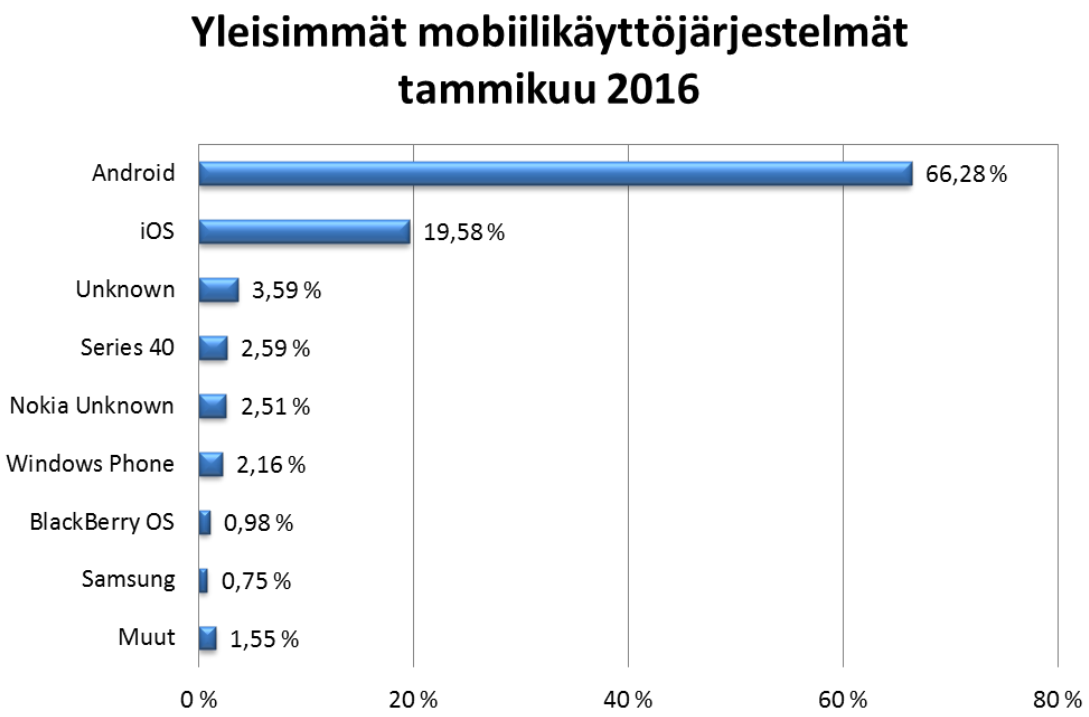
3.2 Suunnittelu

Suunnittelussa hyödynnettiin oikeata virveliä. Tutustuttiin virvelin eri toimintoihin, ääniin ja katsottiin, miten eri osat virvelissä liikkuivat. Tarkastellessa jouduttiin karsimaan muutamia ominaisuuksia pois, koska todettiin niiden vievän liikaa aikaa tai olevan vaikeasti toteutettavia.

Sovelluksen idean ollessa selkeä tehtiin karkea hahmotelma varsinaisesta pelinäkymästä ja testattiin mahdollisia ohjelmointiratkaisuja ja tutustuttiin puhelimen ominaisuuksiin, joita voitaisiin käyttää hyödyksi sovelluksessa.

3.2.1 Julkaisualusta

Mobiililaitteilla on valmistaja, mutta ne sisältävät myös jonkin käyttöjärjestelmän, joka toimii laitteen sydämenä. Mobiilikäyttöjärjestelmä on kuin tietokoneen käyttöjärjestelmä yhdistettynä ominaisuuksilla, kuten esimerkiksi kosketusnäyttö, bluetooth, GPS-paikannin, langaton verkkoyhteys, kamera ja musiikkitoistin. Kilpailu käyttöjärjestelmissä on kovaa, mutta kaksi ehdottomasti suurinta mobiilikäyttöjärjestelmää ovat Googlen omistama Android ja Applen kehittämä iOS, joista Android on selkeästi suurin. (Ashvin 2016.)



Kuva 1. Kahdeksan yleisintä mobiilikäyttöjärjestelmää tammikuussa 2016. (StatCounter 2016)

Googlen omistama Android mobiilikäyttöjärjestelmä on tällä hetkellä yleisin maailmanlaajuisesti. Toiseksi yleisimpänä on Applen kehittämä iOS käyttöjärjestelmä. Muut käyttöjärjestelmät ovat vain pieni osa verrattuna näihin kahteen suurimpaan. (Kuva 1)

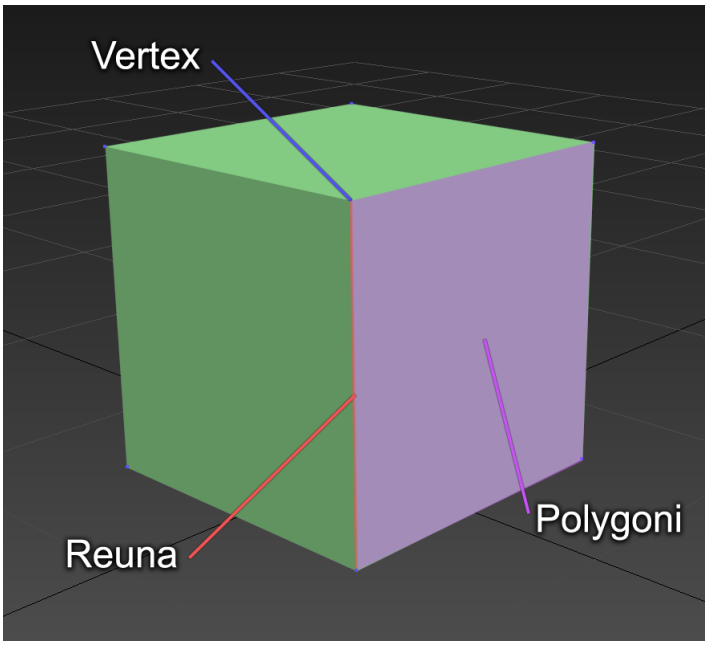
Mobiilisovelluksen kehittämisen alussa on syytä päättää, mille julkaisualustalle se tehdään. Jo suunnittelun alkuvaiheessa oli selvää, että alustan valinta kohdistuu Androidiin, koska se on maailmanlaajuisesti yleisin ja sovelluksen testauksessa käytettävä henkilökohtainen puhelin on varustettu tällä käyttöjärjestelmällä.

3.2.2 Android

Android on Googlen omistama mobiilikäyttöjärjestelmä, joka on yleisimmin asennettuna eri valmistajien tarjoamiin älypuhelimiin ja tabletteihin. Android puhelimet ovat suuresti muokattavia ja käyttäjä pystyy muokkaamaan sen omiin tarpeisiinsa sopivaksi. Sovellukset ladataan laitteille Google Play kaupasta, mutta laitteeseen voi asentaa myös tuntemattomista lähteistä. Android puhelimia ja tabletteja tarjoavia valmistajia ovat Motorola, HTC, Samsung, Sony, Acer, Alcatel, Asus, Huawei, LG, Blackberry ja ZTE. (Barraclough & Todd 2016.)

3.3 3d-mallintaminen

Pelin tärkein elementti on pelaajan käsittelemä virveli. Sen saamiseksi aidon näköiseksi se päätettiin 3d-mallintaa. Yleisin 3d-mallintamistapa on polygonimallintaminen, jossa malli koostuu useista nelikulmioista tai kolmioista. Se muodostuu pisteistä, reunoista ja polygoneista. Vertexit eli pisteet ovat kohta, jossa vähintään kolme reunaa kohtaavat. Reunat ovat kolmen tai useamman polygonin kohtaamispaikka. (Slick 2014.)



Kuva 2. Yksinkertainen laatikon muotoinen polygonimalli.

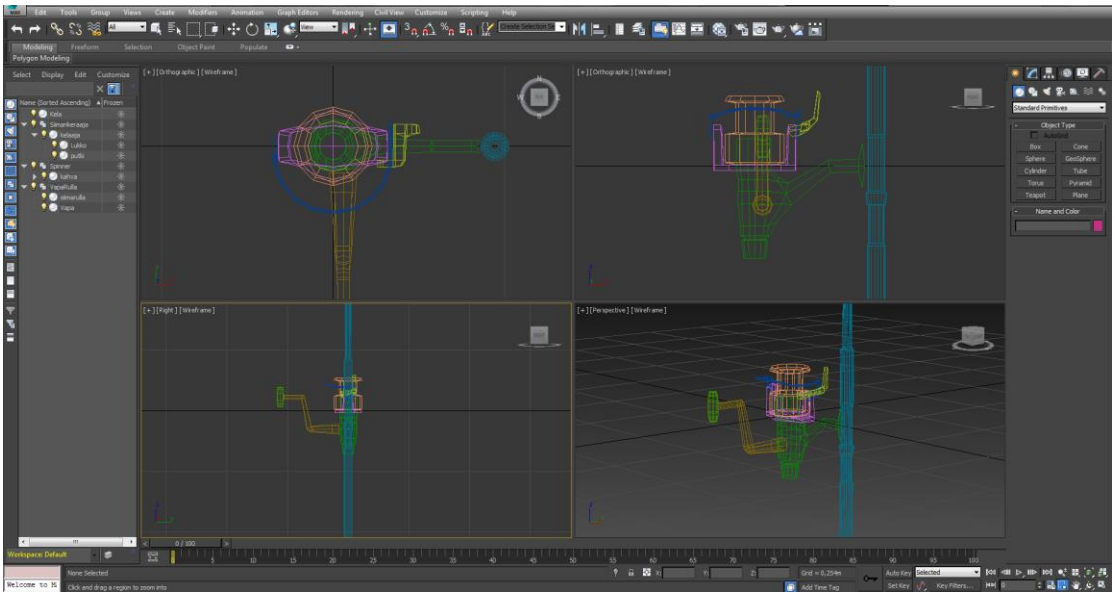
Yksinkertaisen laatikon polygonimallissa on 8 vertexiä, 12 reunaa ja 6 nelikulmaista polygonia. (Kuva 2)

3d-mallintamisessa on tärkeää miettiä mallin polygonimäärää. Mitä suurempi polygonimäärä, sitä kauemmin järjestelmällä menee aikaa sen renderöintiin. Polygonimäärää voi vähentää karsimalla osia mallista, jotka ovat toteutettavissa myöhemmin tekstuureihin piirtämällä. Myös turhat polygonit, jotka eivät ole pelaajan nähtävissä, on hyvä poistaa. (Silverman 2013.)

Toinen tärkeä asia on pitää polygonit nelikulmioina tai kolmioina. Kun malli viedään pelimoottoriin tai se otetaan ulos 3d-mallinnusohjelmasta, kaikista sen polygoneista tehdään kolmioita renderöinnin laskemisen helpottamiseksi. Menetelmä tekee sen yhdistelemällä pisteitä luomalla uusia reunoja niiden väliin. Jos mallissa on enemmän kuin neljä reunaa sisältäviä polygoneja, se

tuottaa ongelmia renderöintiin, koska niistä voidaan luoda kolmioita monella eri tavalla. (Silverman 2013.)

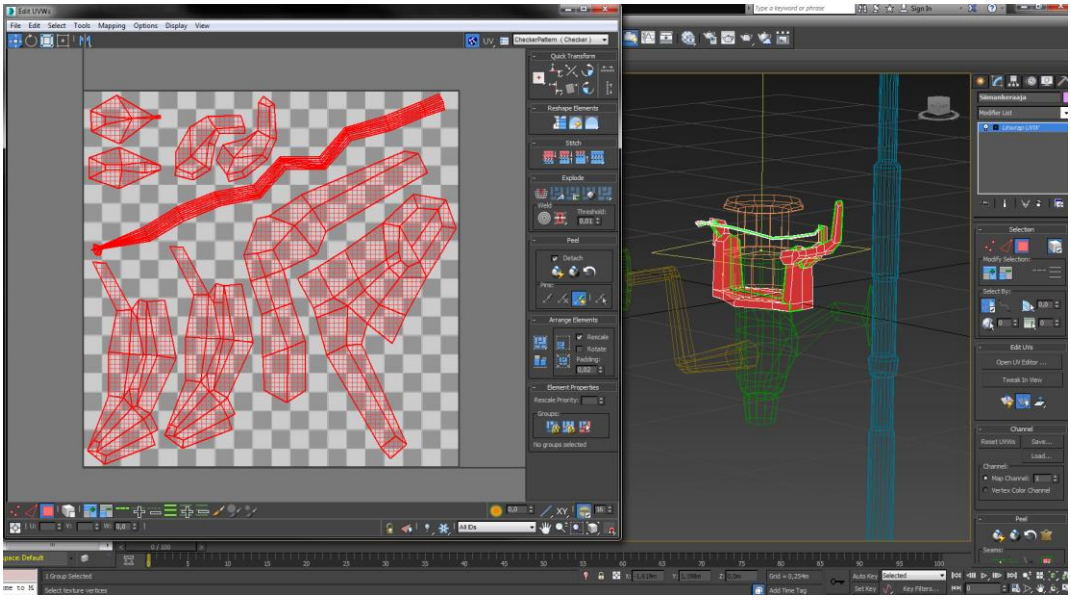
Virveliä mallintaessa pyrittiin pitämään polygonimäärä mahdollisimman pienenä, jotta se pyörisi mahdollisimman sulavasti vanhemmillakin mobiililaitteilla. Polygonimäärää vähennettiin karsimalla hieman osien pyöreydestä tekemällä niistä vähän kulmikkaampia. Lisäksi pienimmät pinnan muutokset jätettiin tehtäväksi myöhemmin tekstuurissa yksityiskohtia varjostamalla.



Kuva 3. Valmis virvelin 3d-malli 3ds Maxissa.

3ds Maxin perusnäkymässä malli näkyy eri suunnista neljästä ruudusta, joita voi muokata itselleen sopivaksi. Halutessaan valitun ruudun saa suurennettua isommaksi. Valmiista virvelin 3d-mallista jätettiin liikkuvat osat erilleen toisistaan, jolloin niiden ohjelmointi pelimoottorissa onnistuisi helpommin. Eri osat näkyvät eri väreillä näkymässä. (Kuva 3)

Polygonimallin valmistuttua oli aika tehdä UV map, joka näyttää kolmiulotteisen mallin pinnan kaksiulotteisessa tekstuurissa. UV mapin kaikki rajatut alueet vastaavat tiettyä kohtaa mallin pinnassa. Unwrapping on prosessi, jossa jokaiselle polygonille annetaan koordinaatit kaksiulotteiseen neliön muotoiseen kuvaan. (Slick 2016.)



Kuva 4. Unwrapping-prosessi.

Virveli jaettiin unwrapping-prosessissa eri osiin luoden useampi UV map. Tällä tavoin saatiin Unityssa muokattua paremmin eri osien olemusta, kuten kiiltävävät ja mattapintaiset osat erilleen. Mallista valitaan halutut tasot, jotka avataan UV-editoriin. Tasot jaotellaan asettamalla saumoja kohtiin, jotka eivät ole helposti nähtävissä, sillä renderöisessä saumakohdat voivat näkyä väärin. Jättäen mahdollisimman vähän turhaa tilaa kaikki tasot asetellaan mahtumaan rajatulle alueelle, joka tallennetaan malliksi tekstuurien tekoon. (Kuva 4)

Kuva 5. Virvelin siimankerääjän ja lukon valmis tekstuuri.

Seuraavaksi piirretään tekstuurit jollakin kuvankäsittelyohjelmalla hyödyntäen tallennettua UV mappia. Virvelin osien muotoja vahvistettiin lisäämällä yksityiskohtia, kuten varjostuksia, tekstejä ja ruuveja tekstuureihin. (Kuva 5)

Virvelin ja sen tekstuurien valmistuttua ne siirrettiin Unityyn. 3D-malli viedään ulos 3ds Maxista Export-toiminnolla ja se tallennetaan .fbx tiedostomuotoon. Virveli sekä tekstuurit siirrettiin erillään Unity-projektiin. Siirtäminen onnistuu raahaamalla ja pudottamalla tiedostot pelimoottoriin. Kun tiedostot on Unityssa, tekstuurit liitetään virveli peliobjektiin ja kaikki on valmista ohjelmointiin.

3.4 Ohjelmointi

Unity 3D -pelimoottorissa peliobjektien käyttäytymistä kontrolloidaan niihin liitetyillä komponenteilla. Unityssa on hyödyllisiä valmiiksi rakennettuja komponentteja, mutta objekteja ja komponenttien asetuksia voidaan myös hallinnoida itse kehitetyillä skripteillä, jotka laukaisevat pelin tapahtumia esimerkiksi ajan tai käyttäjän syötteen mukaan. Valittavia sisäänrakennettuja

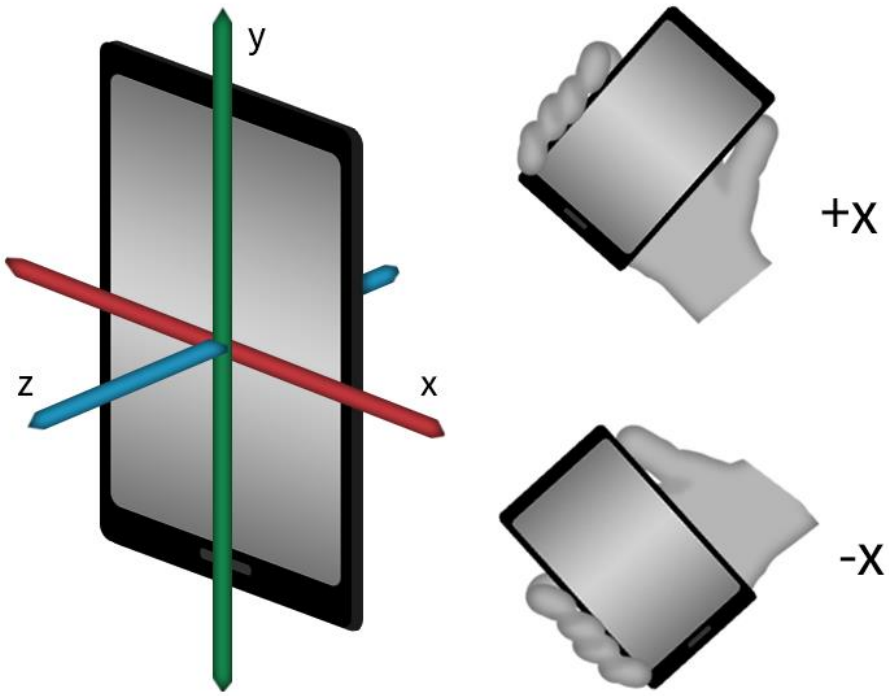
ohjelmointikieliä ovat C# sekä UnityScript, joka muistuttaa JavaScriptiä. (Unity Technologies 2016e.)

Kehitykseen valittiin ohjelmointikieleksi C#. Se on valmiiksi käännetty konekielelle tehden suorittamisesta nopeampaa. C# on myös vahvasti tyypitetty eli esimerkiksi kokonaisluku muuttuja ei voi pitää sisällään liukulukuja, mikä estää virheiden syntymistä. JavaScriptissä muuttuja voi pitää sisällään kaikenlaista dataa. (Gibson 2014, 254-259.)

3.4.1 Heittoliike

Heittoliikkeen havaitsemisessa hyödynnetään puhelimen kiihtyvyysanturia, joka tunnistaa puhelimen kallistuksen ja asennon. Mobiililaitteissa olevien kiihtyvyysanturien avulla varmistetaan, että kuvat esitetään oikeinpäin riippuen missä asennossa laitetta pidetään. Kiihtyvyysanturi auttaa kehittäjiä tekemään peleistä ja sovelluksista käyttäjäystävällisempiä. Anturi mittaa painovoimasta tai laitteen kallistuksesta aiheutuvaa kiihtyvyyttä ja pystyy tunnistamaan kulman, jossa laitetta pidetään. (Gujarati 2013.)

Mobiililaitteen kiihtyvyysanturi mittaa laitteen lineaarisen kiihtyvyyden vaihtelua kolmiulotteisessa avaruudessa. Kiihtyvyysanturi käyttää ensisijaisia x-, y- ja z-akseleita arvojen mittaamiseen. Laitetta pidettäessä pystyasennossa edessä x-akseli on positiivinen kallistaessa oikealle, y-akseli on positiivinen suoraan ylöspäin ja z-akseli on positiivinen osoittaessa itseä kohti. Arvot vaihtelevat -1 ja 1 välillä. (Unity Technologies 2016f.)



Kuva 6. Kiihtyvyysanturin antama x-arvo Unity 3D -pelimoottorissa.

Heittoliikkeen ohjelmoinnissa käytetään kiihtyvyysanturin antamaa x-akselin lukemaa, josta pystytään päättelemään puhelimen tämän hetkinen asento. Kallistaessa puhelinta oikealle, x-akselin arvo kasvaa, kun taas vasemmalle kallistaessa, arvo pienenee. (Kuva 6)

Pelaajan halutessa uistimen lentämään on hänen tehtävä heiton valmistelut ennen varsinaisen heittoliikkeen aloittamista. Jotta uistimen heitto on mahdollista, on uistimen oltava kelattuna nollassa metrissä ja virvelin lukko täytyy olla avattuna.

```
//THROWING MOTION
//update values
xForce = Input.acceleration.x; //current acceleration X
throwTimer-=Time.deltaTime*4f; //decrease throwtimer

//throw starting angle, phone tilted right (back)
//and distance 0m
if(xForce > 0.7f && kelattu == true)
{
    canThrow = true;
    setDistanceMax(); //sets max distance based on spoon
}
//throw motion started, phone tilted back to center
if(canThrow == true && xForce < 0.7f)
{
    canThrow = false;
    timerReset (); //resets timer to 1f
}
//throw motion ended, throw motion must be finished before timer gets to zero, calculate force
if(xForce < -0.6f && throwTimer>0f && throwed == false && OpenLock.lockOpen)
{
    throwed = true;
    kelattu = false;
    targetdistance = distanceMax * throwTimer; //targetdistance is maxdistance * timer (0-1f);
}
```

Ohjelmalistaus 1. C#-ohjelmointikielellä toteutettu heittoliike.

Heittoliike voi alkaa pelaajan kallistaessa puhelinta oikealle eli taakse heitettäessä virvelillä eteenpäin. Puhelimen kallistuessa pois lähtöasennosta asetetaan heitto alkaneeksi ja resetoidaan heittoajastin yhteen. Ajastinta pienennetään jatkuvasti. Jos ajastin on suurempi kuin nolla heittoliikkeen päätösasennossa, heitto on onnistunut ja lasketaan heiton pituus. Heiton voimakkuus määräytyy heiton nopeudesta eli mitä nopeammin heittoliike päättyy, sitä suuremmaksi jää ajastimen arvo, jolla kerrotaan heiton maksimipituus. (Ohjelmalistaus 1)

3.4.2 Taustan liike

Pelin realistisuuden parantamiseksi taustalle oli tärkeää saada liikettä. Virvelin taakse luotiin taustakuva, joka kallistuu oikealle ja vasemmalle. Tausta pyörii puhelimen kiihtyvyysanturin lukemien mukaan pitäen horisontin samana kuin oikeassa elämässä.

Kiihtyvyysanturin lukemat vaihtelevat herkästi ja se tunnistaa pienenkin liikkeen aiheuttaen nykimistä. Unity-dokumentaation lowpass-funktiolla voidaan tasoittaa lukemia ja päästään eroon arvojen herkkyydestä. (Unity Technologies 2016f.)

```
float AccelerometerUpdateInterval = 1.0f/100.0f;
float LowPassaWidthInSeconds = 0.1f;
Vector3 lowPassValue = Vector3.zero;

...

Vector3 temp = -(lowpass ());
temp.z = 0f;
transform.up = temp.normalized;

...

//this smooth the accelerometer values and gets rid of noise
Vector3 lowpass()
{
    float LowPassFilterFactor = AccelerometerUpdateInterval / LowPassaWidthInSeconds;
    lowPassValue = Vector3.Lerp (lowPassValue, Input.acceleration, LowPassFilterFactor);
    return lowPassValue;
}
```

Ohjelmalistaus 2. C#-ohjelmointikielellä toteutettu taustakuvan liikkeen pehmentäminen.

Aluksi alustetaan lowpass-funktion käyttämät arvot, jotka määrittävät, kuinka herkästi tausta liikkuu. Taustan liikkeen suuruus suodatetaan Update-metodissa kutsuttavalla lowpass-funktiolla, joka tekee liikkeestä sulavampaa. Liikkeen z-arvon nollaus estää taustan kallistumisen pelaajaa kohti. (Ohjelmalistaus 2)

Kuvan lisäksi taustalle luotiin lintuja. Linnut ilmestyvät taustalle satunnaisesti kuvan vasempaan tai oikeaan reunaan. Reunoilla ovat lintujen ilmestymiskohdat ja lintujen lentosuunta on pelialuetta kohti. Aina uuden linnun syntyessä ilmestymiskohta vaihtaa korkeutta, jotta linnut eivät lentäisi koko ajan samalla korkeudella.

```
//Decrease ratetimer
rateTimer -= Time.deltaTime;
if(rateTimer < 0f)
{
    randomizeSpawns (); //changes spawnpoints altitude

    //Load bird from resources, use pos/rot from birdSpawns
    bird = (GameObject)Instantiate (Resources.Load ("bird"), startPosition,startRotation);
    rateTimer = Random.Range(2f,15f); //randomize next bird
    side = Random.Range (0,2); //randomize left or right side
    Destroy(bird, 20f); //destroy bird after x seconds
}
```

Ohjelmalistaus 3. C#-ohjelmointikielellä toteutettu lintujen luominen.

Lintujen ilmestymistiheyttä hallitaan ajastimella, jota pienennetään jatkuvasti. Aina ajastimen ollessa pienempi kuin nolla vaihdetaan molempien ilmestymiskohtien korkeutta ja luodaan klooni bird-peliobjektista, joka ladataan Resources-kansiosta. Peliobjektin kloonaus onnistuu Unityn Instantiate-funktiolla, johon syötetään alkuperäinen peliobjekti, sijainti ja rotaatio (Unity

Technologies 2016g). Tämän jälkeen arvotaan seuraavan linnun ilmestymisaika 2 - 15 sekunnin väliltä ja seuraava ilmestymispuoli. Klooni tuhotaan 20 sekunnin jälkeen, kun lintu on ehtinyt lentää näytöltä ulos. (Ohjelmalistaus 3)

Jokaisella linnulla on oma skripti, joka arpoo linnulle värin harmaa, valkoinen tai musta. Update-metodissa lintua liikutetaan satunnaisella nopeudella peliruudun ohi Unityn Transform.Translate-funktiolla, joka liikuttaa objektia haluttujen akselien suuntaisesti (Unity Technologies 2016h). Tässä tapauksessa lintua liikutetaan x-akselin suuntaisesti oikealle tai vasemmalle.

3.4.3 Kelaaminen

Kelaaminen tapahtuu koskettamalla virvelin kahvaa ja pyörittämällä sitä vastapäivään, jolloin uistin tulee lähemmäs. Oikean virvelin tavoin myötäpäivään pyörittäessä kahva lukittuu aina tiettyyn asentoon, jos kahva on kelan vasemmalla puolella ja kelan lukitus on päällä.

```
//TOUCH MOVED
if(Input.GetTouch(0).phase == TouchPhase.Moved && touchingSpinner && kelaaja.canSpin)
{
    handleOff = false;
    handleFreeze (); //freezees handle,when grabbed

    Vector3 pos = Camera.main.WorldToScreenPoint(transform.position);
    Vector3 dir = (Vector3)Input.GetTouch(0).position - pos; //touchposition is vector2 -> convert to vector3
    angle = Mathf.Atan2(dir.y, dir.x) * Mathf.Rad2Deg;
    Quaternion targetRot = Quaternion.AngleAxis(angle, Vector3.forward);
    transform.rotation = Quaternion.RotateTowards(transform.rotation, targetRot, Time.deltaTime * speed);
}
```

Ohjelmalistaus 4. C#-ohjelmointikielellä toteutettu kelaaminen kosketuksella.

Unityn WorldToScreenPoint-funktio (Unity Technologies 2016i) muuttaa objektin pelimaailman sijainnin sijainniksi näytöllä. Aina kun pelaaja koskettaa näyttöä, verrataan kosketuksen ja kahvan sijaintia näytöllä ja lasketaan uuden kulman suuruus ja pyöritetään kahvaa kosketuksen suuntaan. (Ohjelmalistaus 4)

Varsinainen kela pyörii pelaajan koskettaman kahvan mukana. Kela kopioi liikkeensä kahvan asennosta, mutta pyörii tuplasti nopeammin eli kahvan pyöriessä puoli kierrosta kela on jo pyörinyt täyden kierroksen.

3.4.4 Kameran liike

Kameralla on hidas elävöittävä pyörimisliike koko ajan. Vedessä kalan nykiessä pyörimisliikkeen nopeutta kasvatetaan hetkellisesti huomattavasti luoden tärinäefekti näytölle.

```
//Zooms out so the fish wont appear small
void zoomOut()
{
    Camera.main.fieldOfView = Mathf.Lerp (Camera.main.fieldOfView,zoomOutTarget,zoomSpeed * Time.deltaTime);
    this.transform.rotation = Quaternion.RotateTowards(this.transform.rotation, showcameraRot, cameraturnSpeed*Time.deltaTime);

    //Disable all scripts in virveli
    MonoBehaviour[] scripts = virveli.GetComponentsInChildren<MonoBehaviour>();
    foreach(MonoBehaviour script in scripts)
    {
        script.enabled = false;
    }
    virveli.transform.rotation = Quaternion.RotateTowards(virveli.transform.rotation,targetvirveliRot, 20f *Time.deltaTime);
}
void zoomIn()
{
    Camera.main.fieldOfView = Mathf.Lerp (Camera.main.fieldOfView,zoomInTarget,zoomSpeed * Time.deltaTime);
    this.transform.rotation = Quaternion.RotateTowards(this.transform.rotation, normalcameraRot, cameraturnSpeed*Time.deltaTime);

    virveli.transform.rotation = Quaternion.RotateTowards(virveli.transform.rotation,normalvirveliRot, 20f *Time.deltaTime);

    //enable all scripts when back to startrotation
    if(virveli.transform.rotation == normalvirveliRot)
    {
        MonoBehaviour[] scripts = virveli.GetComponentsInChildren<MonoBehaviour>();
        foreach(MonoBehaviour script in scripts)
        {
            script.enabled = true;
        }
    }
}
```

Ohjelmalistaus 5. C#-ohjelmointikielellä toteutetut kameran zoomaus-funktiot.

Kalan tai uistimen näyttöön kameralle luotiin zoomaus-funktiot, jotka kasvattavat ja pienentävät näkökenttää, jotta kala mahtuu näytölle eikä vaikuta liian pieneltä. Funktioissa myös käännetään kameraa hieman ja aktivoidaan sekä deaktivoidaan kaikki virvelipeliobjektin skriptit. Tällä tavoin estetään pelaajaa käsittelemästä virveliä, kun kalaa tai uistinta näytetään. (Ohjelmalistaus 5)

3.4.5 Kalan saanti

Kalan saamisen ja käyttäytymisen ohjelmoinnissa tavoiteltiin vaihtelevuutta. Kalan syöntiyritykset ja kiinniotto koodattiin mahdollisimman epäsäännölliseksi, mutta niihin pystyy kuitenkin vaikuttamaan uistimen valinnoilla ja kelausnopeudella.

Ohjelmoinnissa hyödynnettiin Unityn Random.Range-funktiota syönnin satunnaistamiseksi. Funktio ottaa vastaan minimi- ja maksimiarvon, jonka jälkeen se palauttaa satunnaisen arvon niiden väliltä. Arvot voivat olla

kokonais- tai liukulukuja. Tällä vaihtelevalla luvulla saadaan hallittua kalan syöntitapahtumien alkamisia sekä onnistumisia. (Unity Technologies 2016j.)

```
//Bitetimer lower then zero
if(nextbiteTimer < 0f)
{
    nextbiteTimer = Random.Range(2f,10f); //Randomize next bite
    DoesFishBite(); //check bite
}
```

Ohjelmalistaus 6. C#-ohjelmointikielellä toteutettu kalan puraisun ajastaminen.

Aina kelatessa ja uistimen ollessa vedessä vapaana puremisen ajastinta pienennetään ajan mukaan. Ajastimen ollessa negatiivinen arvotaan seuraavan puraisun ajankohta asettamalla ajastin kahden ja kymmenen sekunnin välille. Tämän jälkeen kutsutaan kerran funktiota, joka katsoo nappasiko kala kiinni uistimeen. (Ohjelmalistaus 6)

```
//Bite check
void DoesFishBite()
{
    //Set bite chances based on current spoon (fishChance,junkChance)
    SetBiteChances();

    //First check if fish bites
    randomBite = Random.Range (0,fishChance);
    if(randomBite == 0 && !junkHooked && !junkCreated)
    {
        fishHooked = true; //set fish is hooked
        DoesFishEscape(); //check will it escape
    }
    else
        fishHooked = false;

    //Fish did not bite, check junkChance
    randomBite = Random.Range(0,junkChance);
    if(randomBite == 0 && !fishHooked && !fishCreated)
    {
        junkHooked = true; //set junk
    }
    else
        junkHooked = false;
}
```

Ohjelmalistaus 7. C#-ohjelmointikielellä toteutettu kalan tarttuminen syöttiin.

Aluksi asetetaan kalan ja muun roinan tarttumisen todennäköisyysarvot, jotka vaihtelevat eri uistimien mukaan. Seuraavaksi arvotaan kokonaisluku nollan ja asetetun todennäköisyysarvon väliltä. Kokonaisluvun ollessa nolla kala tarttuu kiinni, jonka jälkeen katsotaan vielä pääseekö se karkuun, mikä tapahtuu

epäsäännöllisen ajan kuluttua. Jos kala ei syö, käydään muun roinan tarttuminen samalla tavalla läpi. (Ohjelmalistaus 7)

Kalan ollessa kiinni uistimessa se nykii satunnaisin väliajoin. Nykiessä kamera tärisee ja puhelin värisee. Puhelimen värinä laukaistaan Unityn Handheld.Vibrate-komennolla (Unity Technologies 2016k). Laukaistessa puhelin värisee tasaisesti sekunnin ajan, joten siihen luotiin epäsäännöllisyyttä vaihtelemalla sen tiheyttä luoden tuplavärinät satunnaisesti. Tällä tavoin jokaisen kalan kelaaminen on erilainen.

Kalan kelattuaan pelaajan täytyy nostaa se ylös. Se tapahtuu kallistamalla puhelin pystyasentoon, jolloin virveli pyörii takaisin perusasentoon ja jokin kalalaji arvotaan ja kalasta luodaan klooni ruudulle. Kalan tiedot ilmestyvät ruudulle kalan kanssa. Tiedoissa ovat lajin nimi, pituus ja paino. Pelaaja voi joko ottaa kalan talteen tai päästää sen vapaaksi. Ottaessaan sen kala tallennetaan listalle, jossa näkyy saatujen kalojen määrä, minimi-, maksimi- ja kokonaispainot.

3.4.6 Tietojen tallennus

Peliin lisättiin pelattavuutta kasvattamalla uistimia saadun kalamäärän mukaan. Pelaaja saa aluksi käyttöönsä kolme perusuistinta. Uistimia on kokonaisuudessaan 12 kappaletta ja ne avautuvat kalasaaliin kasvaessa. Lisäksi pelaajalla on saaduista kaloistaan lista, josta hän näkee etenemistään. Jotta nämä elementit saataisiin pysymään, oli peliin ohjelmoitava tietojen tallennus.

Sovelluksessa tietojen tallennus tapahtuu lokaalisti. Pelissä on peliobjekti, jonka komponenttina on tietojen tallennus skripti, joka pitää sisällään onko äänet päällä, kalalistan tiedot, nykyisen uistimen, avatut uistimet ja seuraavan uistimen aukeamiseen tarvittavan kalamäärän. Tätä peliobjektia ei tuhota missään vaiheessa peliä, jotta tiedot eivät katoaisi.

Peliobjektin tuhoutumisen saa estettyä kutsumalla DontDestroyOnLoad-funktiota. Tämä säilyttää peliobjektin ja sen komponentit kokonaisuudessaan näkymän vaihtuessa. Unityssa pelinäkymän vaihtuessa kaikki näkymässä olevat objektit tuhotaan ja uuden näkymän objektit luodaan. (Unity Technologies 2016l.)

Jotta tiedot pystytään tallentamaan, on luotava serialisoitu luokka, joka pitää sisällään tallennettavat tiedot. Serialisoinnilla muunnetaan data muotoon, joka voidaan tallentaa ja myöhemmin palauttaa. [Serializable]-komento kertoo Unitylle, että skripti voidaan serialisoida eli kaikki muuttujat voidaan tallentaa. (Daily 2014.)

```
[Serializable]
class PlayerData
{
    public bool soundOff;
    public int spoonIndex;
    public int ownedSpoons;
    public int newSpoonLimit;

    public int perchCount;
    public float perchMax;
    public float perchMin;
    public float perchTotal;

    public int pikeCount;
    public float pikeMax;
    public float pikeMin;
    public float pikeTotal;

    public int pikeperchCount;
    public float pikeperchMax;
    public float pikeperchMin;
    public float pikeperchTotal;

    public int salmonCount;
    public float salmonMax;
    public float salmonMin;
    public float salmonTotal;

    public int troutCount;
    public float troutMax;
    public float troutMin;
    public float troutTotal;

    public int alltimeCount;
    public float alltimeWeight;
}
```

Ohjelmalistaus 8. C#-ohjelmointikielellä toteutettu serialisoitu luokka, jossa tallennettavat tiedot muuttujina.

Peliin luotiin serialisoitu PlayerData-luokka sisältäen kaikki halutut tallennettavat tiedot. Luokka sisältää totuusarvomuuttujan eli onko äänet päällä vai pois sekä kokonais- ja liukulukuja uistimista, kalojen määristä ja painoista. (Ohjelmalistaus 8)

Seuraavaksi tehtiin tallennus-funktio, joka asettaa pelaajan tiedot sisältävän peliobjektin muuttujien arvot serialisoidun luokan muuttujiin ja tallentaa ne tiedostoon. Tietojen tallennuksessa tarvitaan serialisoinnin hallinnointiin

BinaryFormatter sekä File.Create-funktio tiedoston luontiin. File.Create-funktio luo uuden tiedoston siihen asetettuun tiedostopolkuun. (Daily 2014.)

```
//Saves players Data
public void Save()
{
    //New binaryformatter
    BinaryFormatter bf = new BinaryFormatter();

    //Create file, put in data, serialize, close file
    FileStream file = File.Create(Application.persistentDataPath +"/playerInfo.dat");

    PlayerData data = new PlayerData();
    data.soundOff = soundOff;
    data.spoonIndex = currentSpoonIndex;
    data.ownedSpoons = ownedSpoonsCount;
    data.newSpoonLimit = newSpoonValue;

    data.perchCount = perchCount;
    data.perchMax = perchMax;
    data.perchMin = perchMin;
    data.perchTotal = perchTotal;

    data.pikeCount = pikeCount;
    data.pikeMax = pikeMax;
    data.pikeMin = pikeMin;
    data.pikeTotal = pikeTotal;

    data.pikeperchCount = pikeperchCount;
    data.pikeperchMax = pikeperchMax;
    data.pikeperchMin = pikeperchMin;
    data.pikeperchTotal = pikeperchTotal;

    data.salmonCount = salmonCount;
    data.salmonMax = salmonMax;
    data.salmonMin = salmonMin;
    data.salmonTotal = salmonTotal;

    data.troutCount = troutCount;
    data.troutMax = troutMax;
    data.troutMin = troutMin;
    data.troutTotal = troutTotal;

    data.alltimeCount = fishCountAlltime;
    data.alltimeWeight = fishWeightAlltime;

    bf.Serialize(file,data);
    file.Close();
}
```

Ohjelmalistaus 9. C#-ohjelmointikielellä toteutettu tietojen tallentaminen.

Aluksi tehdään BinaryFormatter, jonka jälkeen luodaan playerInfo.dat-tiedosto ja annetaan sille sijainti, joka on sama kuin sovelluksella. Tämän jälkeen tehdään PlayerData-luokan olio data ja asetetaan arvot sen muuttujiin, jonka jälkeen serialisoidaan ja suljetaan tiedosto. (Ohjelmalistaus 9)

Tietojen tallennus suoritetaan aina pelaajan saadessa kalan tai uuden uistimen. Aina arvojen muuttuessa on tiedot tallennettava, jotta estetään tiedon katoaminen pelaajan sulkiessa pelin suoraan ilman valikoiden

läpikäymistä. Lisäksi pelistä poistuessa tiedot tallennetaan vielä viimeisen kerran.

Tietojen tallentamisessa tiedosto avataan, jonka jälkeen syötetään sen sisälle arvot ja suljetaan. Lataaminen tapahtuu samalla tavalla, mutta nyt vain haetaan tiedot tiedostosta. File.Open-funktio avaa jo olemassa olevan tiedoston siihen syötetystä tiedostopolusta (Daily 2014).

```
//Loads players Data
public void Load()
{
    //If file exists
    if(File.Exists(Application.persistentDataPath + "/playerInfo.dat"))
    {
        //Open file take data out, Close file
        BinaryFormatter bf = new BinaryFormatter();
        FileStream file = File.Open(Application.persistentDataPath + "/playerInfo.dat", FileMode.Open);
        PlayerData data = (PlayerData)bf.Deserialize(file);
        file.Close();

        //now put data
        soundOff = data.soundOff;
        currentSpoonIndex = data.spoonIndex;
        ownedSpoonsCount = data.ownedSpoons;
        newSpoonValue = data.newSpoonLimit;

        perchCount = data.perchCount;
        perchMax = data.perchMax;
        perchMin = data.perchMin;
        perchTotal = data.perchTotal;

        pikeCount = data.pikeCount;
        pikeMax = data.pikeMax;
        pikeMin = data.pikeMin;
        pikeTotal = data.pikeTotal;

        pikeperchCount = data.pikeperchCount;
        pikeperchMax = data.pikeperchMax;
        pikeperchMin = data.pikeperchMin;
        pikeperchTotal = data.pikeperchTotal;

        salmonCount = data.salmonCount;
        salmonMax = data.salmonMax;
        salmonMin = data.salmonMin;
        salmonTotal= data.salmonTotal;

        troutCount = data.troutCount;
        troutMax = data.troutMax;
        troutMin = data.troutMin;
        troutTotal = data.troutTotal;

        fishCountAlltime = data.alltimeCount;
        fishWeightAlltime = data.alltimeWeight;
    }
}
```

Ohjelmalistaus 10. C#-ohjelmointikielellä toteutettu tietojen lataaminen.

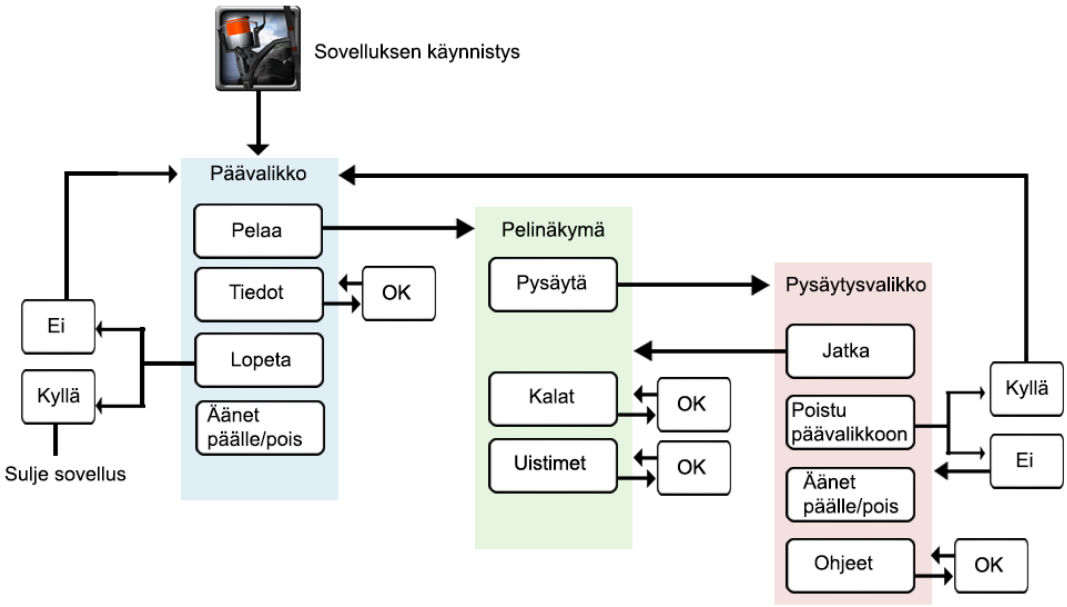
Aluksi tarkistetaan onko kyseistä tiedostoa olemassa. Jos playerInfo.dat-tiedosto on olemassa, se avataan, haetaan tiedot ja suljetaan. Kun data on haettu, voidaan asettaa tiedot peliobjektin muuttujiin. Jos tiedostoa taas ei ole, ei tehdä mitään. (Ohjelmalistaus 10)

Tietojen lataaminen tapahtuu aina pelin käynnistyessä. Arvot säilyvät jatkossa tuhoutumattomassa peliobjektissa, joten lataus tarvitsee suorittaa vain kerran.

Jos tiedostoa ei ole vielä olemassa ja kyseessä on pelin ensimmäinen käynnistys, tietoja ei ladata ja jatketaan alkuarvoilla.

3.4.7 Valikot ja käyttöliittymä

Peliin määriteltiin kaksi näkymää päävalikko sekä pelinäkymä. Unityssa voidaan jakaa näkymät eri tasoihin. Luodessaan uuden projektin Unity luo uuden tyhjän näkymän muutamilla oletuspeliobjekteilla, kuten pääkamera ja valo riippuen onko projekti kaksi- vai kolmeulotteinen. Näkymiä voidaan käyttää esimerkiksi luomaan päävalikko ja yksittäisiä tasoja. (Unity Technologies 2016m.)

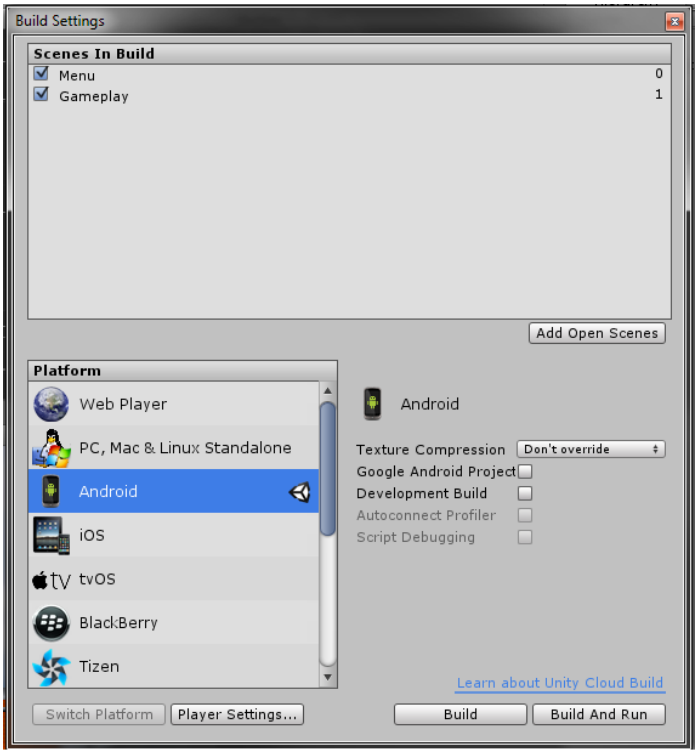


Kaavio 1. Valikoiden kulkukaavio.

Sovelluksen käynnistyessä ladataan päävalikko, jossa on vaihtoehdot pelaa, tiedot ja lopetus. Pelin asetuksissa on vain äänien käyttöönotto, joten asetuksilta jätettiin pois oma alivalikko ja lisättiin mute-näppäin muiden valintojen alapuolelle. Pelaa vaihtoehdosta siirrytään varsinaiseen pelinäkymään, jossa on vaihtoehdot saadut kalat, uistimet sekä pysäytä. Pysäyttäessä avautuu pysäytysvalikko pelinäkymän keskelle ja peli keskeytyy. Pysäytysvalikosta voidaan poistua päävalikkoon, jatkaa peliä, asettaa äänet tai lukea ohjeet. (Kaavio 1)

Taso ladataan kutsumalla Application.LoadLevel()-funktiota, joka ottaa vastaan näkymän nimen tai indeksin, joka ladattavalla tasolla on. Ennen tason

lataamista se pitää lisätä projektiin koontiversioasetuksissa, joka löytyy Unityn valikosta File->Build Settings. (Unity Technologies 2016n.)



Kuva 7. Unityn Build Settings-ikkuna.

Päävalikko ja pelinäkymä lisätään koontiversioasetuksissa painamalla Add Open Scenes-nappia, joka lisää kaikki avoimet näkymät luetteloon. Luettelon oikeassa reunassa näkyy näkymän indeksi. (Kuva 7)

```
//Receives scene index and loads it
public void NextLevelButton(int index)
{
    Application.LoadLevel(index);
}
//Close App
public void CloseGame()
{
    Application.Quit();
}
```

Ohjelmalistaus 11. C#-ohjelmointikielellä toteutettu näkymän vaihtaminen.

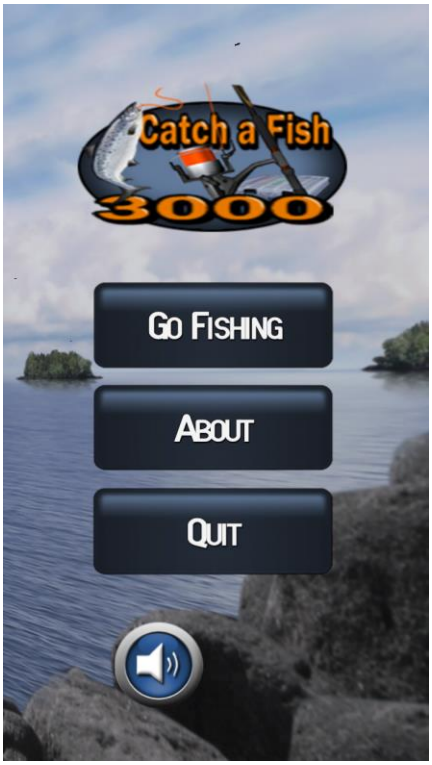
Valikoiden vaihtoehtojen skripti sisältää funktiot seuraavan tason lataamiseen ja sovelluksen sulkemiseen. Esimerkiksi pelinäkymään siirryttäessä kutsutaan tason lataus funktiota, jolle annetaan indeksiksi yksi. Sovellus suljetaan komennolla Application.Quit(). (Ohjelmalistaus 11)

Verkkosivun tai sovelluksen käytettävyydellä tarkoitetaan kuinka helppoa käyttöliittymää on käyttää. Käytettävyys on jaettu viiteen laatuominaisuuteen:

Opittavuus, tehokkuus, muistettavuus, virheettömyys ja kuinka miellyttävä sitä on käyttää. Opittavuudella tarkoitetaan kuinka helposti käyttäjä pystyy suorittamaan perustehtäviä ensimmäisellä kerralla ja tehokkuudella kuinka nopeasti käyttäjä pystyy toimimaan oppiessaan käytön. Muistettavuudella tavoitellaan käyttäjän kykyä tehokkaaseen käyttöön palatessaan pidemmän ajan jälkeen. Virheettömyydessä estetään käyttäjää tekemästä virheitä ja virheiden tapahtuessa autetaan palautumaan niistä. (Nielsen 2012.)

Käyttäjäystävällisyyttä lisäämään sovelluksen käyttöliittymää kehitettäessä pyrittiin pitämään toiminnot ja siirtymiset helposti ymmärrettävissä, jotta pelaaminen olisi helppoa jo ensimmäisellä avauskerralla. Valikoiden vaihtoehdot pidettiin vähäisinä estäen turhaa pelaajan hämmennystä. Kaikki sovelluksen napit luotiin samalla pohjalla ja saman muotoisiksi pitäen ne helposti tunnistettavina ja lisäksi näytön ollessa pieni pyrittiin pitämään napit suhteellisen isoina, jotta niitä on helppo painaa peukalolla tai etusormella. Isojen virheiden tekemisen estämiseksi luotiin tarkistus kysymys aina ennen isoja toimintoja, kuten päävalikkoon poistuminen, sovelluksen sulkeminen ja kalalistan tyhjennys.

Itse virvelin käsittely pelinäkymässä jätettiin tarkoituksella hieman pimentoon saaden pelaaja itse kokeilemaan toimintoja virveliä koskettamalla. Jos pelaajalle tuottaa vaikeuksia käsitellä virveliä, löytyy pysäytysvalikosta nappi, josta saa ohjeet näytölle tarvittaessa.



Kuva 8. Päävalikko.

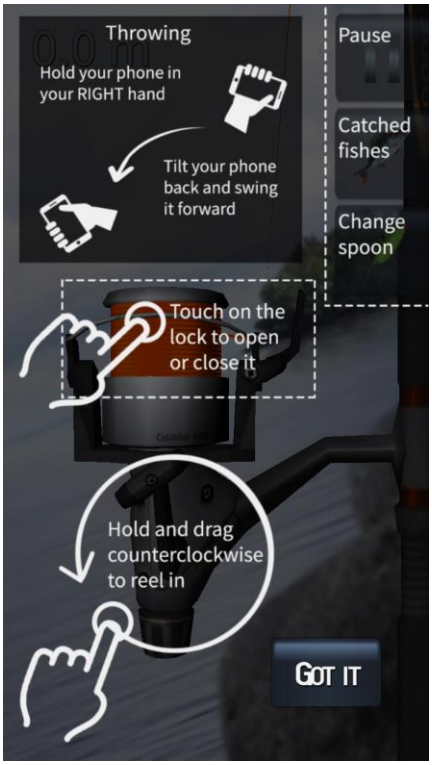
Päävalikossa näkyy logon lisäksi Go Fishing-painike pelinäkymään siirtymiseen, About-painike pelin tietojen tarkasteluun sekä Quit-painike pelin sulkemiseen. Lisäksi ääniasetusten mute-näppäin löytyy muiden painikkeiden alapuolelta. Painikkeet ovat isoja ja sijoitettu keskelle ruutua helpottamaan niiden painamista. (Kuva 8)

Kuva 9. Pelinäkymä.

Pelinäkymään on sijoitettu kolme painiketta oikeaan yläreunaan vavan varteen. Pysäytys-, kalalista- ja uistimet-painike. Jos virveliä pidetään pelin tarkoituksen mukaisesti oikeassa kädessä, painikkeita voi helposti hallinnoida peukalolla tai tarvittaessa vasemmalla kädellä. Lisäksi ruudun vasemmassa yläkulmassa on matka, kuinka pitkällä uistin on vavasta. (Kuva 9)

Kuva 10. Pysäytysvalikko.

Pysäytys-painiketta painaessa pelin aika pysäytetään ja ruutu jäädytetään paikoilleen. Pysäytysvalikosta löytyy vaihtoehdot jatka tai palaa päävalikkoon. Lisäksi mute- sekä ohjenäppäin. (Kuva 10)



Kuva 11. Ohjeet näkyvissä.

Ohje-näppäintä painaessa peli pysyy pysähdyksissä mutta pysäytysvalikko piilotetaan. Ohjeissa käytetään tekstin lisäksi kuvia havainnollistamaan toimintoja. Ohjeista poistuessa palataan takaisin pysäytysvalikkoon. (Kuva 11)

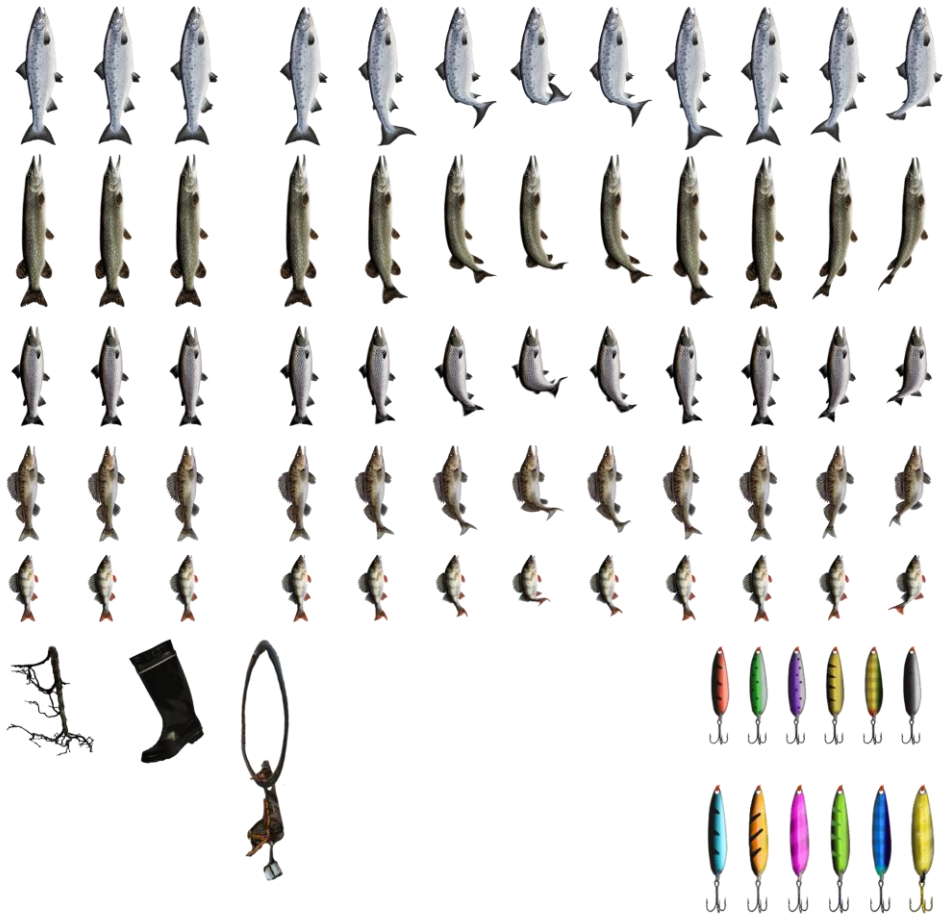
Kuva 12. Kala ruudulla.

Kalan ollessa ruudulla sen tiedoissa ilmenee suomen- ja englanninkielinen nimi, pituus ja paino. Pelaajalla on mahdollisuus ottaa kala talteen tai päästää se irti. Ottaessaan kala tallentuu pelaajan kalalistaan. (Kuva 12)

3.5 Grafiikka

Taustakuva ja virveli viimeisteltiin suoraan lopulliseen muotoonsa jo ennen ohjelmointia. Ohjelmoinnissa käytetty placeholder-grafiikka päivitettiin näyttävämmäksi pelin ollessa pelattavassa muodossa. Muuta grafiikkaa ovat kalat, roina, uistimet ja valikoissa näkyvät napit ja reunat.

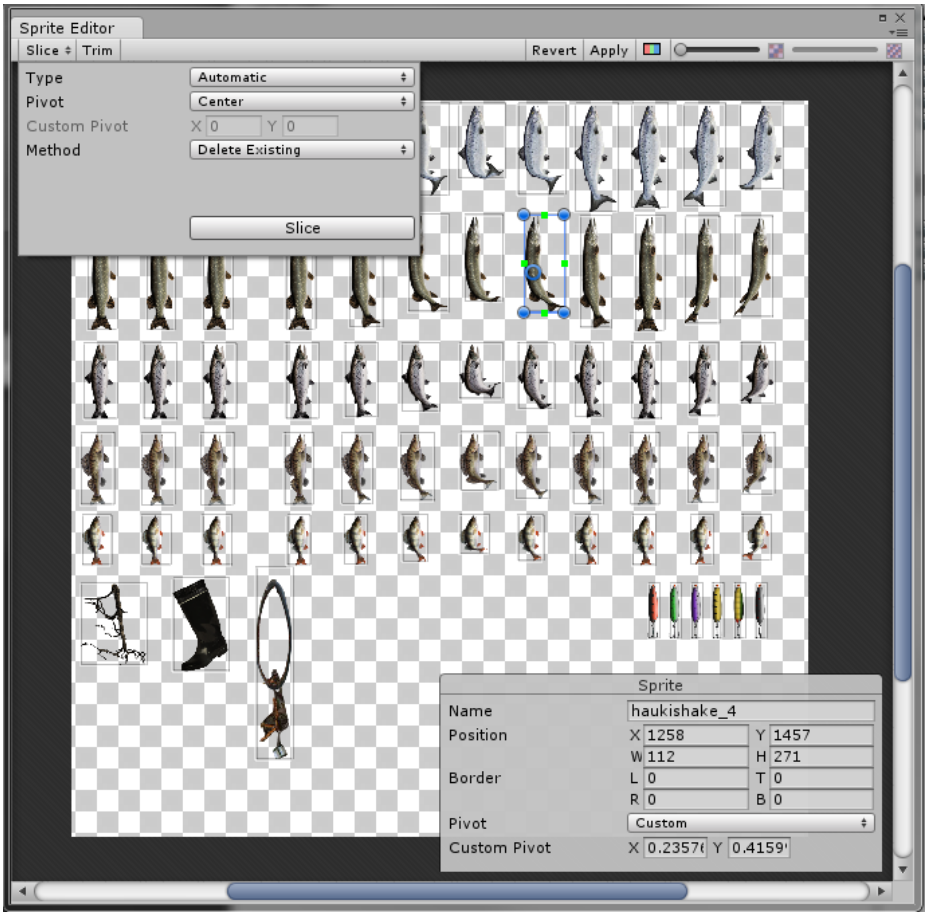
Muu grafiikka luotiin kaksiulotteisena Unityn Sprite-editoria käyttämällä. Sprite tekstuuri voi sisältää yhden elementin, mutta yleisimmin monta elementtiä yhdistettynä yhteen isompaan kuvaan. Sprite-editoria käyttääkseen tekstuuri pitää asettaa 2d ja UI tyyppiseksi Spriteksi. Sprite-editorilla kuva voidaan leikata useampaan osaan luoden yhdestä kuvasta monta elementtiä peliin. (Unity Technologies 2016o.)



Kuva 13. Sprite-tekstuuri.

Kalat, roina ja uistimet piirrettiin yhteen suureen kuvaan, joka tallennettiin .png tiedostomuotoon ja lisättiin Unityyn. Kalat piirrettiin eri asennoissa myöhempää animontia varten. (Kuva 13)

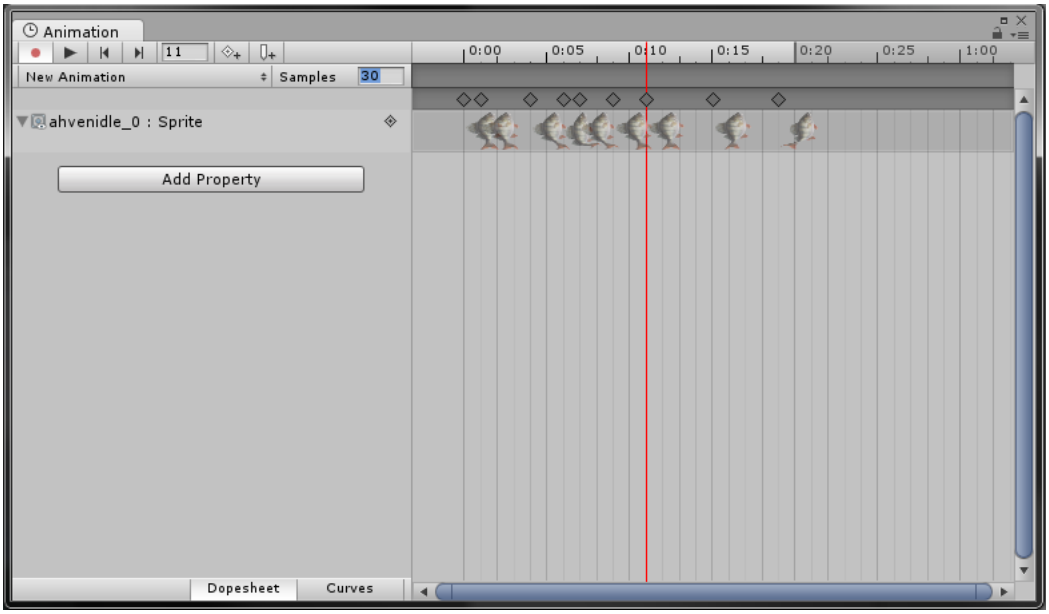
Unityn Sprite-editorissa on mahdollista leikata kuvat manuaalisesti pilkkomalla kuvat paloihin tai antaa Unityn tehdä sen automaattisesti. Automaattisessa Sprite-editor tunnistaa elementtien rajat läpinäkyvyyden mukaan. (Unity Technologies 2016o.)



Kuva 14. Sprite-editor Unityssa.

Kaksiulotteinen kuva jaettiin automaattisesti osiin. Leikatut palat eroteltiin nimeämällä ne elementin mukaan. Kalojen kuvia korjattiin muuttamalla kalojen keskipisteen sijaintia kalan keskelle, jolloin kalat pysyvät samassa kohtaa kuvan vaihtuessa animoinnissa. (Kuva 14)

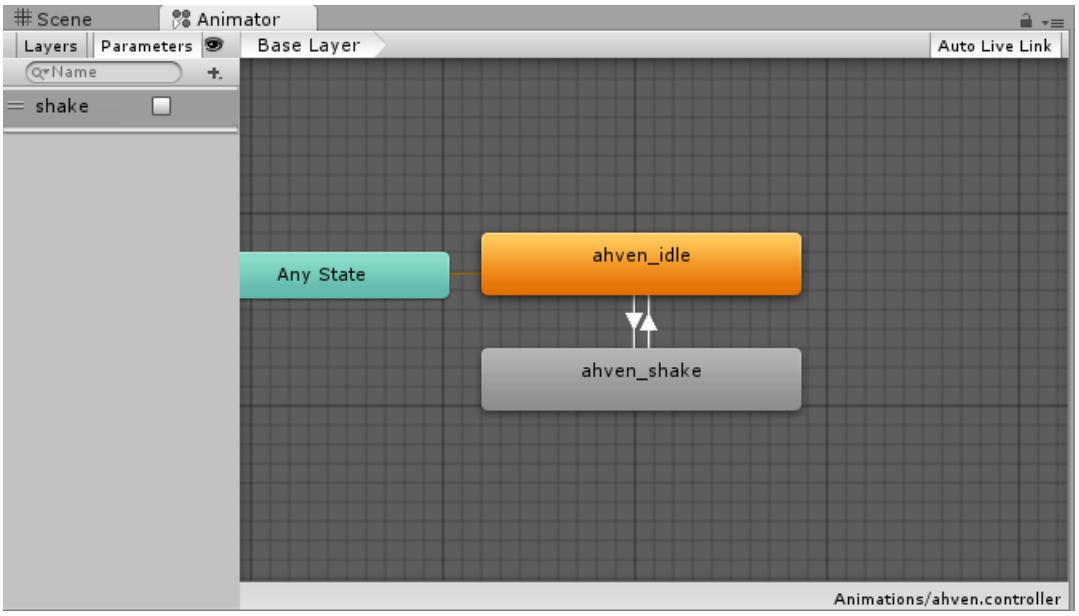
Animointi Unityssa tapahtuu Animation-ikkunassa. Leikatut kuvat raahataan ikkunaan, jossa kuvien vaihtuvuus nopeutta voi säätää. Yleisesti peleissä yksi animaatio ei riitä esittämään kaikkia objektin liikkeitä. Unityssa voi tehdä useamman erilaisen animaation objektille sen eri liikkeisiin. (Dominguez 2014.)



Kuva 15. Animation-ikkuna Unityssa.

Kaloille tehtiin omat animaationsa raahaamalla ne Animation-ikkunaan. Ikkunassa kuvat aseteltiin haluttuun järjestykseen ja nopeus säädettiin sopivaksi. Jokaiselle kalalle luotiin kaksi animaatiota idle ja shake. Idle näyttää kalan paikoillaan liikuttaen hieman eviä ja suuta, kun shake heiluttaa koko pyrstöä puolelta toiselle nopeasti. (Kuva 15)

Tehtyjä animaatioita hallinnoidaan Animator-ikkunassa, jossa voi tehdä yhden tai useamman tilan, joka pyörittää yhtä animaatiota. Tilojen välille voidaan luoda siirtymisiä muuttujan tai kuluneen ajan mukaan. Animaatio vaihtuu siirtyessä tilasta toiseen. (Dominguez 2014.)



Kuva 16. Animator-ikkuna Unityssa.

Animaattorissa animaatiot jaettiin kahteen eri tilaan. Siirtymiseen määritettiin totuusarvomuuttuja shake, jonka ollessa tosi siirrytään shake-animaatioon ja toisin päin. (Kuva 16)

```
//Fish movement, All animations have shake bool which determites state
if(shake == true)
{
    //Set animation to shake
    anim.SetBool("shake",true);
    //Rotate fishGameObject left and right
    transform.rotation = Quaternion.Euler(0f,0f,maxRot * Mathf.Sin (Time.time * speed));
}
else
{
    //animation idle, rotate fishGameObject back to startRotation
    anim.SetBool("shake",false);
    this.transform.rotation = Quaternion.RotateTowards(this.transform.rotation, startRot, 1f);
}
```

Ohjelmalistaus 12. C#-ohjelmointikielellä toteutettu kalan animaatiotilan vaihtaminen.

Kalan liikkeen skriptissä on totuusarvomuuttuja shake, jonka ollessa tosi animaattorin totuusarvomuuttuja shake asetetaan todeksi, jolloin animaattori toistaa kalan shake-animaatiota. Kalan heilumista korostetaan kallistamalla kuvaa edestakaisin nopeasti ja idle-animaatiossa kuva käännetään takaisin alkuperäiseen pystyasentoon. (Ohjelmalistaus 12)

3.6 Äänet

Äänimaailma on vahvasti läsnä kalastamisessa, joten peliin oli saatava paljon ääniä luomaan oikeanlaista tunnelmaa. Internetistä löytyy paljon valmiiksi äänitettyjä luonnonääniä. Taustalla kuuluvat luonnonäänet luotiin ilmaisia ja julkisia ääniä muokkaamalla. Pelaajan käsittelemän virvelin ääniefektit äänitettiin itse omalla puhelimella oikeata virveliä käyttämällä.

Unityssa peliobjektiin liitetty Audio Source-komponentti hallinnoi sen ääniä. Komponentille annetaan äänitiedosto, jota voidaan käyttää neljällä komennolla Play(), Stop(), Pause(), ja PlayOneShot(). Komentojen sulkeiden sisään lisätään toistettava äänitiedosto. PlayOneShot()-komento toistaa äänen kerran ja se on parempi ääniefektien kutsumiseen kuin Play(), joka on sopivampi jatkuvan äänen toistamiseen. (Goldstone 2009, 114.)

Saadakseen äänet kuulumaan ei riitä vain niiden toistaminen. Lisäksi tarvitaan yksi Audio Listener-komponentti liitettynä yhteen peliobjektiin. Yleisimmin Audio Listener on liitettynä pääkameraan. Audio Listener-komponenttia voi olla vain yksi samassa pelinäkymässä ja se toimii mikrofonin tavoin

kuunnellen kaikkia ääniä ympärillään, jotka ovat lähtöisin eri Audio Source-komponenteista. (Unity Technologies 2016p.)

Sovelluksen Audio Listener on sijoitettu pääkameraan. Pelaajalle annetaan mahdollisuus asettaa pelin äänet päälle tai pois mute-näppäimestä, joka löytyy päävalikosta sekä pause-valikosta. Näppäintä painettaessa Audio Listenerin äänenvoimakkuus asetetaan nollaan, jolloin peli on äänetön.

Pelissä jokaiselle ääniä sisältävälle peliobjektille on asetettu oma Audio Source-komponentti. Ääniklippi toistetaan kutsumalla sitä peliobjektin omassa skriptissä haluttuun aikaan. Taustaäänenä kuuluva veden lotina aloitetaan heti pelin käynnistyttyä Play()-komennolla ja se pyörii taustalla jatkuvasti. Muita ääniä kutsutaan PlayOneShot()-komennolla, joka toistaa äänen vain kerran.

```
//AUDIO
AudioSource audiofishJump;
AudioClip[] waterSplashes;
int isfishJumping = 0;

...

void Start ()
{
      audiofishJump = gameObject.AddComponent<AudioSource>();
      waterSplashes = new AudioClip[2];
      waterSplashes[0] = Resources.Load("Sounds/WaterSplash") as AudioClip;
      waterSplashes[1] = Resources.Load("Sounds/WaterSplash2") as AudioClip;
      audiofishJump.clip = waterSplashes[Random.Range(0,2)];
}

...

//Fish Jump Sound
void playfishJumpSound()
{
    //adjust volume based on distance, random pitch
    audiofishJump.volume = 1f - curDistance * 0.015f + 0.2f;
    audiofishJump.pitch = Random.Range(0.8f,1.2f);

    //first check that audio isnt already playing
    if(!audiofishJump.isPlaying)
    {
        isfishJumping = Random.Range(0,4); // % chance to jump
        if(curDistance == 0f) //distance 0 -> splash always
        {
            isfishJumping = 0;
        }
        if(isfishJumping == 0)
        {
            audiofishJump.PlayOneShot(waterSplashes[Random.Range(0,2)]);
        }
    }
}
```

Ohjelmalistaus 13. C#-ohjelmointikielellä toteutettu äänen toistaminen kalan hypätessä.

Kalan hypätessä vedessä loiskahdus on ohjelmoitu kahta ääniklippiä käyttämällä. Aluksi muuttujat alustetaan ja peliobjektille luodaan Audio Source-komponentti. Tämän jälkeen luodaan ääniklipeille oma taulukko, johon klipit ladataan. Kalan hyppäämiseen tehty funktio hoitaa itse äänen toistamisen. Funktiota kutsutaan kalan ollessa vedessä samaan aikaan puhelimen värinän kanssa. Audio Source-komponentissa on mahdollista vaihdella äänenvoimakkuutta ja sävelkorkeutta, joita muunnetaan funktion alussa, jotta ääni ei olisi aina täysin sama. Jos ääniklippi ei jo soi, arvotaan hyppääkö kala, sillä kala ei ole pinnassa joka nykäisyssä, mutta etäisyyden ollessa nollassa loiskahdus kuuluu aina. Arvotun hyppäämisen käydessä toteen toistetaan jokin ääniklippi taulukosta kerran PlayOneShot()-komennolla. (Ohjelmalistaus 13)

3.7 Testaus

Pelin testaus jäi vähemmälle tässä projektissa ajanpuutteen vuoksi. Kehittämisen aikana peliä testattiin itse ja aina sopivin väliajoin annettiin uusin versio kokeiltavaksi tutuille palautteen toivossa. Palautteesta poimittiin parannukset ja korjaukset, jotka pystyttiin toteuttamaan aikataulun mukaan.

Testauksessa käytettiin eniten Samsung Galaxy S3-puhelinta ja satunnaisesti uudempaa korkeampi resoluutioisempaa Samsung Galaxy S4-puhelinta varmistaakseen, että peli toimii ja näkymät ovat oikein. Kehityksen loppuvaiheilla syntyi vielä mahdollisuus testata peliä Samsung Galaxy S7 Edgellä, jossa on vieläkin korkeampi resoluutio, mikä johti vielä fonttien koon hienosäätöön.

4 JULKAISU GOOGLE PLAY-KAUPASSA

4.1 Lisenssit

Ennen julkaistavan sovelluksen kehittämisen aloittamista on syytä tarkistaa ohjelmien lisenssit. Lisenssistä selviää voiko ohjelmaa käyttää kaupallisen sovelluksen tekemiseen vai onko se tarkoitettu vain henkilökohtaiseen käyttöön ja ohjelman opetteluun.

Unity 3D antaa luvan kaupallisten pelien tekemiseen ilmaisella versiollakin, mutta vuosittaiset tulot eivät saa ylittää 100 000 dollaria (Unity Technologies

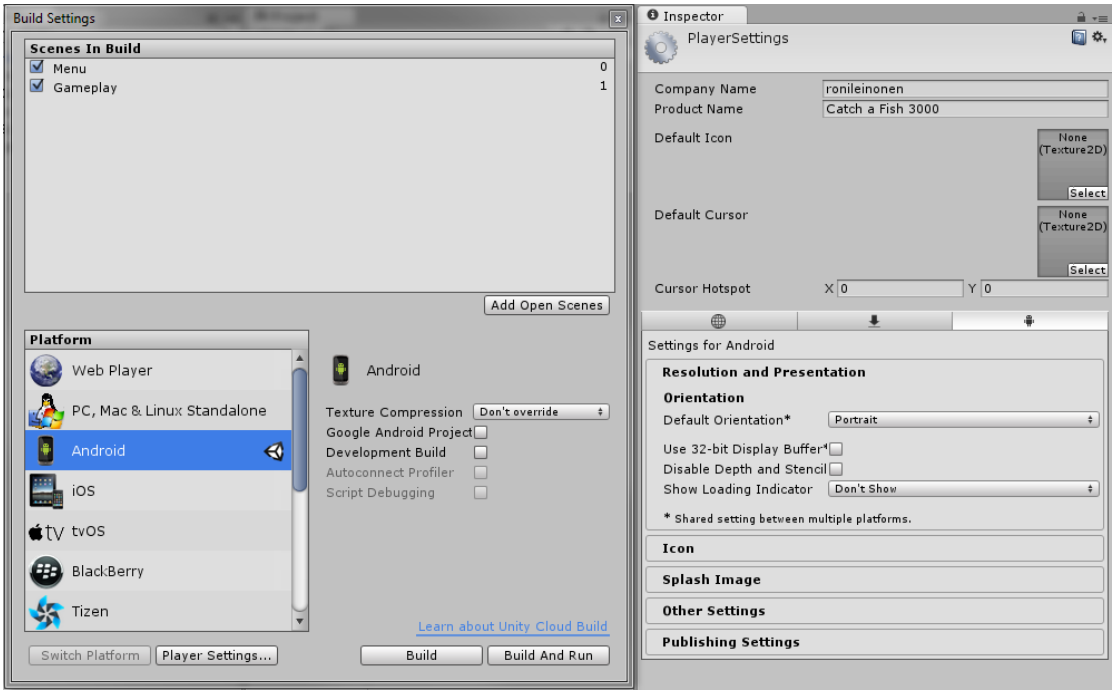
2016q). Sovelluksen äänien muokkaamiseen käytetty Audacity on avoimen lähdekoodin täysin ilmainen ohjelma, joten sillä tehtyjä ääniä saa vapaasti jakaa eteenpäin. Autodeskin 3ds Max opiskelijalisenssi on tarkoitettu ainoastaan ohjelman opetteluun eikä sillä tehtyjä tuotoksia saa käyttää kaupallisessa sovelluksessa.

Sovellusta ei kehitetty kaupalliseksi ja se on saatavilla täysin ilmaiseksi, mutta 3ds Maxilla tehdystä virvelistä oli luovuttava julkaistavassa versiossa. Vaikka sovellus ei olekaan kaupallinen, sen julkaisu ja jakaminen Google Play-kaupassa ei liity ohjelman opetteluun.

Julkaisuversion 3d-mallinnusohjelmaksi valittiin Blender, joka on ilmainen ohjelma ja antaa vapauden kaupalliseen käyttöön ilmaiseksi (Blender Foundation 2016). 3d-mallinnusprosessin vaiheet etenevät samassa järjestyksessä. Ensin luodaan virveli polygoneista, jonka jälkeen tehdään UV map tekstuureja varten. Virvelin laatu kärsi hieman Blenderillä tehtäessä, sillä verratuna 3ds Maxiin Blender on hieman vaikeampi käyttää. Virveli valmistui suhteellisen nopeasti, koska se ei sisällä animaatioita vaan sen osia liikutetaan ohjelmoimalla Unityssa.

4.2 Apk-tiedoston luonti

Sovelluksen asennus Android-mobiililaitteeseen vaatii apk-tiedoston luomisen. Apk on tiedostomuoto tiedostolle, joka asentaa sovelluksen tai pelin Android käyttöjärjestelmän sisältäviin laitteisiin. Apk-tiedosto luodaan Unityn Build Settings-ikkunan kautta.



Kuva 17. Build Settings ja Player Settings.

Alustaksi valitaan Android ja Player Settings-välilehdessä voidaan muokata sovelluksen tietoja. Sovellukselle annetaan nimi, asetetaan pystysuuntaiseksi ja lisätään sovellukselle kuvake. Asetuksista voidaan asettaa myös esimerkiksi minimivaatimukset laitteelle. Apk-tiedosto luodaan Build-painikkeesta. (Kuva 17)

Apk-tiedoston luomiseen tarvitaan Android SDK-pakettia, jota Unity pyytää ensimmäisellä luontiyrityksellä. Android Software Development Kit sisältää työkalut ja komponentit mahdollistaen sovelluksien kehittämisen Android käyttöjärjestelmille.

4.3 Julkaisu

Sovelluksen julkaiseminen Google Play-kaupassa vaatii julkaisijatilin rekisteröimisen. Rekisteröinti edellyttää perustietojen lisäksi 25 dollarin rekisteröintimaksua, joka on kertaluonteinen. Sovelluksen ollessa maksullinen tarvitaan Google Payment Merchant Account, mutta se ei ole välttämätön ilmaisessa sovelluksessa. Rekisteröitymisen jälkeen pystytään kirjautumaan sisään Googlen kehittäjäkonsoliin, jossa hallinnoidaan kaikkia tilillä julkaistuja sovelluksia. Kehittäjäkonsolissa ladataan sovelluksen apk-tiedosto, määritetään sovelluksen tiedot ja lisätään ruudunkaappaukset. Julkaisua painettaessa sovellus ilmestyy Play-kauppaan muutaman tunnin sisällä. (Android developers 2016.)

5 JATKOKEHITYS

Tulevaisuudessa peli pitää julkaista myös muille alustoille Androidin lisäksi. Applen iOS käyttöjärjestelmä on ensimmäisenä suunnitelmissa sen ollessa toiseksi yleisin tällä hetkellä ennen muita alustoja. Lopulta julkaisu yleisimmille alustoille.

Jatkossa itse peliä voisi kehittää lisäämällä siihen enemmän ja erilaisia uistimia. Tällä hetkellä pelissä on vain lusikkauistimia, joista puolet hieman isompia. Pelattavuutta toisi lisää uistinten ominaisuuksien eri variaatiot, kuten toisella uistimella saisi enemmän tiettyä kalalajia, kuin toisella. Peliin tulisi lisätä uistimien lisäksi sisäinen valuutta tai pelaajan etenemistä seuraavat kokemuspisteet. Uistimet olisi pelaajan kannalta mielekkäämpää ostaa itse kaupasta valitsemalla, kuin niiden avautumien tietyssä ennalta määrätyssä järjestyksessä.

Kalojen mittoja voisi hyödyntää antamalla pelaajille tietoa kalalajien alimitoista. Pelaajan ottaessa talteen alimittaisen kalan seuraisi rangaistus, joka voisi olla sakko. Tämä toisi pelaajille tietoa kalastuksessa huomioitavista asioista.

Peli tarvitsee runsaasti lisää kalastuspaikkoja. Erilaisia paikkoja lisäämällä pelaajan mielenkiinto säilyy pidempään. Eri paikkoja voisi olla lampi, joen- ja merenranta sekä kalastaminen olisi hyvä olla mahdollista myös veneessä.

Kelaamista voisi parantaa ohjelmoimalla kelaamisnopeuden vaikuttamaan kalan saantiin merkittävästi. Jos kelaa liian nopeasti, kalan saanti pienenee ja liian hitaasti kelatessa uistin voisi jäädä pohjaan tai kerätä roskaa koukkuun.

Tärkein jatkokehityskohde on kuitenkin pelin saaminen moninpeliksi. Pelaajat voisivat vertailla tilastojaan, katsoa kuka saa eniten kalaa ja esitellä saaliitaan. Moninpeli toisi peliin siihen tarvittavaa koukuttavuutta.

6 YHTEENVETO

Opinnäytetyön tavoitteena oli saada julkaisukelpoinen sovellus Android-puhelimelle. Sovellus vaatisi vielä enemmän testaamista ja kehittämistä, mutta tämänhetkinen versio on jo julkaistavissa. Tavoitteeseen päästiin ja projekti valmistui suunnitelmien mukaan.

Työssä opittiin 3d-mallintamista kahdella eri ohjelmalla. Alun perin tarkoituksena oli käyttää vain yhtä 3d-mallinnusohjelmaa, mutta jälkeenpäin ajateltuna oli hyvä saada kokemusta useammasta ohjelmasta. 3d-mallintaminen Autodeskin maksullisella ammattilaiskäyttöön suunnitellulla 3ds Max ohjelmalla oli mielekkäämpää näistä kahdesta. Blender Foundationin ilmaisen Blender 3d-mallinnusohjelman käyttö oli aluksi hieman haastavaa. Perustoimintojen oppimisen jälkeen mallintaminen sujui paremmin, mutta se oli silti hitaampaa kuin 3ds Maxilla. 3D-mallin animointi herätti myös mielenkiintoa, mutta sitä ei todettu tarpeelliseksi tässä kehityksessä.

Unity 3D -pelimoottoria oli helppo käyttää ja sen käytön oppimiseen ei mennyt aikaa. Ohjelmointia helpotti Unityn kattava dokumentaatio, josta sai tarvittaessa apua helposti. Kokonaisuutena Unity 3D:stä jäi positiivinen kuva sen käyttäjäystävällisyyden vuoksi. Sovelluksen eri ominaisuuksien toteuttaminen syntyi nopeasti ja turhautumisilta vältyttiin.

Sovelluksesta tuli alkuperäisen vision mukainen ja haluttuun lopputulokseen päästiin. Yksinkertainen simulaatiomainen peli ajanviettoon. Hieman jäi harmittamaan monipelin puuttuminen, mutta se on lisättävissä peliin myöhemmin.

LÄHTEET

Android Developers. 2016. Get Started with Publishing. Developer.android.com. Saatavissa: http://developer.android.com/distribute/googleplay/start.html [viitattu 20.4.2016].

Ashvin. 2016. Top 10 popular mobile OS in the world. Mymobotips.com. Saatavissa: http://mymobotips.com/best-top-10-mobile-os-in-the-world/ [viitattu 3.4.2016].

Audacity. 2016. Welcome to Audacity. Audacityteam.org. Saatavissa: http://www.audacityteam.org/ [viitattu 17.3.2016].

Autodesk. 2016. 3ds Max. Autodesk.com. Saatavissa: http://www.autodesk.com/products/3ds-max/overview [viitattu 17.3.2016].

Barraclough, C. & Todd, A. 2016. What is Android and what is an Android phone. Recombu.com. Saatavissa: https://recombu.com/mobile/article/what-is-android-and-what-is-an-android-phone_M12615.html [viitattu 3.4.2016].

Blender Foundation. 2016. Free to Use. Free to Change. Free to Share. Free to Sell Your Work. Blender is FREE SOFTWARE. Blender.org. Saatavissa: https://www.blender.org/about/license/ [viitattu 20.4.2016].

Daily, E. 2014. How to save and load your players progress in Unity. Gamedevelopment.tutsplus.com. Saatavissa: http://gamedevelopment.tutsplus.com/tutorials/how-to-save-and-load-your-players-progress-in-unity--cms-20934 [viitattu 4.4.2016].

Dominguez, A. 2014. Using Spritesheets with Unity3D. Strandedsoft.com. Saatavissa: http://www.strandedsoft.com/using-spritesheets-with-unity3d/ [viitattu 12.4.2016].

Gibson, J. 2014. Introduction to Game Design, Prototyping and Development. Indiana, US: Pearson Education, Inc.

Goldstone, W. 2009. Unity Game Development Essentials. Birmingham, Uk: Packt Publishing Ltd.

Gujarati, P. 2013. What is Accelerometer and how does it work on smartphones. Techulator.com. Saatavissa: http://www.techulator.com/resources/8930-How-does-smart-phone-accelerometer-work.aspx [viitattu 17.3.2016].

Nielsen, J. 2012. Usability 101: Introduction to Usability. Nngroup.com. Saatavissa: https://www.nngroup.com/articles/usability-101-introduction-to-usability/ [viitattu 8.4.2016].

Silverman, D. 2013. 3D Primer for Game Developers: An Overview of 3D Modeling in Games. Gamedevelopment.tutsplus.com. Saatavissa: http://gamedevelopment.tutsplus.com/articles/3d-primer-for-game-developers-an-overview-of-3d-modeling-in-games--gamedev-5704 [viitattu 12.3.2016].

Slick, J. 2014. Anatomy of a 3d Model. 3d.about.com. Saatavissa: http://3d.about.com/od/3d-101-The-Basics/a/Anatomy-Of-A-3d-Model.htm [viitattu 12.3.2016].

Slick, J. 2016. Surfacing 101 – Creating a UV Layout. 3d.about.com. Saatavissa: http://3d.about.com/od/3d-101-The-Basics/a/Surfacing-101-Creating-A-UV-Layout.htm [viitattu 12.3.2016].

StatCounter. 2016. Top 8 Mobile Operating Systems on Jan 2016. Statcounter.com. Saatavissa: http://gs.statcounter.com/#mobile_os-ww-monthly-201601-201601-bar [viitattu 29.1.2016].

Unity Technologies. 2016a. Get Unity. Unity3d.com Saatavissa: http://unity3d.com/get-unity [viitattu 17.3.2016].

Unity Technologies. 2016b. Multiplatform. Unity3d.com Saatavissa: http://unity3d.com/unity/multiplatform/ [viitattu 17.3.2016].

Unity Technologies. 2016c. Monodevelop. Docs.unity3d.com. Saatavissa: http://docs.unity3d.com/Manual/MonoDevelop.html [viitattu 17.3.2016].

Unity Technologies. 2016d. Unity Manual. Docs.unity3d.com. Saatavissa: http://docs.unity3d.com/Manual/index.html [viitattu 17.3.2016].

Unity Technologies. 2016e. Creating and using scripts. Docs.unity3d.com. Saatavissa: http://docs.unity3d.com/Manual/CreatingAndUsingScripts.html [viitattu 18.3.2016].

Unity Technologies. 2016f. Mobile input. Docs.unity3d.com. Saatavissa: http://docs.unity3d.com/Manual/MobileInput.html [viitattu 20.3.2016].

Unity Technologies. 2016g. Object Instantiate. Docs.unity3d.com. Saatavissa: http://docs.unity3d.com/ScriptReference/Object.Instantiate.html [viitattu 24.3.2016].

Unity Technologies. 2016h. Transform.Translate. Docs.unity3d.com. Saatavissa: http://docs.unity3d.com/ScriptReference/Transform.Translate.html [viitattu 18.3.2016].

Unity Technologies. 2016i. Camera.WorldToScreenPoint. Docs.unity3d.com. Saatavissa: http://docs.unity3d.com/ScriptReference/Camera.WorldToScreenPoint.html [viitattu 20.3.2016].

Unity Technologies. 2016j. Random.Range. Docs.unity3d.com. Saatavissa: http://docs.unity3d.com/ScriptReference/Random.Range.html [viitattu 25.3.2016].

Unity Technologies. 2016k. Handheld.Vibrate. Docs.unity3d.com. Saatavissa: http://docs.unity3d.com/ScriptReference/Handheld.Vibrate.html [viitattu 31.3.2016].

Unity Technologies. 2016l. Object.DontDestroyOnLoad. Docs.unity3d.com. Saatavissa: http://docs.unity3d.com/ScriptReference/Object.DontDestroyOnLoad.html [viitattu 4.4.2016].

Unity Technologies. 2016m. CreatingScenes. Docs.unity3d.com. Saatavissa: http://docs.unity3d.com/Manual/CreatingScenes.html [viitattu 8.4.2016].

Unity Technologies. 2016n. Application.LoadLevel. Docs.unity3d.com. Saatavissa: http://docs.unity3d.com/ScriptReference/Application.LoadLevel.html [viitattu 8.4.2016].

Unity Technologies. 2016o. SpriteEditor. Docs.unity3d.com. Saatavissa: http://docs.unity3d.com/Manual/SpriteEditor.html [viitattu 10.4.2016].

Unity Technologies. 2016p. AudioListener. Docs.unity3d.com. Saatavissa: http://docs.unity3d.com/Manual/class-AudioListener.html [viitattu 6.4.2016].

Unity Technologies. 2016q. Unity pro and Unity personal software license agreement 5.X. Unity3d.com. Saatavissa: https://unity3d.com/legal/eula [viitattu 18.4.2016].