OULUN AMMATTIKORKEAKOULU

Tommi Vanhala

PELI- JA SOVELLUSKEHITYS ANDROID-KÄYTTÖJÄRJESTELMÄLLE

PELI- JA SOVELLUSKEHITYS ANDROID-KÄYTTÖJÄRJESTEL-MÄLLE

Tommi Vanhala
Opinnäytetyö
Syksy 2016
Tietotekniikan koulutusohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietotekniikan koulutusohjelma, ohjelmistokehityksen suuntautumisvaihtoehto

Tekijä(t): Tommi Vanhala
Opinnäytetyön nimi: Peli- ja sovelluskehitys Android-käyttöjärjestelmälle
Työn ohjaaja(t): Veijo Väisänen, Jaakko Kaski
Työn valmistumislukukausi ja -vuosi: Syksy 2016
Sivumäärä: 10 + 2 liitettä

Tämä opinnäytetyö toteutettiin kahdessa eri osassa, joten opinnäytetyö eroaa toteutukseltaan perinteisestä toteutuksesta. Ensimmäisessä osaopinnäytetyössä tutustuttiin pelien tekemiseen Android-käyttöjärjestelmälle Unity-pelinkehitysohjelmistoa käyttäen. Työssä tutustuttiin myös yksinkertaisten 3D-mallien tekemiseen Blender-ohjelmistoa käyttäen. Pelissä käytettävien skriptien ohjelmointiin käytettiin C#-ohjelmointikieltä.

Opinnäytetyön toisessa osassa suunniteltiin ja toteutettiin Android-käyttöjärjestelmälle urheilijoille suunnattu harjoituspäiväkirjasovellus. Toisen osan tilaajana toimi urheiluseura TKF Finland Team. Android-sovelluksen toiminnollisuus kehitettiin käyttäen Java-ohjelmointikieltä ja ulkoasut toteutettiin XML-ohjelmointikielellä. Työssä suunniteltiin ja toteutettiin sovelluksen käyttämä palvelinohjelmisto PHP-ohjelmointikielellä. Palvelinohjelmiston toteutuksen apuna käytettiin Slim-viitekehystä.

Kokonaisuudessaan opinnäytetyön aikana Android-käyttöjärjestelmälle ohjelmoiminen tuli tutuksi niin pelinkehityksessä kuin normaalin sovelluksen kehityksessä. Ensimmäisessä osassa toteutettu peli saatiin kehitettyä demovaiheeseen. Pelin tekeminen opetti pelikehitystä Unity-kehitysympäristössä. Peliä ei kehitetä tällä hetkellä. Toisessa osassa toteutettu ohjelma on jatkuvassa kehityksessä ja tällä hetkellä testausvaiheessa.

Asiasanat: Android, Java, PHP, Unity, HTTP, Ohjelmointi

ABSTRACT

Oulu University of Applied Sciences
Information Technology, Software development

Author(s): Tommi Vanhala
Title of thesis: Development of game and application for Android operating system
Supervisor(s): Veijo Väisänen, Jaakko Kaski
Term and year when the thesis was submitted: Autumn 2016
Pages: 10 + 2 appendices

This thesis was done in two parts. First part consists of game development for Android operating system using Unity game engine. Scripts used in the game were made with C# programming language. The first part also handles how to model simple 3D models using software called Blender.

The second part consists of application development for Android operating system. Application made during second part is train diary aimed for athletes. The second part was made for sports club named TKF Finland Team. The frontend for the application was made with Java and XML programming languages. The backend was made with PHP programming language with the help of Slim framework.

Software development for the Android operating system became familiar in both game development and application development during the thesis. The game made in the first part was a demo that taught techniques of Unity game programming. The game is not in development anymore. The application made in the second part is currently in development and testing is in progress.

Keywords: Android, Java, PHP, Unity, HTTP, Programming

SISÄLLYS

1 JOHDANTO

Opinnäytetyön eri osien työstäminen aloitettiin toisen lukuvuoden kevätlukukaudella, jolloin kirjoitettiin ensimmäinen osa. Alun perin suunnitelmana oli työstää kolme pienempää työtä ja koostaa näistä yksi iso kokonaisuus. Tämä työ on kuitenkin tehty kahdessa osassa, joista ensimmäinen on viiden opintopisteen ja toinen kymmenen opintopisteen. Opinnäytetyön osittaminen aloitettiin Oulun ammattikorkeakoulun tietotekniikan koulutusohjelmassa vuonna 2014. Toisessa osassa työstetyn sovelluksen kehittäminen aloitettiin kesällä 2016 ja dokumentaatio laadittiin syksyn 2016 aikana.

Työn ensimmäinen osa on pintaraapaisu siitä, miten Unity-pelinkehitysohjelmistolla päästään pelinkehityksessä alkuun. Osassa työstetty peli suunnattiin Android-käyttöjärjestelmälle. Pelissä käytetyt mallit tehtiin Blender-mallinnusohjelmistolla. Ensimmäisen osan aihe oli kirjoittajan itse keksimä ja tavoitteena oli saada dokumentaatiosta informatiivinen aloitusopas.

Toinen osa tehtiin urheiluseura TKF Finland Teamille. Työn osassa oli tarkoituksena saada aikaan sovellus Android-käyttöjärjestelmälle. Myös palvelinohjelmiston kehittäminen kuului toiseen osaan. Työn aihe tuli suoraan urheiluseuran yhteyshenkilöltä.

2 ENSIMMÄISEN OSAN ESITTELY

Opinnäytetyön ensimmäinen osa (liite 1) tehtiin toisen opintovuoden keväällä. Osan aiheena oli 3D-pelin tekeminen Android-käyttöjärjestelmälle Unityllä. Pelissä käytettävät skriptit toteutettiin C#-ohjelmointikielellä ja 3D-mallinnukseen käytettiin Blender-ohjelmistoa.

Ensimmäisessä osassa toteutetun pelin tarkoituksena oli perehdyttää pelinkehityksen aloitukseen Android-käyttöjärjestelmälle Unity-pelinkehityssovellusta käyttäen. Työn aikana saatiin aikaan toimiva demo, joka pyöri oikealla Android-laitteella. Peliä voitaisiin jatkokehittää, mutta tällä hetkellä sen kehittäminen on lopetettu.

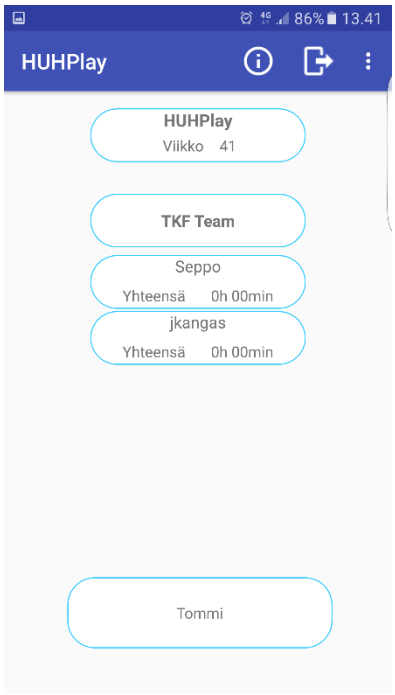
Ensimmäisestä osasta saatiin aikaan dokumentti, jossa kerrotaan, miten pelinkehitys voidaan aloittaa Unity-ohjelmistolla sekä, miten tehdään yksinkertaisia 3D-malleja Blender-ohjelmistolla. Työn aikana Android-käyttöjärjestelmä, Unity-pelinkehitysohjelmisto ja Blender-mallinnustyöohjelmisto tulivat tutuiksi.

3 TOISEN OSAN ESITTELY

Opinnäytetyön toisen osan (liite 2) sovelluksen kehittäminen aloitettiin kolmannen ja neljännen vuoden välisenä kesänä. Toisen osan tilaajana toimi urheiluseura TKF Finland Team. Työn dokumentti tehtiin neljännen vuoden syksyn aikana. Aiheena oli HUHPlay-sovelluksen kehittäminen Android-käyttöjärjestelmälle. Sovelluksen toiminnollisuuden kehittämiseen käytettiin Java- ja ulkoasun kehittämiseen XML-ohjelmointikieliä. Palvelinohjelmiston kehittämiseen käytettiin PHP-ohjelmointikieltä ja sille asennettua Slim-viitekehystä.

Toisessa osassa toteutettu sovellus on urheilijoille suunnattu harjoituspäiväkirja. Sovelluksen ideana on antaa urheilijoille helppo ja mukana kulkeva tapa merkata sekä seurata omia ja muiden harjoituksia. Sovellus on jatkuvassa kehityksessä ja tällä hetkellä testausvaiheessa.

Sovellukseen saatiin toteutettua kaikki suunnitellut ominaisuudet. Näitä olivat käyttäjäksi rekisteröityminen, sisäänkirjautuminen, harjoituksien kirjaaminen ja niiden vertailu. Kuten kuvasta 1 nähdään, niin käyttöliittymä haluttiin pitää mahdollisimman yksinkertaisena ja keskittyä toiminnollisuuden kehittämiseen.



KUVA 1. HUHPlayn päänäkymä

Työn aikana perehdyttiin PHP-ohjelmointikieleen ja HTTP-pyyntöjen reitityk-seen. Työtä oli erittäin mielenkiintoista toteuttaa, koska työ tulisi oikeasti urheili-joiden käyttöön. Haastetta sovelluksen toteutukseen toi se, että sovellus piti suunnitella siten, että urheiluseurat voisivat kaupata sovelluksen käyttökoodeja ja näin kerätä rahaa.

4 YHTEENVETO

Yhden ison kokonaisuuden sijaan, tämä opinnäytetyö toteutettiin kahdessa eri osassa. Osat eroavat paljon toisistaan, sillä ensimmäinen osa on enemmän pelikehitystä ja toinen osa sovelluskehitystä. Kummankin osan aikana toteutettu tuotos on kuitenkin tehty Android-käyttöjärjestelmälle, joten yhtäläisyyksiäkin löytyy.

Opinnäytetyön osat on tehty koulutuksen eri vaiheissa. Ensimmäistä osaa alettiin työstää toisen opiskeluvuoden kevätlukukaudella ja toista osaa kolmannen ja neljännen vuoden välisenä kesänä. Vertaillessa osia keskenään on nähtävissä selkeää kehitystä niin kirjoittamisessa kuin ohjelmoinnissakin. Työn osat ovat erillisiä kokonaisuuksia, koska toinen osa toteutettiin tilaustyönä urheiluseuralle.

Toisessa osassa tehdyn sovelluksen kehittäminen vaati paljon koulussa opittuja ohjelmointitaitoja sekä paljon uuden tiedon opettelemista. Sovellusta työstäessä kuitenkin huomasi, kuinka paljon vuoden aikana oli oppinut ohjelmoinnista ja eri arkkitehtuureista. Myös koulutukseen kuuluneet yrityslähtöiset tuotekehitysprojektit olivat antaneet paljon uudenlaista taitoja konkreettisten projektien merkeissä. Molemmista osista sai kuitenkin uutta tietoa eri alueilta ja tehdyistä raporteista tuli ehjiä kokonaisuuksia.

Joillekin opiskelijoille osaopinnäytetyön suorittaminen sopii mainiosti, sillä kerralla tehtävät kokonaisuudet eivät ole niin suuria ja pienien osien työstäminen on paljon ohjatumpaa. Itse henkilökohtaisesti koen, että yksi isompi kokonaisuus olisi voinut olla helpompaa ja luontevampaa. Isompaa kokonaisuutta tehdessä vaarana kuitenkin olisi voinut olla opinnäytetyön valmistumisen viivästyminen.

OULUN AMMATTIKORKEAKOULU

Tommi Vanhala

3D-PELIEN TEKEMINEN ANDROID-KÄYTTÖJÄRJESTELMÄLLE UNITYLLÄ

3D-PELIEN TEKEMINEN ANDROID-KÄYTTÖJÄRJESTELMÄLLE UNITYLLÄ

Tommi Vanhala
Opinnäytetyö, osa 1
Kevät 2015
Tietotekniikan koulutusohjelma
Oulun ammattikorkeakoulu

SISÄLLYS

SANASTO

2D	Two dimensional eli kaksiulotteinen.
3D	Three dimensional eli kolmiulotteinen.
APK	Androidin ohjelmapakettitiedostomuoto.
Blend	Blenderillä luotujen projektien tiedostomuoto
Face	Blenderissä objektin sivu.
Float	Liukuluku (esimerkiksi 1,005)
Foreach	Lause, jonka avulla käydään määrätyn tyyppiset objektit läpi annetusta taulukosta tai listasta.
If	Lause, jonka avulla tarkastellaan onko jokin väittämä tosi.
Objekti	Esine tai asia, joka kuuluu pelimaailmaan, esimerkiksi seinä tai valo.
RGB	Eri värejä muodostaessa, RGB-luku ilmaisee punaisen, vihreän ja sinisen värin suhteellisen määrän.
Skripti	Sarja ohjeita tietokoneelle siitä, miten tietokoneen tulisi käyttäytyä.
Skene	Unityssä taso (scene), joka pitää sisällään kaikki objektit.
Törmäytin	(Collider) Käytetään tunnistamaan osumia kahden objektin välillä.
Vector2	Kaksiulotteinen vektori (x, y -tasoilla).
Vector3	Kolmiulotteinen vektori (x, y, z -tasoilla).

1 JOHDANTO

Työn tarkoituksena on tutustua Unity-pelinkehitysohjelmistoon, Blender-3D-mallinnusohjelmistoon ja Android-käyttöjärjestelmään, sekä siihen miten kaikki kolme saadaan yhdistettyä pelin muodossa. Aluksi kerrotaan, mitä käytettävät sovellukset ovat ja mitä ne tarjoavat käyttäjille, sekä tutustutaan Android-käyttöjärjestelmän historiaan. Tämän jälkeen tutustutaan, miten Unity-ohjelmistolla saadaan käännettyä Android-käyttöjärjestelmälle peli. Teorian jälkeen katsotaan esimerkin avulla, miten Unity toimii. Työn lopuksi pohditaan, miten Unityä voidaan hyödyntää tulevaisuudessa, sekä pohditaan opittuja asioita.

2 OHJELMISTOT

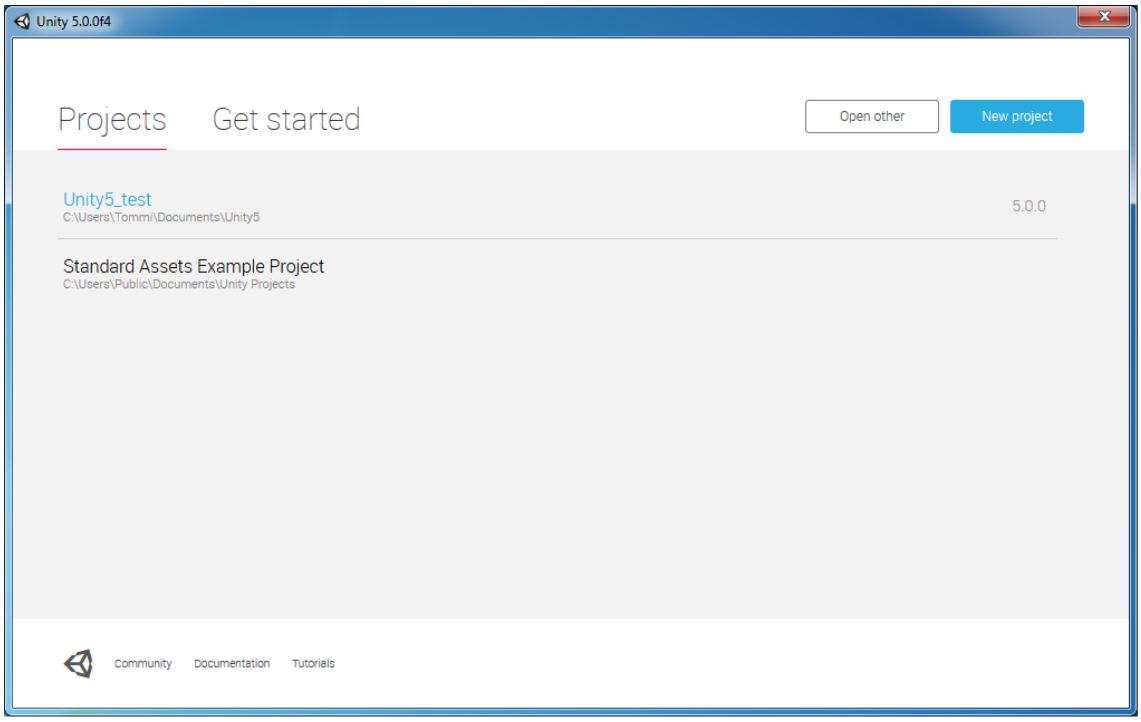
Pelinkehitykseen on kehitelty monenlaisia ohjelmia helpottamaan kehittäjien työtä. On valmiita pelinkehitysympäristöjä, 3D-mallinnusohjelmistoja ja todella kattavia kuvanmuokkausohjelmistoja. Pelinkehitysympäristöt tuovat pelikehittäjille yksinkertaisempaa lähestymistapaa kehitykseen, sillä yleensä ohjelmistoissa on pelimoottori jo valmiina. Ilman ohjelmistoja pelimoottorin kehittäminen olisi työlästä ja rahaa vievää.

Teknologian kehittyessä markkinoille on tullut peleille uusia alustoja, joista suurimmassa osassa ovat älypuhelimet. Älypuhelimet ovat yleistyneet räjähdysmäisesti viime vuosina. Pelkästään Yhdysvalloissa vuonna 2013 oli 143 miljoonaa älypuhelinta. (1.)

2.1 Unity-pelinkehitysympäristö

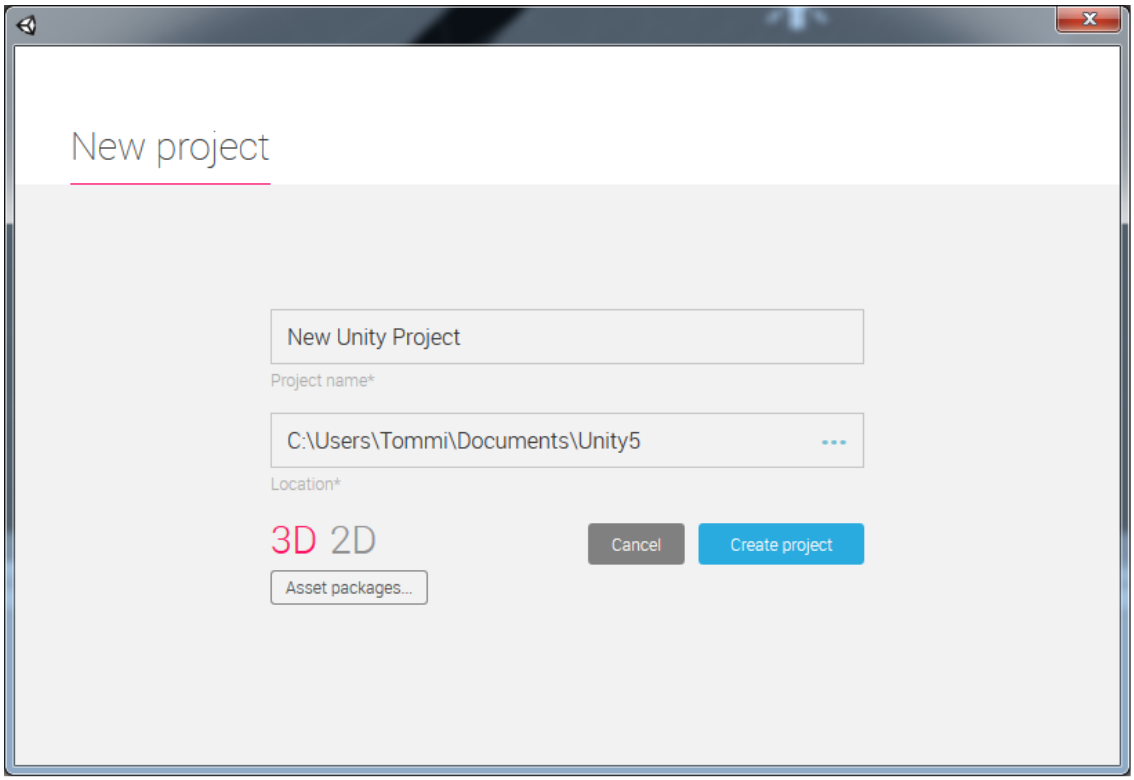
Unity-pelinkehitysympäristön kehittäminen alkoi jo vuonna 2001. Ensimmäinen versio Unity1 julkaistiin vuonna 2005 Applen WWDC-konferenssissa. Nykyisin Unity kulkee versionumerolla 5.0.0, joka julkaistiin kolmas maaliskuuta 2015. Unity tukee 21:tä eri julkaisualustaa muun muassa Androidia, iOSia ja Windowsia. (2.)

Kuvassa 1 nähdään ruutu, joka aukeaa aukaistessa Unity-ohjelmiston. Uusi projekti luodaan oikeassa yläkulmassa olevasta New project -näppäimestä. Jos halutaan avata jo valmis projekti, valitaan Open other tai näkyvästä listasta jokin projekti.



KUVA 1. Unity-pelinkehitysympäristön aloitusruutu

Kuten kuvasta 2 nähdään, uutta projektia luodessa Unity haluaa tietää projektille nimen, tallennussijainnin ja sen, käytetäänkö 3D- vai 2D-maailmaa, sekä halutessaan voi valita valmiita kirjastoja peliä varten. Työhön ei tarvita valmiita kirjastoja, valitsemme 3D-maailman, annamme projektille nimeksi ”Opinnäytetyö” ja sijaintina käytämme oletussijaintia.



KUVA 2. Uuden projektin luominen

2.2 Blender-3D-mallinnusohjelmisto

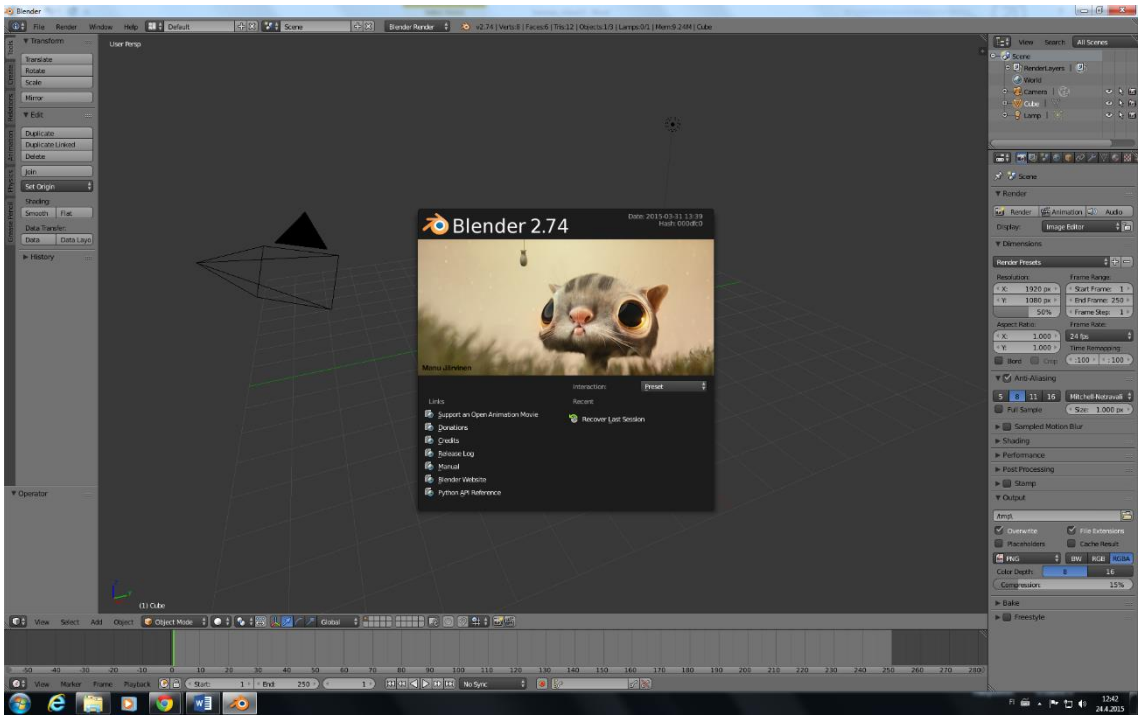
Blender on Blender Foundationin kehittämä ilmainen ja avoimen lähdekoodin 3D-mallinnusohjelmisto (3). Blenderillä voidaan tehdä myös paljon muutakin kuin pelkkää 3D-grafiikkaa, kuten esimerkiksi animaatioita ja jopa pelejä. Myös videoiden editointi onnistuu Blenderillä. (4.) Työssä tutustutaan kuitenkin vain yksinkertaisten 3D-esineiden mallinnukseen Blender-työkalulla. Kirjoitushetkellä Blenderin uusin versionumero on 2.74, joka on julkistettu maaliskuun lopussa vuonna 2015. Jotta Blenderin käyttö olisi nopeampaa ja tehokkaampaa on siihen asetettu eri näppäimille erilaisia oikopolkuja eri asioihin. Taulukossa 1 esitellään muutama eniten käytetty.

TAULUKKO 1. Blenderin pikanäppäimet

Näppäin / näppäinyhdistelmä	**Tarkoitus**
S	Skaalaa objektia
R	Käännä objektia x-, y- ja z-tasoissa
Shift + A	Luo uusi objekti
A	Valitse kaikki / poista valinta
X	Poista
Numpad-numerot	Aseta kamera ennalta määrättyyn pisteeseen
E	Venytä valittua kulmaa tai sivua

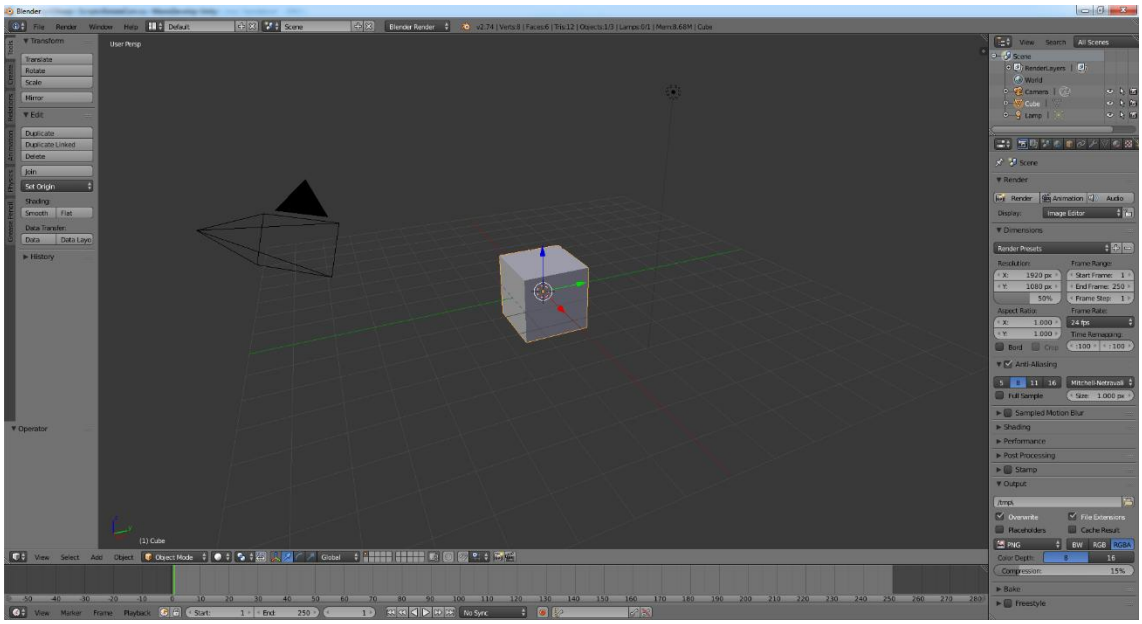
Huomioitavaa Blenderin käytössä on se, että oletusasetuksilla valinta tapahtuu hiiren oikealla näppäimellä ja hiiren vasen näppäin liikuttaa 3D-kursoria. Rullaa painettaessa ja hiirtä liikuttaessa voidaan käännellä kameraa ja rullaamalla voidaan tarkentaa ja loitontaa kameraa.

Kuten kuvassa 3 nähdään ruutu, joka aukeaa joka kerta kun Blender aukaistaan. Aloitusruudusta voidaan valita viimeksi auki ollut blend-tiedosto, palauttaa viime istunnon tai aukaista linkkejä sivustoihin, joissa kerrotaan Blenderistä tai sen käytöstä. Jos halutaan aloittaa tyhjältä pöydältä, klikataan aloitusruudun takana näkyvää ristikkoa.



KUVA 3. Blenderin aloitusruutu

Kun Blenderillä luodaan uusi projekti, niin oletuksena näkymään luodaan kuutio ja kamera. Kuvassa 4 nähdään kyseinen oletusasetelma.



KUVA 4. Blenderin oletusasetelma

2.3 Android-käyttöjärjestelmä

Android-käyttöjärjestelmä tuli markkinoille, kun Yhdysvalloissa julkistettiin T-Mobile G1 –puhelin. (5.) Androidin viimeisin ohjelmistoversio on 5.1 koodinimeltään ”Lollipop”. Jokaiselle merkittävälle ohjelmistopäivitykselle, alkaen versiosta 1.5, on annettu koodinimi, jonkin jälkiruuan mukaan aakkosjärjestyksessä. Koodinimet näkyvät taulukossa 2.

TAULUKKO 2. Android-versiot ja niiden koodinimet (6)

Versionumero(t)	**Koodinimi**
1.0	Ei koodinimeä
1.1	Ei koodinimeä
1.5	Cupcake
1.6	Donut
2.0, 2.0.1, 2.1	Eclair
2.2.x	Froyo
2.3 - 2.3.7	Gingerbread
3.0 – 3.2.x	Honeycomb
4.0.1 – 4.0.4	Ice Cream Sandwich
4.1.x – 4.3.x	Jelly Bean
4.4 – 4.4.4	KitKat
5.0 – 5.1	Lollipop

Ennen jokaista päivitystä, versiosta 1.5 lähtien, Google on laittanut Android-pääkonttorin eteen koodinimeä kuvaavan patsaan, jotta ihmiset ovat voineet arvailla tulevan version koodinimeä.

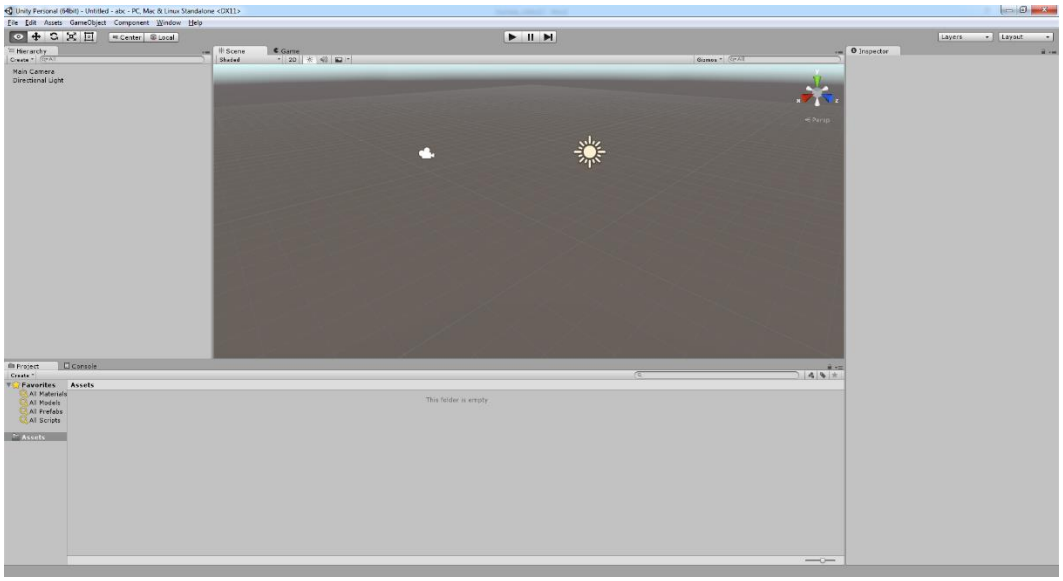
3 PELI

Työssä tutustutaan edellä mainittuihin sovelluksiin käytännössä toteuttamalla ohjelmistoja hyödyntäen yksinkertainen 3D-peli. Pelinkehitysympäristönä on Unity, yksinkertaisten 3D-mallinnusten tekemiseen käytetään Blenderiä ja peli suoritetaan Android-käyttöjärjestelmäpohjaisella älypuhelimella.

Pelin tarkoituksena on kerätä kaikki tason tähdet ja tämän jälkeen siirtyä maali-alueelle. Pelissä on vain yksi taso, mutta uusien tasojen luominen jatkokehitystä ajatellen olisi yksinkertaista. Pelaajan liikuttaminen tapahtuu pyyhkäisemällä sormella eteenpäin, jolloin pelaaja liikkuu eteenpäin. Kääntyminen tapahtuu pyyhkäisemällä joko oikealle tai vasemmalle, riippuen siitä mihin suuntaan halutaan kääntyä. Työssä käydään myös läpi, miten tasojen lisääminen käytännössä tapahtuisi.

3.1 Teoria

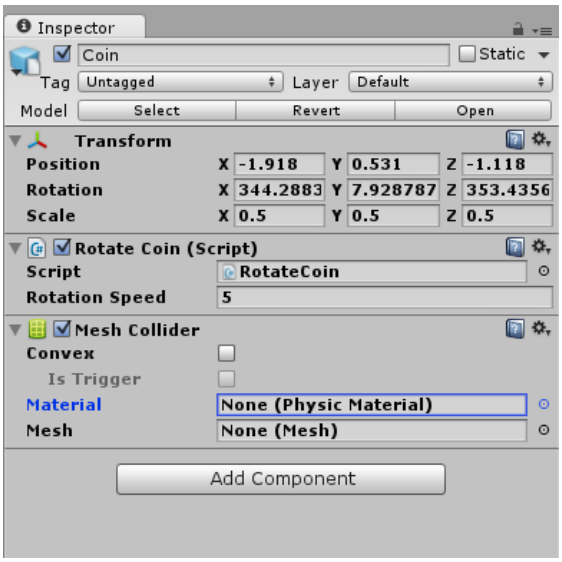
Kun tehdään Unityllä uusi projekti, avautuu näkymä oletusskenestä. Oletusskenessä on valmiiksi asetettu pääkamera, jonka kautta pelaaja näkee pelimaailman, sekä suuntavalo, jonka avulla pelialue valaistaan. Työssä ei käytetä pelaajalle minkäänlaista hahmoa, vaan hahmona toimii pelkästään pääkamera, jota liikutellaan. Kuvassa 5 nähdään millainen Unityn oletusprojektin asetelma on. Kuvassa 5 nähdään myös Unityn muokkausnäkymä. Keskellä nähdään pelimaailma, vasemmassa laidassa sijaitsee objekti hierarkia, alareunassa on projektiselain ja oikeassa laidassa on objektitietoruutu.



KUVA 5. Unityn oletusprojekti

Objekti hierarkiassa nähdään kaikki skenessä olevat objektit tai objektiryhmät. Hierarkiasta voidaan myös valita haluttu objekti aktiiviseksi klikkaamalla hiiren vasemmalla näppäimellä. Kun objektia klikataan kahdesta hiiren vasemmalla näppäimellä, kohdennetaan pelimaailman kamera kyseiseen objektiin. Kun hiiren oikeaa näppäintä painetaan hierarkian sisällä, voidaan poistaa, kopioida, monistaa tai uudelleen nimetä valmiita objekteja tai luoda täysin uusia objekteja.

Kun jokin objekti on valittuna joko hierarkiasta tai pelinäkymästä, avautuu oikeaan reunaan sen tiedot. Tietoruudusta objekteille voidaan luoda uusia komponentteja esimerkiksi fysikaalisia ominaisuuksia tai skriptejä. Kuvassa 6 on esimerkki mitä tietoja objekteista kerrotaan tietoruudussa. Esimerkistä nähdään, että kolikolle on annettu objektin muotoinen törmäytin, skripti RotateCoin sekä kolikon paikka, kulma ja skaalauskerroin.



KUVA 6. Kolikko-objektin tiedot

Jokaiselle objektille voidaan luoda myös tageja, joiden avulla skripteissä on helppo kutsua haluttua objektiryhmää.

Alareunasta löytyy projektiselain, joka osoittaa projektin sijainnin Assets-hakemistoon. Projektiselaimesta löytyy kaikki 3D-mallit, skriptit, skenet ja kuvat mitä Assets-hakemiston alta löytyy. Selaimen avulla uusien objektien luominen on yksinkertaista, tarvitsee vain raahata haluttu 3D-malli pelinäkymään, jolloin siitä luodaan peliobjekti.

3.2 Päävalikko

Yleensä peleissä on jonkin näköinen päävalikko, josta voidaan aloittaa peli, poistua pelistä tai katsoa tekijälistaus. Tässä työssä tehdään hyvin yksinkertainen päävalikko, josta voidaan vain aloittaa peli ja poistua pelistä. Aloitetaan luomalla uusi skene valitsemalla File->New Scene tai pikanäppäimellä Ctrl + N. Kun uusi skene on luotu, vaihdetaan 2D-katselutilaan, jotta päävalikon hahmottaminen olisi helpompaa. Katselutilan vaihtaminen onnistuu painamalla 2D-näppäintä peli-ikkunan yläpuolelta.

Tilan vaihtamisen jälkeen luodaan kaksi uutta painiketta valitsemalla ylävalikosta GameObject->UI->Button. Ensimmäisen painikkeen luomisen aikana,

Unity luo uuden kanvaasin, johon valikon elementit sijoitetaan. Kanvaasi toimii ikään kuin mallinäyttönä. Jokaiselle luodulle painikkeelle, Unity luo painikkeen alaobjektiksi teksti-objektin, jota muokkaamalla voidaan muokata painikkeessa näkyvää tekstiä. Klikataan teksti aktiiviseksi objektihierarkiassa ja objektitietoruudussa voidaan vaihtaa teksti halutuksi, tässä tapauksessa Start ja Quit. Nyt kun painikkeet ovat valmiina, pitää niille luoda omat skriptit, jotka kertovat mitä tehdään painiketta klikattaessa. Aloitetaan toiminnallisuuden toteuttaminen luomalla tyhjä peliobjekti, josta tulee painikkeiden manageri. Tehdään edellä luodulle peliobjektille uusi C#-skripti, Add Component->New Script.

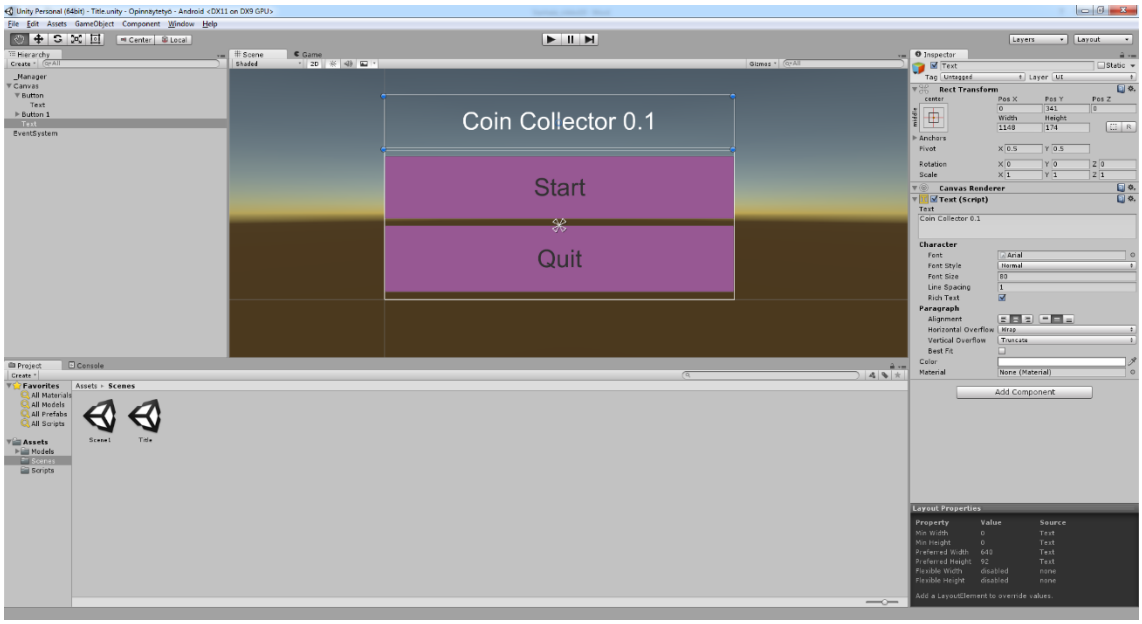
Tälle skriptille emme tarvitse Start tai Update-metodeja, joten ne voidaan poistaa. Tilalle tehdään kaksi uutta julkista metodia, joiden palautusarvoksi asetetaan void. Nämä voidaan nimetä miten haluaa, mutta työssä käytetään toimintaa kuvaavia nimiä Quit ja ChangeToScene. Molempiin metodeihin tarvitaan vain yksi rivi koodia, ChangeToScene-metodiin lisätään koodi, jonka avulla ladataan toinen skene ja Quit-metodiin lisätään pelin sammuttamiseen käytetty koodi. Kuvassa 7 nähdään valmiit metodit.

```
1  using UnityEngine;
2  using System.Collections;
3
4  public class ChangeScene : MonoBehaviour {
5
6      public void ChangeToScene () {
7          Application.LoadLevel ("Scene1");
8      }
9
10     public void Quit(){
11         Application.Quit ();
12     }
13
14 }
15
```

KUVA 7. Painikkeiden toiminnallisuuden toteutus

Skriptin teon jälkeen, liitetään painikkeiden painallustapahtuma juuri tehtyihin metodeihin. Valitaan objektihierarkiasta aktiiviseksi haluavamme painike, esimerkiksi Start-painike. Objektitietoruutuun tulee näkyviin painikkeen tiedot ja

alareunassa näemme On Click () -listauksen. Valitaan listauksesta +-näppäin ja raahataan aiemmin luotu manageriobjekti kohtaan, jossa lukee None (Object). Tämän jälkeen valitaan oikeanreunimmaisesta valikosta ChangeScene->ChangeToScene. Kun valinta on tehty, pitäisi painikkeen toteuttaa ChangeToScene-metodin toiminnallisuus aina painiketta klikattaessa. Quit-painikkeelle tehdään samoin, mutta oikeanreunimmaisesta valikosta valitaan vain ChangeToScene->Quit. Metodien liittämisen jälkeen voidaan muokata painikkeiden koot mieluisaksi klikkaamalla ne aktiiviseksi peliruudussa ja raahaamalla kulmissa olevista sinisistä palloista. Kuvassa 8 nähdään millaiselta esimerkki päävalikko näyttää Unityn sisällä.



KUVA 8. Päävalikko Unityn sisällä

3.3 Skriptit

Pelejä tehdessä itse ohjelmointi tapahtuu skriptien avulla, jotka voidaan kirjoittaa kolmella eri ohjelmointikielellä, C#, JavaScript ja Boo. Työssä käytetään C#-ohjelmointikieltä.

Jotta pelin kerättävät objektit, kolikot, saadaan pyörimään, tarvitaan siihen yksinkertainen skripti. Aluksi pitää luoda malli objektille Blenderissä ja exportata

se projektin Assests-hakemistoon. Tämän jälkeen mennään Unityn muokkaimeen ja raahataan objekti peliin. Kun objekti on liitetty peliin, valitaan se klikkaamalla sitä hiiren vasemmalla näppäimellä. Tämän jälkeen oikeaan reunaan tulee näkyviin objektin tiedot sen paikasta, koosta ja skaalauksesta pelimaailmaan nähden. Painetaan Add Component, valitaan New Script ja syötetään skriptin nimi ja käytettävä ohjelmointikieli, tässä tapauksessa nimi on RotateCoin ja ohjelmointikielenä on CSharp. Kun skripti on luotu, se menee automaattisesti projektin Assets-kansioon, josta se aukaistaan koodimuokkaimeen kaksoisklikkaamalla.

Luodaan pelaaja käyttämällä Unityssä jo valmiina olevaa rakennetta Character Controller. Ensiksi lisätään tyhjä peliobjekti klikkaamalla hiiren oikealla näppäimellä objektihierarkiasta ja valitsemalla Create Empty. Uudelleen nimetään luotu peliobjekti kuvailemaan tämän tarkoitusta, tässä tapauksessa Character, klikkaamalla hiiren oikealla näppäimellä juuri luotua objektia ja valitsemalla Rename. Tämän jälkeen valitaan yläpalkista Component->Physics->Character Controller. Character Controllerin luonnin jälkeen siirretään juuri tehty pelaaja halutulle paikalle. Koska pelaaja näkee pelimaailman pääkameran kautta, tulee se raahata samaan kohtaan mihin edellä luotu pelaaja siirrettiin. Lopuksi raahataan objektihierarkiaa käyttäen Main Camera Characterin päälle, jotta ne sitoutuvat yhteen. Täten Main Camerasta tulee Characterin alaobjekti.

Oletusasetuksin Unity aukaisee skriptit MonoDevelop-Unity nimisellä ohjelmistolla. Unityllä luodussa skriptissä on oletusrakenne, jossa on valmiiksi esitelty luokka ja luokan kaksi metodia Start ja Update. Start- ja Update-metodit ovat Unityssä ennalta määritettyjä ja ne toimivat vain jos MonoBehaviour on käytössä. MonoBehaviour on Unityn kantaluokka, joka sisältää kaikkien skriptien alustukset (7, linkit UnityEngine->Classes->MonoBehaviour). Start-metodia kutsutaan skriptissä vain kerran, heti kun skripti alustetaan. (7, linkit UnityEngine->Classes->MonoBehaviour->Start). Update-metodia kutsutaan joka kerta kun tapahtuu uusi renderöinti (7, linkit UnityEngine->Classes->MonoBehaviour->Update). Seuraavaksi esitellään pelissä käytetyt skriptit, sekä niiden käyttötarkoitukset.

3.3.1 Pelaajan liikuttaminen

Pelaajan liikuttamiseen käytettävä skripti on työn monimutkaisin ja pisin skripti, koska työssä käytetään kosketusnäyttöä liikuttamaan pelaajaa. Pelaajan liikuttaminen tapahtuu vetämällä sormea näytöllä johonkin suuntaan, esimerkiksi liikuttaessa eteenpäin, liu'utetaan sormi näytöllä alhaalta ylöspäin. Jos sormea liu'utetaankin vasemmalta oikealle tai päinvastoin, ei pelaaja liiku vasemmalle tai oikealle, vaan kääntyy suuntaan, johon sormea liu'utettiin. Kuvassa 9 nähdään millaista koodia käytetään, kun seurataan sormen liikettä näytöltä.

Skriptin alussa luodaan kaksi uutta muuttujaa: firstTouchPosition, joka on Vector2-tyyppinen ja forwardSpeed, joka on julkinen ja tyyppiä float. Muuttujien luonnin jälkeen käytetään Start-metodia, jossa ainoastaan piilotetaan kursori näytöltä. Update-metodia käytetään laskemaan ja tutkimaan mihin suuntaan sormea liu'utettiin näytöllä.

Update-metodin alussa luodaan uusi olio luokasta CharacterController ja annetaan tälle nimeksi cc. Cc alustetaan pelaajan hahmon tiedoilla, jotka tulevat skenestä, tähän käytetään GetComponent<>-metodia. Olion luonnin jälkeen luodaan kokonaislukumuuttuja, fingerCount, jota käytetään laskemaan sormien lukumäärä näytöllä. Näiden alustuksien jälkeen käydään jokainen kosketustapahtuma läpi käyttäen Foreach-lausetta. Ensimmäinen if-lause tutkii onko sen hetkisen kosketustapahtuman vaihe alussa Jos lause on tosi, niin asetetaan kyseisen tapahtuman koordinaatit firstTouchPosition-muuttujaan. Seuraava if-lause tarkastelee onko sen hetkinen kosketustapahtuma päättynyt tai lopetettu. Jos tapahtuma ei ole kumpaakaan näistä, siirrytään lauseen sisälle katsomaan mihin suuntaan sormi liikkui. Ihan aluksi kasvatetaan fingerCount-muuttujaa yhdellä, jotta sormien lukumäärä näytöllä olisi selvillä.

Kolmatta if-lausetta käytetään tarkastelemaan, onko näytöllä vain yksi sormi ja onko sen hetkisen kosketustapahtuman vaihe liikkumassa tai päättynyt. Mikäli nämä ehdot täyttyvät edetään koodissa lauseen sisälle, jossa aluksi luodaan uusi Vector2-tyyppinen muuttuja touchFacing ja annetaan sille arvoksi

firstTouchPosition vähennettynä sen hetkisen kosketuksen sijainnille. Saatu vähennyslasku normalisoidaan eli asetetaan vektorin pituudeksi 1,0, mutta suunta säilytetään samana. Seuraavat kaksi if-lausetta ovat lähes identtisiä, ainoana erona näiden kahden välillä on Vector2.Dot-metodiin annetun toisen parametrin arvo. Ensimmäisessä if-lauseessa käytetään ylöspäin suuntaavaa vektoria, ja toisessa käytetään alaspäin suuntaavaa vektoria. Nämä kaksi if-lausetta tarkastelee osoittaako touchFacing-vektori, eli sormen liukusuuntavektori, ylös tai alas. Samalla tarkistetaan onko sormi liikkunut vähintään 20 pikseliä näytöllä. Mikäli jommankumman lauseen ehdot täyttyvät, alustetaan uusi Vector3 nimeltänsä speed ja sille annetaan arvoiksi 0, 0 ja liikkumisnopeus forwardSpeed. Tämän jälkeen speed-vektori kerrotaan itsellänsä ja tämän hetkisellä katselukulmalla, jotta liikkuminen tapahtuu oikeaan suuntaan. Lopuksi käytetään cc:n metodia SimpleMove, jolle annetaan parametriksi juuri saatu speed. SimpleMove-metodin avulla liikutetaan Character-objektia pelissä.

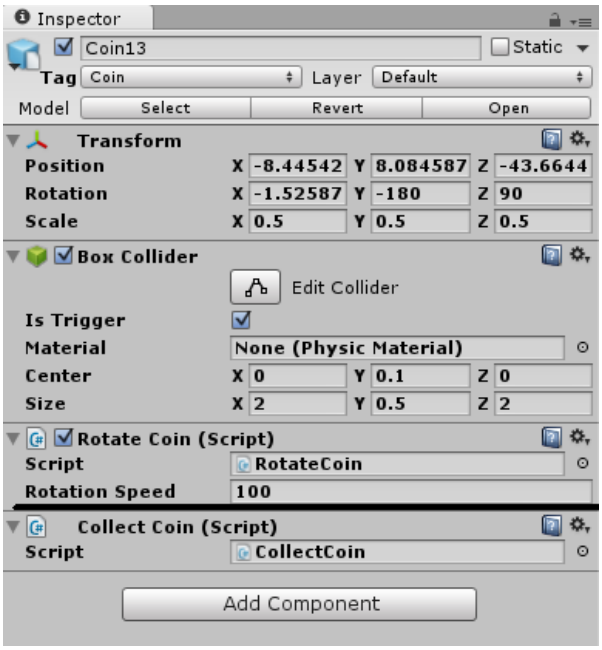
Oikealla ja vasemmalle liukuvan kosketuksen tarkistukset ovat lähes identtisiä ylöspäin ja alaspäin tarkastellessa, erona on vain se, että tässä tapauksessa Vector2.Dot-metodin toiseksi parametriksi syötetään vasemmalle tai oikealle suuntautunut vektori. Koska emme liikuta pelaajaa vasemmalle tai oikealle, on if-lauseiden sisällä oleva koodi erilainen. Haluamme vain kääntää kameraa haluttuun suuntaan. Tämä onnistuu laskemalla float-tyyppiseen muuttujaan firstTouchPositionin ja sen hetkisen kosketuksen välinen etäisyys käyttämällä Vector2.Distance-metodia. Kun välimatka on saatu laskettua, käännetään pelaajaa saadun tuloksen verran y-akselin suhteen.

```
using UnityEngine;
using System.Collections;

public class PlayerMovement : MonoBehaviour {

    Vector2 firstTouchPosition;
    public float forwardSpeed;
    // Use this for initialization
    void Start () {
        Cursor.visible = false;
    }
    // Update is called once per frame
    void Update () {
        CharacterController cc = GetComponent<CharacterController> ();
        int fingerCount = 0;
        foreach (Touch t in Input.touches) {
            if(t.phase == TouchPhase.Began)
            {
                firstTouchPosition = t.position;
            }
            if(t.phase != TouchPhase.Ended && t.phase != TouchPhase.Canceled)
            {
                fingerCount++;
                if(fingerCount == 1 && t.phase == TouchPhase.Moved || t.phase == TouchPhase.Ended)
                {
                    Vector2 touchFacing = (firstTouchPosition - t.position).normalized;
                    if(Vector2.Dot(touchFacing, -Vector2.up) > 0.8 && Vector2.Distance(firstTouchPosition, t.position) > 20)
                    {
                        //Swipe up
                        Vector3 speed = new Vector3(0, 0, forwardSpeed);
                        speed = transform.rotation * speed;
                        cc.SimpleMove(speed);
                    }
                    if(Vector2.Dot(touchFacing, Vector2.up) > 0.8 && Vector2.Distance(firstTouchPosition, t.position) > 20)
                    {
                        //Swipe down
                        Vector3 speed = new Vector3(0, 0, -forwardSpeed);
                        speed = transform.rotation * speed;
                        cc.SimpleMove(speed);
                    }
                    if(Vector2.Dot(touchFacing, -Vector2.right) > 0.8 && Vector2.Distance(firstTouchPosition, t.position) > 20)
                    {
                        //Swipe right
                        float rotRight = Vector2.Distance (firstTouchPosition, t.position);
                        transform.Rotate(0, rotRight/100, 0);
                    }
                    if(Vector2.Dot(touchFacing, Vector2.right) > 0.8 && Vector2.Distance(firstTouchPosition, t.position) > 20)
                    {
                        //Swipe left
                        float rotLeft = Vector2.Distance (firstTouchPosition, t.position);
                        transform.Rotate (0, -rotLeft/100, 0);
                    }
                }
            }
        }
```

KUVA 9. Pelaajan liikuttamiseen käytetty skripti (8)

3.3.2 Kolikon pyörittäminen

Kolikon pyörittämiseen tarvittava skripti on huomattavasti paljon yksinkertaisempi kuin hahmon liikuttamiseen tarvittava. Käytännössä pyörittäminen saadaan toteutettua kirjoittamalla vain yhden koodirivin, valmiiksi luodun koodivartalon lisäksi. Työssä kuitenkin luotiin yksi muuttuja luokan alussa, jotta pyörimisnopeutta olisi helppo vaihtaa Unityn sisällä. Mikäli Unityn skriptien muuttujat ovat julkisia (public), voidaan niiden arvoja säädellä helposti Unityn objektitietoruudusta. Kuvassa 10 nähdään esimerkki, miltä kolikon pyörittämisessä käytetty rotationSpeed-muuttuja näyttää Unityn sisällä.



KUVA 10. RotateCoin-skriptin rotationSpeed-muuttuja objektitietoruudussa

Kolikon pyörittäminen tapahtuu Update-metodin sisällä, sillä kolikko halutaan pyörivän kokoajan. Pyörittäminen toteutetaan kääntelemällä objektin nykyistä kulmaa transform.Rotate-metodilla. Metodin parametreiksi annetaan oikealle suuntautuva vektori, nykyisen update-kutsun ja edellisen update-kutsun välinen aikaero kerrottuna rotationSpeedillä, sekä objektin oma akseliavaruus. Oikealle suuntautuva vektori on akseli, jota pyöritetään, rotationSpeed kerrottuna aika-erolla merkitsee kulmaa kuinka paljon pyöritetään ja viimeinen parametri kertoo vain käytetäänkö globaalia akseliavaruutta vai objektin omaa akseliavaruutta. Kuvassa 11 nähdään työssä käytetty skripti.

```
1  using UnityEngine;
2  using System.Collections;
3
4  public class RotateCoin : MonoBehaviour {
5      public float rotationSpeed = 5;
6      // Use this for initialization
7      void Start () {
8
9      }
10
11     // Update is called once per frame
12     void Update () {
13         transform.Rotate (Vector3.right, Time.deltaTime * rotationSpeed, Space.Self);
14     }
15 }
16
```

KUVA 11. Kolikon pyörittämiseen käytetty skripti

3.3.3 Kolikon kerääminen

Kolikon keräämiseen käytettävä skripti on yhtä yksinkertainen, kuin kolikon pyörittämiseen tarvittava skripti. Tämä johtuu siitä, että Unity osaa itse laskea ja toteuttaa kahden törmäyttimen välisen fysiikan. Pelintekijä joutuu ainoastaan asettaa törmäyttimet peliobjekteille ja kertomaan moottorille mitä tehdään, kun kaksi törmäytintä kohtaa. Jotta kuvassa 12 näkyvä metodi OnTriggerEnter toimii, pitää kolikon törmäyttimen asetuksista laittaa Is Trigger käyttöön. Tämä onnistuu valitsemalla objektihierarkiasta kolikon, ja jos kolikolla ei ole vielä törmäytintä lisätään halutun mallinen, työssä laatikon muotoinen, törmäytin Add Component->Physics->Box Collider. Tämän jälkeen objektitietoruudussa tulee näkyviin törmäyttimen tiedot, josta voidaan aktivoida Is Trigger. OnTriggerEnter-metodi ottaa parametriksi törmäyttimen, joka aktivoi törmäyttimen, tässä tapauksessa pelaajan törmäytin. If-lauseen sisällä tarkastellaan, onko törmäyttimen objektin nimi Character. Mikäli if-lause toteutuu, tuhotaan se peliobjekti, johon törmättiin, työssä kolikko.

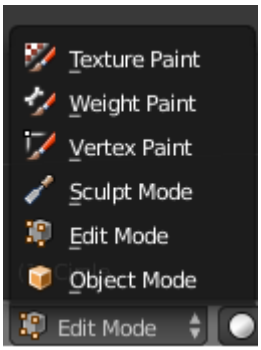
```
1  using UnityEngine;
2  using System.Collections;
3
4  public class CollectCoin : MonoBehaviour {
5
6      void OnTriggerEnter(Collider info)
7      {
8          if (info.name == "Character") {
9              Destroy (gameObject);
10         }
11     }
12 }
13
```

KUVA 12. Kolikon keräämiseen käytetty skripti

3.4 Tasot

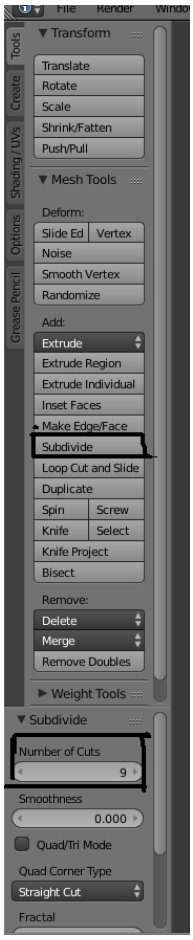
Eri tasojen pelialue voidaan tehdä joko Unityssä tai Blenderissä. Unityllä tehtäessä ei saada niin monipuolisia alueita aikaan kuin Blenderillä, joten työssä käytetään pelialueen tekemiseen Blenderiä. Tason luominen aloitetaan tekemällä uusi projekti, minkä jälkeen poistetaan valmiina oleva kuutio. Kun projekti on

tyhjä, luodaan uusi alusta (Plane) painamalla Shift + A ja valitsemalla valikosta Mesh->Plane. Blenderiin tulee 2x2 kokoinen neliö, joka skaalataan 25-kertaiseksi painamalla S, kirjoittamalla 25 ja painamalla tämän jälkeen Enter-näppäintä. Skaalauksen jälkeen vaihdetaan muokkaustilaan painamalla alareunasta Object Modea ja tämän jälkeen valitsemalla kuvan 13 mukaisesta valikosta Edit Moden.



KUVA 13. Blenderin eri tilat

Muokkaustilassa Blenderin objekteja voidaan muokata, leikata tai venyttää. Leikataan edellä luotu neliö 10x10 kokoiseksi ruudukoksi valitsemalla vasemmalta, kuvan 14 mukaisesti, Subdivide, jonka jälkeen valitaan alempaa Number of Cuts määräksi 9.

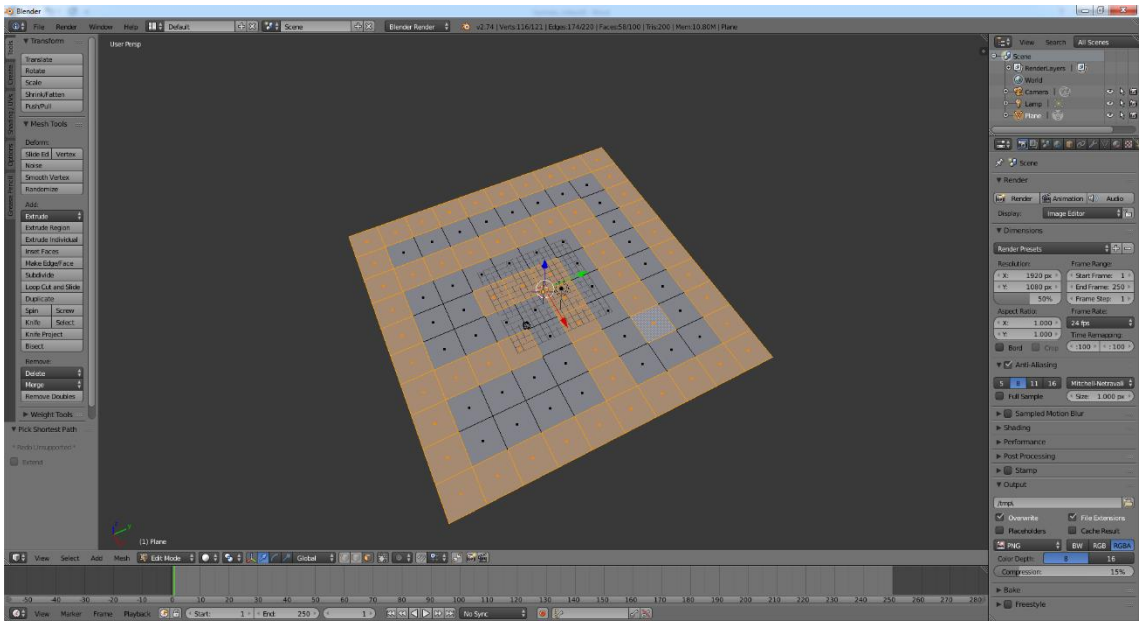


KUVA 14. Objektin paloittelu

Neliön paloittelun jälkeen valintatilaksi asetetaan kuvan 15 mukaisesta alareunan näppäimestä Face Select -tila.

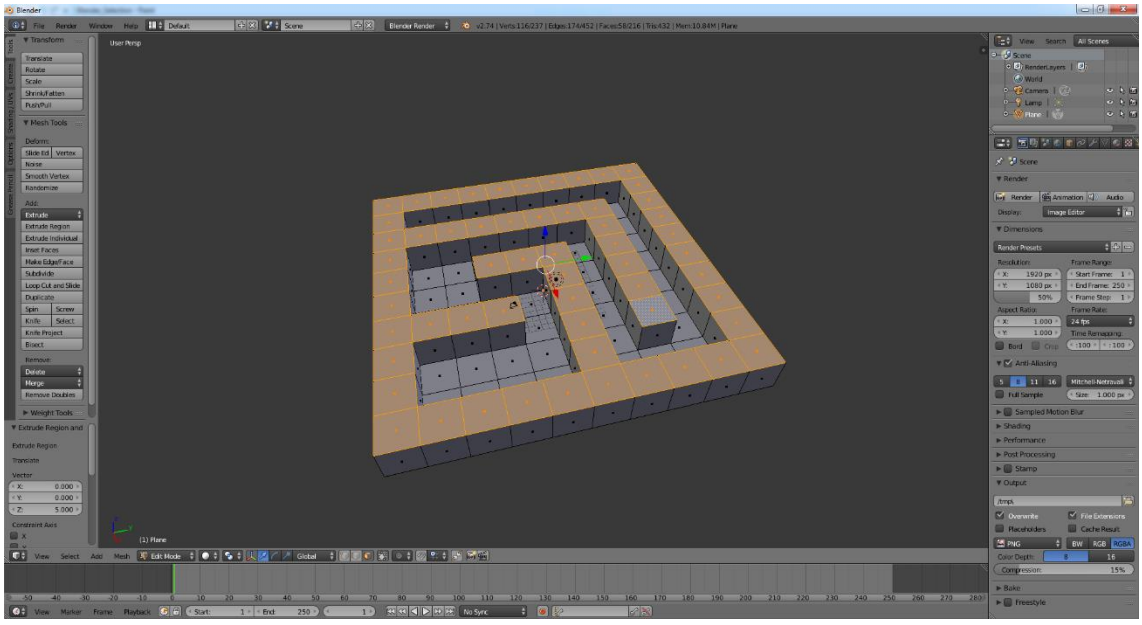
KUVA 15. Blenderin Face Select -tilan valinta

Tämän jälkeen valitaan halutut kohdat, joista tehdään pelialueen seinät, kuvassa 16 näkyy esimerkki valinta. Valinta tapahtuu hiiren oikealla näppäimellä. Jos halutaan valita useampi face siten, että edelliset valinnat eivät katoa, pidetään Shift-näppäin pohjassa.



KUVA 16. Esimerkkivalinta Blenderissä

Valinnan jälkeen venytetään valitut facet painamalla E-näppäintä ja näppäilemällä korkeus, esimerkiksi 5. Kuvassa 17 nähdään, miltä lopputuloksen tulisi näyttää.



KUVA 17. Valmis kenttä Blenderissä

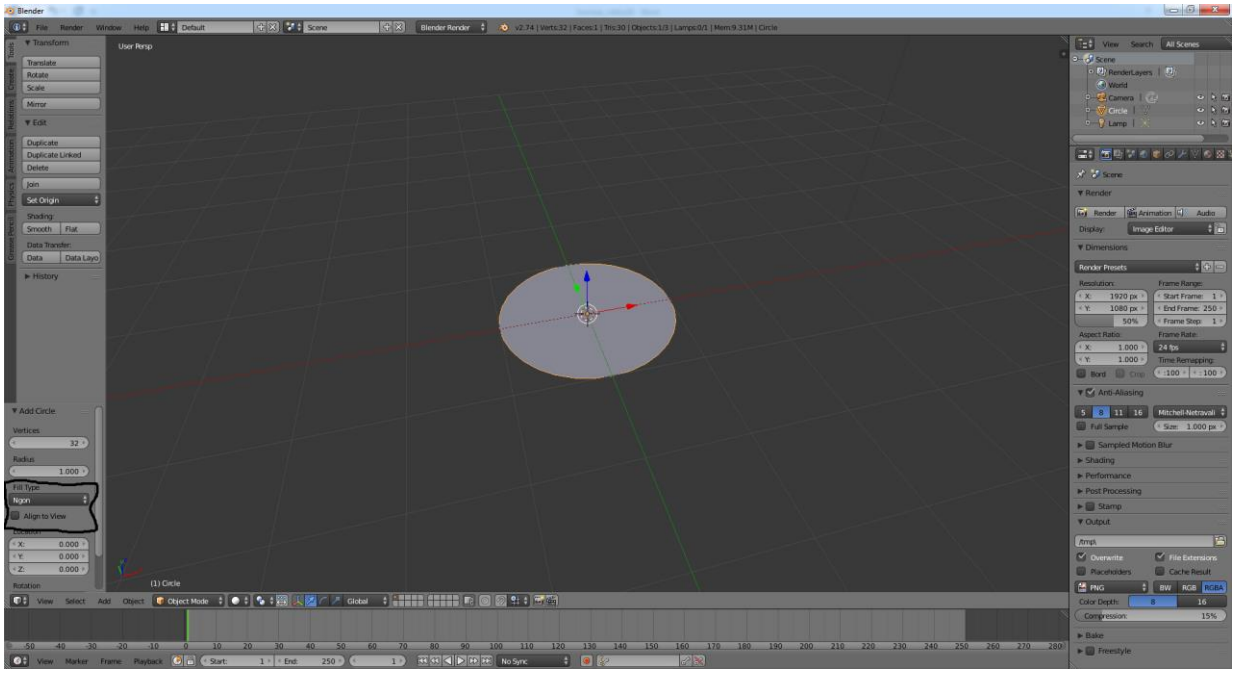
Valmis taso tallennetaan sekä Blender-projektina että fbx-tiedostomuotoon. Projekti tallennetaan siksi, että mahdolliset muutokset ovat helpompi toteuttaa, ja

fbx-muotoa käytetään liittämään malli Unityyn. Mikäli peliin halutaan luoda lisää tasoja, onnistuu se samalla tavoin kuin yllä on kerrottu. Unityn sisällä täytyy luoda uusi skene valitsemalla yläpalkista File->New Scene ja liittämällä toinen taso siihen. Tasojen vaihtaminen onnistuu käyttämällä samankaltaista toteutusta kuin päävalikossa käytettiin ChangeToScene-metodissa. Ainoana erona se, että Application.LoadLevel-funktion parametriksi annetaan uuden skenen nimi. Tätä funktiota voitaisiin kutsua esimerkiksi silloin kun kaikki kolikot on kerättynä pelialueelta.

3.5 3D-mallit

Pelin keräiltävät eli kolikot tehdään myös Blenderiä käyttäen. Tässä työssä kolikot ovat ainoa 3D-malli tasojen lisäksi mitä käytetään. Yksinkertaisen kolikon luominen Blenderissä on yksinkertaista, eikä sen tekemiseen vaadita paljoa opettelemista tai aikaa. Aloitetaan kolikon tekeminen luomalla uusi projekti, valitaan yläpalkista File->New->Reload Start-Up File, tai aukaisemalla uusi Blender ikkuna. Projektin luonnin jälkeen, poistetaan kuutio valitsemalla se hiiren oikealla näppäimellä ja painamalla X->Enter, jotta saataisiin tyhjä pelikenttä johon tehdä kolikko. Kun olemme tyhjentäneet projektin, luomme ympyrän muotoisen alustan. Tämä onnistuu painamalla Shift + A ja valitsemalla aukeavasta valikosta Mesh->Circle. Projektissa pitäisi olla nyt ympyrän muotoiset ääriviivat, joiden säde on 1,0. Jotta kolikosta tulisi kiinteä, pitää se täyttää. Täyttäminen on-

nistuu valitsemalla vasemmasta alakulmasta Fill Type->Ngon. Kuvassa 18 näkyy Fill Type -valikon sijainti.



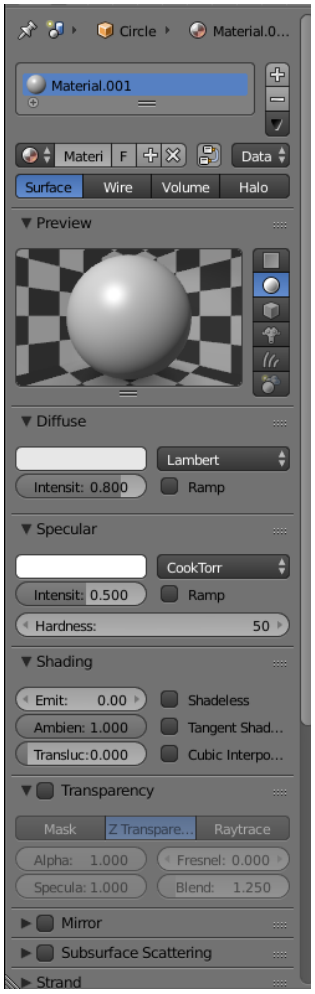
KUVA 18. Fill Type -valikon sijainti Blenderissä

Kun ympyrä on täytetty, siirrytään Blenderissä muokkaustilaan. Kolikon halutaan saavan hieman kokoa, joten venytetään ylöspäin osoittavaa sivua hieman. Venytys onnistuu painamalla E-näppäintä ja liikuttamalla hiirtä ylöspäin.

Käytännössä kolikko on tässä vaiheessa jo valmis. Jotta kolikko esittää enemmän kultakolikkoa, tulee sille antaa materiaaliksi eli väriksi kultainen. Tämä voidaan tehdä suoraan Blenderissä tai Unityssä. Mikäli halutaan monimutkaisempaa tekstuuria tai materiaalia objekteille, suositellaan käyttämään Blenderiä. Vaikka työssä ei tehdäkään monimutkaista materiaalia, havainnollistetaan materiaalin lisääminen kuitenkin Blenderiä käyttäen. Blenderin tilalla ei ole tässä vaiheessa väliä. Aloitetaan materiaalin lisääminen valitsemalla oikeasta reunasta, skenehierarkian alapuolelta, kuvan 19 mukainen ympyrän mallinen valikko.

KUVA 19. Blenderin materiaalivalikon ikoni

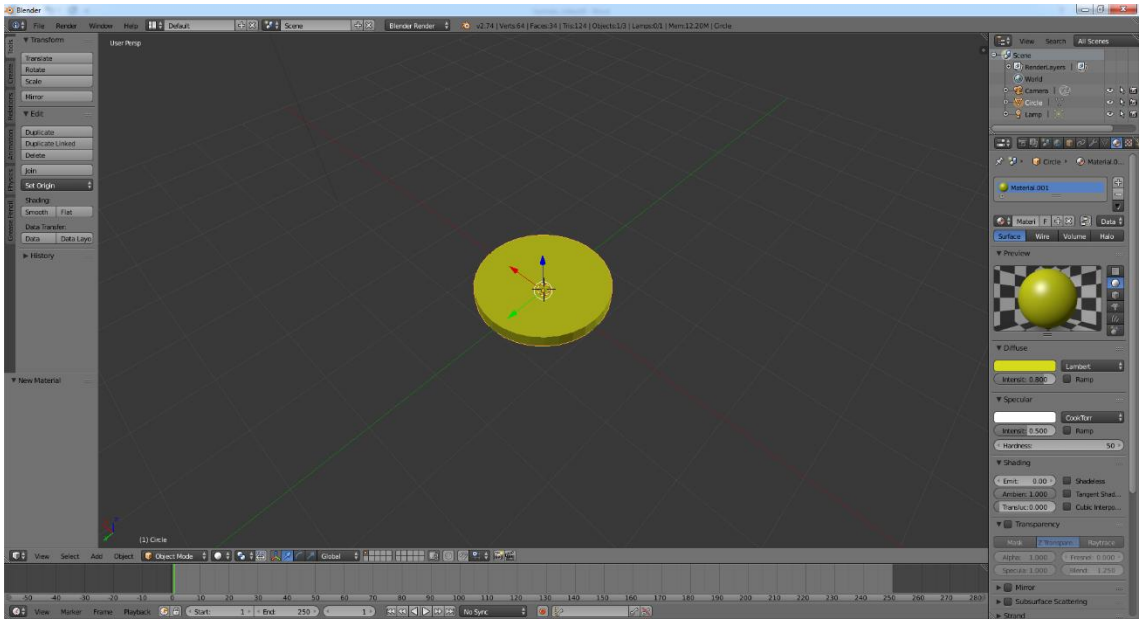
Kun materiaalivalikon ikonia on painettu, pitäisi avautua materiaalilistaus, joka on oletuksena tyhjä. Luodaan uusi materiaali painamalla New-näppäintä, jonka jälkeen avautuu kuvan 20 mukainen materiaalin muokkausvalikko.



KUVA 20. Uusi materiaali Blenderissä

Työssä halutaan vaihtaa vain kolikon väriä, mikä onnistuu yksinkertaisesti painamalla Diffuse-valikon väriä. Painalluksen jälkeen avautuu värikartta, josta voidaan valita haluttu väri tai antaa värin RGB-arvot. Kun väri on valittu, pitäisi muutoksen näkyä Blenderin ruudulla välittömästi. Kuvassa 21 näkyy esimerkki

valmiista kolikosta, jolle on annettu kultainen väri.



KUVA 21. Valmis ja värillinen kolikko

Mikäli käytetään useampaa 3D-mallia, joihin on liitettynä materiaali, kannattaa Blenderissä muokata materiaalin nimi kuvaamaan sitä, mihin se on tarkoitettu. Materiaalin uudelleen nimeäminen onnistuu kaksoisklikkaamalla materiaalin nimeä Blenderin materiaalivalikossa. Kun kolikko on saatu valmiiksi, tallennetaan se fbx-muotoon Unityn projektin Assets-hakemistoon, jotta se olisi käytettävissä pelissä. Tallennettaessa malli fbx-muotoisena Blenderi luo uuden hakemiston Materials, johon tallennetaan jokainen eri materiaali. Normaalitilanteessa näihin ei tarvitse koskea, sillä Unity osaa lukea fbx-muotoisesta mallista suoraan mikä materiaali on tarkoitettu millekin mallille.

3.6 Pelin kääntäminen Android-käyttöjärjestelmälle

Valmiin pelin kääntäminen mille tahansa alustalle on yksinkertaista Unityn sisällä. Android-käyttöjärjestelmälle käännettäessä on vain pidettävä huolta siitä, että pelintekijän tunnistamiseen käytettävät henkilökohtaiset tunnisteet ovat oikein. Tunnisteita päästään muokkaamaan valitsemalla Unityn yläpalkista File-

>Build Settings->Player Settings. Tämän jälkeen oikealle avautuu pelin paketintekoon käytettävät tunnisteet ja asetukset. Jotta Unity pystyy kääntämään pelin asennuspaketiksi, tulee Bundle Identifier asettaa muotoon com.myCompany.product, tässä tapauksessa käytetään com.myOppari.Oppari. Tunnisteiden asettamisen jälkeen lisätään skenet, joiden halutaan olevan mukana, pakkausasetuksiin. Lisääminen onnistuu joko raahaamalla projektihierarkiasta skenet tai painamalla Add Current -painiketta, jolloin Unity lisää sinä hetkenä auki olevan skenen. Skenejen järjestyksellä on sen verran väliä, että ylimpänä listassa oleva skene on se, joka aukeaa ensimmäisenä sovellusta avattaessa. Skenejen lisäämisen jälkeen valitaan haluttu käyttöjärjestelmä, tässä tapauksessa Android, painetaan Switch Platform ja sen jälkeen klikataan Build-painiketta. Tässä vaiheessa Unity kysyy, mihin APK-paketti halutaan tallentaa sekä millä nimellä. Annetaan jokin nimi ja painetaan Save. Tämän jälkeen Unity alkaa pakkaamaan pelin Androidille sopivaksi paketiksi. Paketoinnin valmistumisen jälkeen voidaan saatu paketti siirtää Android-puhelimelle tai -tabletille haluamallaan keinolla.

4 YHTEENVETO

Työn tarkoituksena oli tutustua Unity-pelinkehitysympäristöön, Blender-3D-mallinnusohjelmistoon, sekä lyhyesti Android-käyttöjärjestelmän historiaan. Tavoitteena oli saada aikaiseksi näitä kolmea ohjelmistoa käyttäen yksinkertainen peli. Peliä piti olla vain sen verran valmiina, että sitä voidaan esitellä. Tässä onnistuttiin hyvin. Mikäli peli haluttaisiin julkaista, tulisi siihen käyttää vielä paljon aikaa.

Pelin tekeminen Unityllä on suhteellisen yksinkertaista, sillä se antaa todella hyvän pelimoottorin jo valmiiksi. Jos pelimoottori kehitettäisiin alusta alkaen tyhjästä, veisi se ainakin vuoden päivät toteuttaa. Blender-ohjelmisto antaa käyttäjälle paljon mahdollisuuksia tehdä, mutta ensikertalaiselle ohjelmisto voi olla monimutkainen. Työssäkin käytetyt yksinkertaiset 3D-mallit ovat suhteellisen yksinkertaisia toteuttaa, mutta siirryttäessä vaativimpiin malleihin pitäisi 3D-mallinnuksesta tietää enemmän. Android-käyttöjärjestelmä antaa hyvän alustan mobiilipelikehitykselle. Uusien versioiden tullessa, Android-pohjaisten pelien tehokkuus vain kasvaa.

LÄHTEET

1. Cohen, Heidi 2013. 67 Mobile Facts To Develop Your 2014 Budget. Saatavissa: http://heidicohen.com/67-mobile-facts-from-2013-research-charts/. Hakupäivä 8.4.2015.

2. Unity – Fast Facts. Unity Technologies. 2015. Saatavissa: http://unity3d.com/public-relations. Hakupäivä 8.4.2015.

3. About – blender.org – Home of the Blender project – Free and Open 3D Creation Software. Blender Foundation. 2015. Saatavissa: http://www.blender.org/about/. Hakupäivä 8.4.2015.

4. Features – blender.org – Home of the Blender project – Free and Open 3D Creation Software. Blender Foundation. 2015. Saatavissa: http://www.blender.org/features/. Hakupäivä 8.4.2015.

5. Verge Staff 2011. Android: A visual history. Saatavissa: http://www.theverge.com/2011/12/7/2585779/android-history. Hakupäivä 15.4.2015.

6. Codenames, Tags, and Build Numbers. Android Open Source Project. Saatavissa: http://source.android.com/source/build-numbers.html. Hakupäivä 17.4.2015.

7. Unity Scripting Reference. 2015. Saatavissa: http://docs.unity3d.com/ScriptReference/index.html. Hakupäivä 7.5.2015.

8. Clark, Nigel. 2013. Unity Mobile Touch Interface - Simple Swipe System. Saatavissa: https://www.youtube.com/watch?v=esb0RvqdqvU. Hakupäivä 11.5.2015.

LIITE2/1

OULUN AMMATTIKORKEAKOULU

Tommi Vanhala

HUHPLAY-SOVELLUS ANDROID-KÄYTTÖJÄRJESTELMÄLLE

HUHPLAY-SOVELLUS ANDROID-KÄYTTÖJÄRJESTELMÄLLE

Tommi Vanhala
Opinnäytetyö osa 2 + 3
Syksy 2016
Tietotekniikan koulutusohjelma
Oulun ammattikorkeakoulu

SISÄLLYS

SANASTO

HTTP	Hypertext Transfer Protocol, tiedonsiirtoon käytetty protokolla
PHP	Hypertext Preprocessor, palvelimella käytetty ohjelmointikieli
API	Application Programming Interface eli ohjelmointirajapinta
MySQL	Tietokantaohjelmisto
JSON	Javascript Object Notation, eräänlainen tiedostomuoto
SQL	Structured Query Language, relaatiotietokannoissa käytetty kysely-kieli
AOSP	Android Open Source Project, Androidin avoimen lähdekoodin projekti
XML	Extensible Markup Language, kieli jolla voidaan esittää tai tallentaa tietoja

1 JOHDANTO

Tämä opinnäytetyön osa on tehty urheiluseura TKF Freestyle Teamille, jonka edustajana toimii Seppo Kangas. Työn tarkoituksena oli tehdä urheilijoille suunnattu harjoituspäiväkirja-sovellus. HUHPlayn pääideana on antaa urheilijoille mahdollisuus harjoituksien tallentamiseen ja vertailuun. Työn päämääränä oli saada Android-sovelluksen ja palvelinpuolen toiminnollisuus valmiiksi.

HUHPlayn avulla urheiluseurojen tukijat saavat uudenlaista vastinetta tukirahoilleen. Tukijat voivat seurata urheilijoiden harjoituksia sekä vertailla omia harjoituksiaan urheilijoiden kanssa. Sovellus on suunniteltu antamaan urheiluseuroille mahdollisuus myydä sovellukseen käyttökoodeja, joiden avulla rekisteröidytään sovelluksen käyttäjäksi.

Sovelluksen palvelinpuolen ohjelmointikieleksi valittiin PHP sen yksinkertaisuuden, hyvän dokumentaation ja kattavan kirjastojen vuoksi. Ensimmäiseksi julkaisualustaksi valikoitui Android-käyttöjärjestelmä oman ohjelmointikokemuksen ja alustan helppokäyttöisyyden vuoksi.

2 SOVELLUKSEN TAUSTAT

Älypuhelimet ja mukana kulkeva internetyhteys ovat tuoneet ihmisille uusia keinoja täyttää heidän sosiaalisia tarpeitaan. Eräs näistä keinoista ovat erilaiset sosiaaliset mediat kuten Facebook, Twitter, Snapchat tai Instagram. Sosiaalisen median sovellukset mahdollistavat ihmisten jatkuvan sosiaalisuuden ja tavoitettavuuden.

2.1 Taustat

Työssä tehtävän sovelluksen tarpeet voidaan jakaa kahteen: urheilijoiden ja tukijoiden eri tarpeisiin. Urheilijat tarvitsevat helpon ja mukana kulkevan seuranta- ja vertailupalvelun omille treeneilleen. Sponsorointi taasen on jämähtänyt perinteiseen näkyvyyden myyntiin. Sovellus tuo tukijoille uudenlaista vastinetta urheilijoiden seurantaan, sillä tukijat pääsevät seuraamaan urheilijoiden treenejä. Koska sovelluksessa hyödynnetään ihmisille tuttuja sosiaalisen median toimintatapoja, on sovelluksen käyttäminen helppoa. Yrityksille taasen sovellus tuo uuden tavan edistää henkilökunnan hyvinvointia. (1.)

Sovellus tuo hyötyjä sekä tukijoille että urheilijoille. Haittoja palvelun käyttämisessä ei ole, mikäli suoritusten merkitseminen ja käyttöliittymä pidetään selkeinä ja nopeina. Ainoa mahdollinen haitta voi olla se, että yksittäinen harjoittelija kokee omat harjoituksensa niin yksityiseksi, ettei hän voi niitä jakaa. Urheilijoille sovellus tuo motivaatiota harjoitteluun sekä mahdollisuuden saada taloudellista tukea harjoitteluun. Urheilijat voivat keskenään vertailla tuloksiaan ja kommentoida niitä. Omien harjoituksien jakaminen tuo positiivista sosiaalista painetta harjoitella ahkerasti. Samalla nuoret urheilijat oppivat, miten toimitaan vastuullisesti yhteistyökumppanien kanssa. Tukijoille sovellus tuo täysin uudenlaisen tavan tukea urheilijoita. Poiketen perinteisistä sponsorointimenetelmistä, yrityksen koko henkilöstö ja verkosto pystyvät osallistumaan urheilijoiden seuraamiseen. Samalla sponsoroiva yritys tukee oman henkilöstönsä aktiivisuutta ja liikuntaa. (1.)

2.2 Sosiaalisen median sovellukset ja palvelut

Sosiaalinen media voidaan määritellä sellaiseksi työkaluksi tai ohjelmaksi, jolla jaetaan informaatiota toisille ihmisille (2). Tämän määritelmän mukaan myös työssä tehtävä sovellus on eräänlainen sosiaalinen media, sillä sovellukseen merkitään omien harjoituksien minuuttimääriä, joita urheilijoita seuraavat joukkueet voivat halutessaan tarkastella. Sosiaalisten medioiden käyttäjämäärät ovat kasvaneet todella nopeasti viime vuosina. Esimerkiksi vuonna 2010 Facebookilla oli noin 420 miljoonaa aktiivista kuukausittaista käyttäjää ja 2016 vuoden elokuussa niitä oli 1590 miljoonaa (3).

Sosiaaliset mediat ovat koukuttavia ja etenkin nuoriso käyttää paljon aikaa niiden selaamiseen. Joulukuussa 2015 tehdyn tutkimuksen mukaan 92 prosenttia 18–34-vuotiaista oli rekisteröitynyt Facebookiin ja käytti siellä kuukaudessa keskimäärin aikaa 1050 minuuttia. Toisena esimerkkinä samaisessa tutkimuksessa huomattiin, että Instagram-palvelua käytti 63 prosenttia 18–34-vuotiaista ja he käyttivät siihen noin 300 minuuttia kuukaudessa. (3.) Työssä tehtävän sovelluksen vetonaulana käytetäänkin juuri ihmisten halua jakaa omia harjoituksiaan ja niiden vertailua toistensa kanssa.

2.3 Sovelluksen käyttötapaukset

Sovelluksen käyttäjinä toimivat pääasiassa urheilijat, tukijat, työnantajat sekä työntekijät. Käyttötapaukset ovat hieman erilaiset käyttäjän roolin mukaan. Käytännössä kuitenkin käyttötapaukset voidaan jakaa kahteen osaan, urheilijan käyttötapaukset ja tukijan käyttötapaukset. Varsinaisia käyttötapauksia ovat harjoituksien merkkaaminen ja muiden tuloksien vertaileminen.

Harjoittelun päätyttyä urheilija avaa sovelluksen, menee tarkastelemaan omia pisteitään ja merkkaa harjoittelun määrän. Mikäli urheilija haluaa vertailla omia harjoituksiaan muiden kanssa, hän menee päänäkymään, josta urheilija näkee heti joukkuekavereidensa sen viikon tulokset. Jatkossa urheilija pystyy myös määrittelemään harjoittelun tyypin sekä lisäämään useamman erilaisen treenin jokaiselle päivälle.

Tukijoille sovellus antaa mahdollisuuden seurata heidän sponsoroimiaan urheilijoita. Tukija kirjautuu sovellukseen sisään ja näkee päänäkymästä heti jokaisen seurattavan urheilujoukkueen. Mikäli tukija haluaa tarkastella tarkemmin urheilijoiden suorituksia, hän painaa urheilijan nimeä, jolloin avautuu tarkka viikkonäkymä harjoituksista. Työnantajan ja työntekijän käyttötapaukset ovat identtiset urheilijoiden käyttötapausten kanssa. Jokainen sovellusta käyttävä työntekijä voi merkitä omia suorituksiaan ohjelmaan ja vertailla työporukan keskuudessa näitä. Työnantajalla on myös samat oikeudet.

3 PALVELINOHJELMISTO

Palvelimen käyttö on välttämätöntä sovelluksen toiminnan kannalta, koska käytettävien tietojen tulee olla jokaisen käyttäjän saatavilla. Sovelluksen palvelin on tilattu webbinen.net-palvelusta. Palvelinta käytetään tietojen tallentamiseen ja niiden hakemiseen. Palvelimelle on valmiina asennettuna PHP-tuki ja MySQL-tietokanta.

3.1 Slim-viitekehys

Tietojen tallentamiseen ja lukemiseen käytetään HTTP-pyyntöjä. Taulukossa 1 esitellään jokainen käytettävissä oleva pyyntötyyppi.

TAULUKKO 1. HTTP-pyynnöt ja niiden käyttötarkoitus (4)

Pyynnön nimi	**Käyttötarkoitus**
OPTIONS	Pyydetään käytettävissä olevia asetuksia.
GET	Käytetään tietoja pyydettäessä.
HEAD	Sama kuin GET, mutta palvelin palauttaa vain viestin otsakkeet.
POST	Käytetään tietoja tallennettaessa.
PUT	Käytetään tietoja tallennettaessa tai päivittäessä.
DELETE	Käytetään tietojen poistamiseen.
TRACE	Käytetään tietojen seuraamiseen, nähdään mitä tietoja pyynnön loppupäässä vastaanotetaan.
CONNECT	Varattu sana, jonka avulla voidaan luoda verkkotunneleita.

Työssä tuotetussa sovelluksessa käytetään vain kolmea näistä: GET-, POST- ja PUT-pyyntöjä.

Jotta HTTP-pyyntöjen käsittely palvelimella olisi yksinkertaista ja tehokasta, käytetään palvelimella Slim-viitekehystä. Slim on todella pienikokoinen PHP-viitekehys, jonka avulla voidaan nopeasti kirjoittaa yksinkertaisia ja tehokkaita nettisovelluksia ja APIa. Kirjoitushetkellä Slim-viitekehyksen tuorein versio on 3.5.0, joka julkaistiin 26. heinäkuuta 2016. (5.) Jokaisessa HTTP-pyynnössä on otsake-kohta, joka kertoo pyynnön tyypin. Slimin toiminta perustuu siihen, että se tutkii jokaisen sille lähetetyn HTTP-pyynnön ja vertailee, onko pyyntöä vastaavaa metodia olemassa, ja mikäli on, suorittaa sen.

Otetaan malliesimerkiksi Slimillä toteutettava yksinkertainen GET-pyyntö. Esimerkissä aluksi määritetään luokille ServerRequestInterface- ja ResponseInterface-nimimerkki, jotta kyseisiin luokkiin voidaan viitata lyhyemmillä nimillä. Tässä tapauksessa käytetään Request- ja Response-nimiä. Tämän jälkeen kerrotaan, että käytetään vendor-kansiosta löytyvää autoload.php-tiedostoa. Kyseinen tiedosto sisältää kaiken Slim-viitekehyksen alustukseen tarvittavan koodin. Alustuksen jälkeen luodaan uusi muuttuja nimeltä app Slim-viitekehyksen luokasta App. Slimin App-luokka on jokaisen Slim-ohjelman lähtökohta, sillä sitä käytetään jokaisen HTTP-pyynnön reitittämiseen (6).

Olion alustuksen jälkeen päästään tekemään ensimmäinen reititys. Oletetaan, että käytössämme on osoite http://testi.com. Esimerkin reititys tapahtuu http://testi.com/hello-osoitteelle, jonka perään odotetaan yhtä GET-pyynnön parametria. GET-pyynnöille parametrit annetaan suoraan osoitteen perään esimerkiksi http://testi.com/hello?name=Tommi. Reititykseen annetaan myös funktio, jota kutsutaan aina silloin, kun palvelin huomaa määritetyn HTTP-pyynnön tulleen. Esimerkissä funktio ottaa vain parametrina annetun nimen talteen ja tulostaa sen näytölle. Kaikkien reitityksien määrittelyn jälkeen tulee muistaa kutsua app-olion run-metodia. Ilman viimeistä kutsua Slim ei osaa käsitellä sille lähetettyjä pyyntöjä. Kuvassa 1 nähdään otetun malliesimerkin toteutus ohjelmistokoodissa.

```php
<?php
use \Psr\Http\Message\ServerRequestInterface as Request;
use \Psr\Http\Message\ResponseInterface as Response;

require 'vendor/autoload.php';

$app = new \Slim\App;
$app->get('/hello/{name}', function (Request $request, Response $response) {
    $name = $request->getAttribute('name');
    $response->getBody()->write("Hello, $name");

    return $response;
});
$app->run();
```

KUVA 1. Slim-viitekehyksellä luotu yksinkertainen reititys

3.2 PHP

PHP-ohjelmointikieli on laajasti käytetty ja kohdistettu erityisesti web-kehitykseen. PHP:tä voidaan upottaa myös HTML-koodin sisään, mutta kaikki koodi ajetaan kuitenkin palvelimen puolella. (7.) Kirjoitushetkellä PHP5:n tuorein versio on 5.6.27 ja PHP7:n 7.0.12. Työssä käytettävän palvelimen PHP-versio on 5.6.

Koska työssä tallennettavien tietojen tulee olla verkon ylitse saatavissa, täytyy meidän tallentaa tiedot palvelimen tietokantaan. PHP:n käyttäminen on välttämätöntä tässä tilanteessa, sillä tietokanta sijaitsee palvelimella eikä siihen pääse käsiksi muuten kuin palvelimella ajettavalla ohjelmakoodilla. Muita mahdollisuuksia PHP:n sijasta voisivat olla esimerkiksi Java tai Ruby, mutta työssä käytetään PHP:tä sen yksinkertaisuuden vuoksi.

Jokainen PHP-koodi tulee aloittaa *<?php*-tagilla ja se voidaan lopettaa *?>*-tagiin. Lopetustagin käyttäminen ei ole pakollista. Suositus on, että mikäli koodi on pelkästään PHP-ohjelmointikieltä, jätetään lopetustagi pois (8). Kuvassa 2 nähdään erittäin yksinkertainen esimerkki PHP-ohjelmasta. Esimerkissä alustetaan hello-niminen muuttuja, jolle annetaan arvoksi merkkijono ”Hello World!”.

Tämän jälkeen se tulostetaan käyttäjän näkyville komennolla *echo*. Toisin kuin joissain ohjelmointikielissä PHP:lle ei tarvitse kertoa ollenkaan muuttujien tyyppiä, vaan PHP päättelee muuttujalle annetusta arvosta, mikä tyyppi on kyseessä. Lisäksi PHP:ssa erilaista muihin verrattuna on se, että muuttujiin viitataan $-merkillä.

```php
<?php
  $hello = "Hello World!";
  echo $hello;
```

KUVA 2. Esimerkki PHP-koodista

3.3 Tärkeimmät funktiot

Varsinaiset tietokantakyselyt ovat sijoitettuna omaan tiedostoonsa. Tämän tiedoston funktiot ottavat vain vastaan tarvittavat parametrit ja joko lukevat tai päivittävät tietokantaa. Kaikki olennaisimmat toiminnot ovat sijoitettuna yhteen tiedostoon, jossa ohjataan HTTP-pyynnöt Slim-viitekehyksen avulla.

Jokainen esitelty funktio on toteutettu Slimin avulla. Jokainen funktio palauttaa vastauksen JSON-muodossa, vaikka pyyntö epäonnistuisi. Vastauksen rakenne näkyy kuvassa 3. Eri funktiot voivat palauttaa eri tietoja vastauksessa, mutta perusrakenne on sama.

```
{
    "error": false
    "message": "User creation success"
}
```

KUVA 3. JSON-rakenne esimerkki

3.3.1 Rekisteröinti

Ensimmäinen tärkeä funktio on rekisteröinti. Funktio käyttää POST-pyyntöä. Funktio ottaa vastaan käyttäjänimen, salasanan, sähköpostiosoitteen sekä joukkuekoodin. Tämän jälkeen varmistetaan, että sähköpostiosoite on oikeaa muotoa kuten testi@testi.fi. Mikäli osoite ei ole oikeaa muotoa, palautetaan vastauksena virheilmoitus. Jos osoite on oikeaa muotoa, tarkistetaan joukkuekoodin olemassaolo. Tämän tarkistuksen jälkeen luodaan tunnus ja vastataan tunnuksen luonnin onnistuneen.

3.3.2 Kirjautuminen

Kirjautumisfunktio käyttää POST-pyyntöä ja tämän funktion tarkoituksena on sisäänkirjautuminen. Funktio ottaa vastaan käyttäjänimen ja salasanan, tarkistaa näiden oikeellisuuden ja palauttaa käyttäjätiedot, mikäli tiedot ovat oikein. Virhetilanteen sattuessa funktio palauttaa virheilmoituksen.

3.3.3 Oikeuksien tarkistaminen

Mikäli käytetään sellaista funktiota, jonka käyttämiseen vaaditaan käyttöoikeuksia, tarkistetaan käyttäjän oikeudet ennen toteutusta. Käytännössä tämä tarkistaa, onko käyttäjätunnus oikeutettu tekemään haluttua muutosta.

3.3.4 Treenien käsittely

Funktio *Train* käyttää joko POST-, GET- tai PUT-pyyntöä. Pyynnön tyypin mukaan tehdään eri asioita. Jokaisen *Train*-funktiota edeltää käyttöoikeuksien tarkistaminen.

POST

POST-funktiota käytetään luomaan käyttäjälle tyhjä treeni. Tämän luominen on tärkeää ohjelman toiminnan kannalta, sillä ilman tätä sovellus ei osaa käsitellä yhtäkään treeniä. Tämä funktio ottaa parametrina vastaan päivämäärän, treenin tyypin ja treeniminuuttien määrän. Treeniminuutit ovat aina nolla, koska luodaan

tyhjä treeni. Annettujen parametrien mukaan luodaan treeni ja palautetaan tieto onnistumisesta.

GET

Joissain tapauksissa halutaan tietää kaikki kirjautuneen käyttäjän treenit. Tässä tapauksessa käytetään *Train*-funktiota GET-pyynnöllä. Mikäli käyttäjällä on oikeus kysyä tätä, palautetaan kaikki kirjautuneen käyttäjän treenit.

PUT

Aina kun halutaan tehdä muutoksia olemassa olevaan treeniin, esimerkiksi lisätä tai vähentää minuutteja, käytetään *Train*-funktiota PUT-pyynnöllä. Tämä funktio ottaa vastaan parametreina päivämäärän, treenin tyypin ja treeniminuutit. Tämän jälkeen haetaan haluttu treeni päivämäärän mukaan ja päivitetään kyseinen treeni uusilla arvoilla. Lopuksi palautetaan tieto onnistumisesta.

3.3.5 Seuratut joukkueet

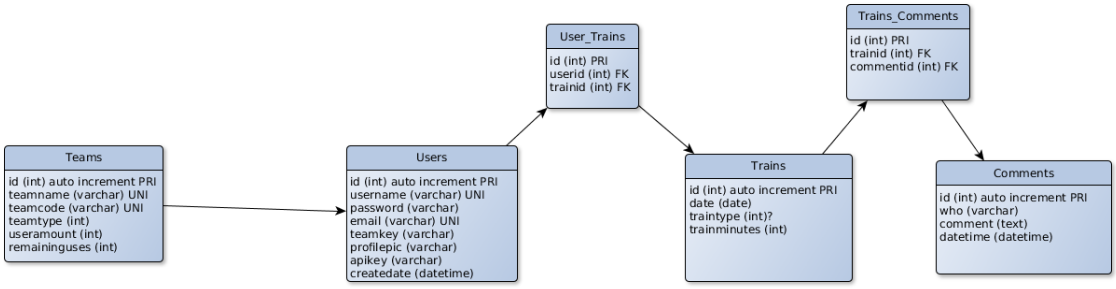
Koska sovellus on sosiaalinen media, jossa ihmiset voivat seurata toisia joukkueita, täytyy olla keino hakea kaikkien seurattujen joukkueiden tiedot. Tähän käytetään funktiota *Team* GET-pyynnöllä. Kyseinen funktio tekee paljon tarkistuksia ja käsittelyitä tiedoille. Kiteytettynä funktion tarkoituksena on hakea kaikki joukkueet, tarkistaa seuraako oma joukkue mitä näistä joukkueista, hakea jokaisen seuratun joukkueen käyttäjät, tallentaa ne väliaikaismuuttujiin ja palauttaa vastauksena nämä tiedot.

3.4 Tietokanta

Tietokantaa käytetään tiedon tallettamiseen ja lukemiseen. Tietokantaohjelmistona toimii MySQL versio 5.5.52. MySQL on suosituin avoimen lähdekoodin SQL-tietokantaohjelmisto. Sen kehittäjänä, jakajana ja tukijana toimii Oracle

Corporation. MySQL on relatiivinen tietokanta, jonka vuoksi kaikki tiedot ovat omissa tauluissaan eivätkä yhtenä isona tietokekona. (9.)

Työssä käytettävä tietokantarakenne näkyy kuvassa 4. Tietokannassa on neljä tiedon tallentamiseen käytettävää taulua sekä kaksi liitostaulua. Liitostauluja käytetään siksi, että voidaan yhdistää käyttäjät ja treenit yhteen sekä treenit ja kommentit yhteen. Yhdistämisellä tarkoitetaan sitä, että tietyllä käyttäjällä on tietyt treenit ja tietyillä treeneillä tietyt kommentit. Tietotauluja ovat *Teams, Users, Trains* ja *Comments*. Jokaiseen talletetaan nimen mukaisia tietoja.



KUVA 4. Tietokantarakenne

Tietokannasta luetaan ja sinne talletetaan tietoja PHP-koodissa, joita kutsutaan HTTP-pyynnöillä Slim-viitekehystä käyttäen.

4 ANDROID-SOVELLUS

Android-käyttöjärjestelmä on Googlen valmistama mobiilikäyttöjärjestelmä. Käyttöjärjestelmä pohjautuu avoimen lähdekoodin Linux-käyttöjärjestelmään ja se on maailman suosituin mobiilikäyttöjärjestelmä. Android-laitteita on satoja miljoonia kappaleita yli 190 eri maassa. Joka päivä noin miljoona uutta Android-laitetta käynnistetään ensimmäisen kerran. (10.)

Sovellus tehdään ensimmäisenä Android-käyttöjärjestelmälle. Sovelluksessa käytetään Volley-kirjastoa, jonka avulla HTTP-pyynnöt toteutetaan. Näkymät toteutetaan XML-ohjelmointikielellä ja toiminnollisuus Java-kielellä.

4.1 Volley

Volley on HTTP-kirjasto, joka helpottaa ja nopeuttaa Android-sovelluksien tietoverkkoyhteyksiä. Tärkeimpinä etuina Volley tuo automaattisen aikataulutuksen, useamman yhtäaikaisen yhteyden ja helpon muokattavuuden. Volleyn käyttöönottaminen on todella yksinkertaista, koska se sijaitsee avoimessa AOSP-tietolähteessä. Lisääminen tapahtuu lisäämällä Android-projektin *build.gradle*-tiedostoon *dependencies*-lohkon sisään rivi *compile 'com.android.volley:volley:1.0.0'*. (11.) Tämän jälkeen käännetään projekti käyttäen Gradle-kääntäjää. Käännöksen jälkeen Volley-kirjasto on käytettävissä projektissa.

Ennen kuin aloittaa Volleyn tai minkä tahansa internetyhteyttä vaativan asian Android-laitteella, pitää ohjelmalle kertoa, että tarvitaan käyttöoikeus internetyhteydelle. Tämä onnistuu lisäämällä rivi <*uses-permission android:name="android.permission.INTERNET" /*> AndroidManifest.xml-tiedostoon heti <*manifest*>-rivin alapuolelle.

Korkealla tasolla Volley-kirjastoa käytetään luomalla RequestQueue eli pyyntöjono ja annetaan sille Request-olioita eli pyyntöjä. Pyyntöjono käsittelee pyynnöt omissa säikeissään. (12.) Kuvassa 5 nähdään esimerkkikoodi Volley-kirjaston käyttämisestä. Kyseisessä esimerkissä aluksi luodaan *RequestQueue*-olio ja luodaan String-tyyppinen muuttuja, jolle annetaan arvoksi osoite

http://www.google.com. Näiden kahden luomisen jälkeen tehdään *StringRequest*-olio, jonka tarkoituksena on pyytää merkkijono-tyyppinen vastaus annetusta osoitteesta. Tälle oliolle annetaan ensimmäiseksi parametriksi HTTP-pyynnön tyyppi, toiseksi parametriksi osoite, kolmanneksi vastauksen kuuntelija ja viimeiseksi virhetilanteen kuuntelija. Lopuksi pitää muistaa lisätä *StringRequest*-olio aiemmin luotuun pyyntöjonoon, jotta ohjelma aloittaa pyynnön.

```
final TextView mTextView = (TextView) findViewById(R.id.text);
...

// Instantiate the RequestQueue.
RequestQueue queue = Volley.newRequestQueue(this);
String url ="http://www.google.com";

// Request a string response from the provided URL.
StringRequest stringRequest = new StringRequest(Request.Method.GET, url,
            new Response.Listener<String>() {
    @Override
    public void onResponse(String response) {
        // Display the first 500 characters of the response string.
        mTextView.setText("Response is: "+ response.substring(0,500));
    }
}, new Response.ErrorListener() {
    @Override
    public void onErrorResponse(VolleyError error) {
        mTextView.setText("That didn't work!");
    }
});
// Add the request to the RequestQueue.
queue.add(stringRequest);
```

KUVA 5. HTTP-pyyntö Volley-kirjastoa käyttäen

4.2 Näkymät

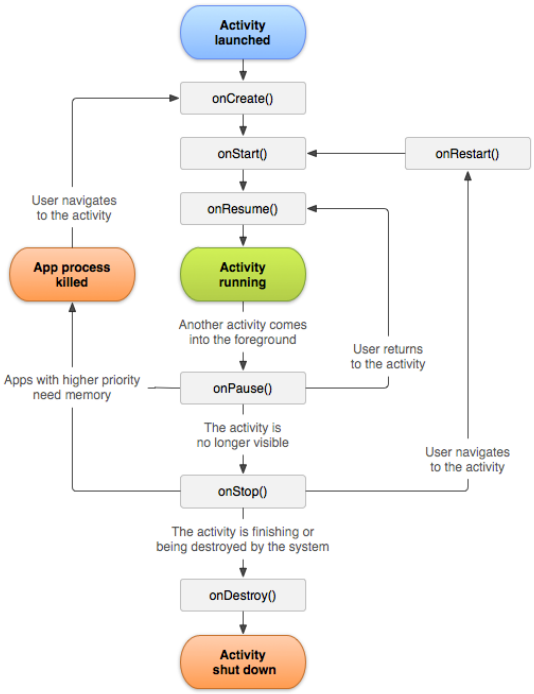
Jokainen Android-sovelluksen näkymä luodaan käyttäen XML-kieltä. XML-kielellä kuvaillaan ohjelmalle, minkälaisia objekteja luodaan ja mihin kohtaan ne asetetaan. XML:ää voidaan käyttää myös tiedon talletukseen samalla periaatteella kuin JSONia. Kuvassa 6 nähdään yhdenlainen tapa esittää tietoja XML-kielellä.

```
<Tieto>
  <kenelle>Kaikille</kenelle>
  <keneltä>Minulta</keneltä>
  <otsikko>Muistutus</otsikko>
  <teksti>Muista ostaa maitoa kaupasta</teksti>
</tieto>
```

KUVA 6. XML-koodi tietokenttänä

4.2.1 Activityn elämänkaari

Android-ohjelmat käyttävät nimitystä Activity jokaisesta näkymästä. Activityn määritelmä on: Activity on yksi kohdennettu asia, jonka käyttäjä voi tehdä (10). Activityllä on tietty elämänkaari, jota se aina seuraa. Suoritus alkaa onCreate-funktiosta, jonka jälkeen kutsutaan onStart-funktiota ja viimein onResume-funktiota. Tämän jälkeen Activity on käynnissä. Kun jokin toinen Activity tulee näkyville, kutsutaan piiloon menevässä näkymässä onPause-funktiota. Mikäli näkymä piiloutuu kokonaan, kutsutaan onStop-funktiota. Jos käyttäjä palaakin edelliseen näkymään, kutsutaan onResume-funktiota ja edellinen Activity on taas käynnissä. OnStop-funktion jälkeen on kolme vaihtoehtoa, mitä Activitylle voi tapahtua. Ensimmäinen on, että Activity tuhotaan, jolloin kutsutaan onDestroy-funktiota ja Activity tuhotaan. Toinen on, että käyttäjä palaa takaisin samaan Activityyn, jolloin kutsutaan onRestart-funktiota ja edetään normaaliin tahtiin. Viimeinen vaihtoehto on, että korkeammalla prioriteetilla olevat ohjelmat vaativat muistia. Tällöin ohjelman suoritus lopetetaan ja jäädään odottelemaan uudelleenkäynnistystä. (13.) Kuvassa 7 esitetään Activityn elämänkaari kuvana.



KUVA 7. Android Activityn elämänkaari (13)

4.2.2 HUHPlayn näkymät

HUHPlayssä on yhteensä viisi erilaista näkymää. Jokaisella näkymällä on oma tehtävänsä. Näkymästä toiseen siirtyminen tapahtuu Java-koodissa käyttäen Intent-luokasta tehtyä oliota. Kuvassa 8 nähdään, miten siirtymä toteutetaan koodissa.

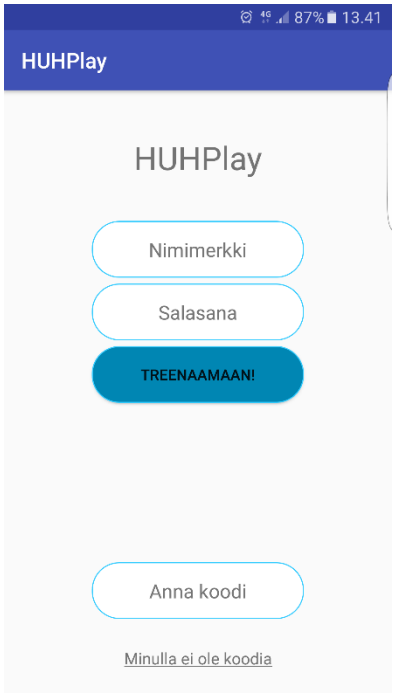
```
function nextActivity() {
  Intent intent = new Intent(getBaseContext(), /*Aukaistavan näkymän class*/näkymä.class);
  startActivity(intent); //Aukaise näkymä
}
```

KUVA 8. Näkymästä toiseen siirtyminen

Activity_login

Activity_login on nimensä mukaisesti sisäänkirjautumisnäkymä, jota käytetään myös rekisteröintinäkymään siirtymiseen. Näkymässä on kolme tekstikenttää, joista ylin on käyttäjätunnusta varten, sen alapuolella salasanaa varten oleva kenttä ja alhaalla rekisteröitymiskoodia varten oleva tekstikenttä. Näkymässä on

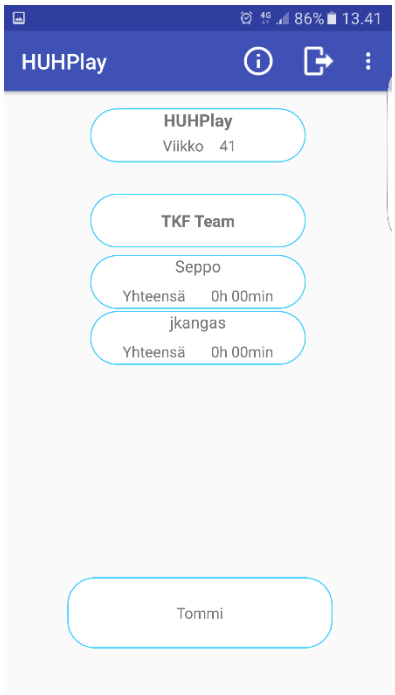
myös kaksi painiketta, joista toinen on kirjautumista ja toinen ilman koodia rekisteröintiä varten. Kuvassa 9 nähdään kirjautumisnäkymä Android-laitteessa.



KUVA 9. Kirjautumisnäkymä

Activity_main

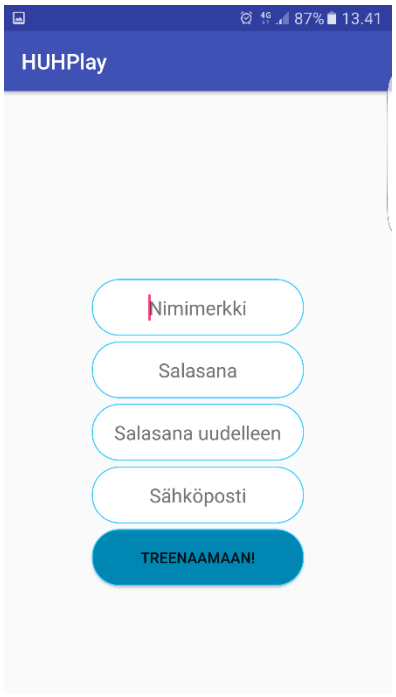
Activity_main on kirjautumisen jälkeen avautuva näkymä. Tämä näkymä on päänäkymä, josta nähdään nykyinen viikko ja seuratut tiimit ja voidaan mennä tarkkailmaan omia treenejä. Yläpalkissa löytyy painikkeet asetuksille, treenivinkeille ja uloskirjautumiselle. Tässä näkymässä voidaan klikata seurattuja joukkueita, jolloin aukeaa joukkueen jäsenien lista. Klikattaessa joukkueen jäsentä avautuu kyseisen henkilön treeninäkymä. Kuvassa 10 nähdään päänäkymä Android-laitteessa.



KUVA 10. Päänäkymä

Activity_register

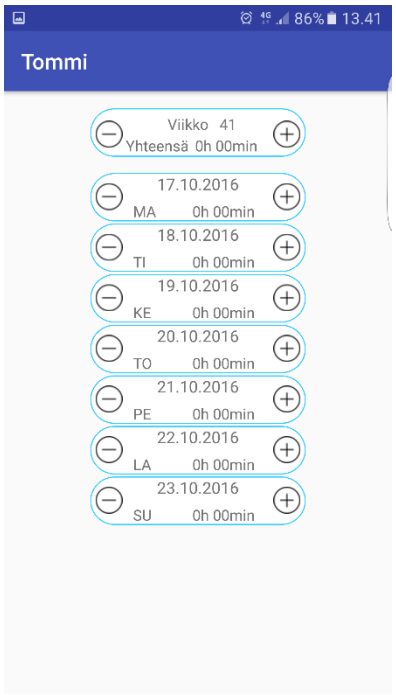
Activity_register-näkymää (kuva 11) käytetään nimensä mukaisesti rekisteröitymiseen. Näkymä on yksinkertaisuudessaan vain neljä tekstikenttää ja painike. Tekstikentät ovat käyttäjätunnusta, salasanaa, salasanan varmennusta ja sähköpostia varten. Painiketta klikkaamalla rekisteröidään käyttäjä. Mikäli rekisteröinti onnistui ilman virheitä, ohjataan käyttäjä kirjautumisnäkymään ja näytetään viesti onnistumisesta. Mikäli rekisteröinnissä tulee jokin virhe, ilmoitetaan siitä käyttäjälle. Salasanan varmennuslaatikko muuttuu punaiseksi, kun salasanat eivät ole samoja.



KUVA 11. Rekisteröitymisnäkymä

Activity_trains

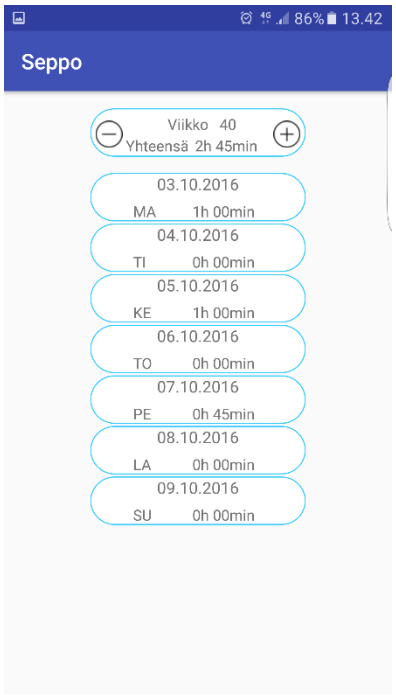
Activity_trains-näkymää käytetään vain omien treenien katsomiseen ja päivittämiseen. Näkymän ylimpänä näkyy viikko ja viikon treenien yhteisminuutit. Viikkoa voidaan selata plus- ja miinus-painikkeilla. Tämän alapuolella näkyy lista viikonpäivistä, joihin voidaan lisätä treeniminuutteja plus- ja miinus-painikkeilla. Kuvassa 12 nähdään treeninäkymä Android-laitteessa käytettynä.



KUVA 12. Harjoitusten kirjaamisnäkymä

Activity_points

Activity_points eli pistenäkymä (kuva 13) on hyvin paljon samankaltainen kuin treeninäkymä. Ainoana erona on, että tähän näkymään päästään, kun halutaan tarkastella jonkin toisen käyttäjän treenejä. Tässä näkymässä voidaan selata viikkoja, jolloin nähdään jokaisen viikon treenit, mutta treenejä ei voida muokata. Yläpalkissa lukee se käyttäjätunnus, kenen treenejä ollaan tarkastelemassa.



KUVA 13. Seurattavan henkilön harjoitukset tarkemmin

4.3 Toiminnollisuus

Android-käyttöjärjestelmän sovellusten toiminnollisuus voidaan luoda käyttäen muun muassa C++ tai Java -ohjelmointikieliä. Työssä on käytetty Java-ohjelmointikieltä sen yksinkertaisuuden ja laajan tuen vuoksi. Android-ohjelmoinnissa jokaista näkymää vastaa oma Java-luokkansa, jonka sisällä voidaan käsitellä näkymässä tapahtuvia muutoksia tai käyttäjän interaktioita. Lisäksi sovelluksessa laajennetaan Androidin omia luokkia. Laajentamisella saadaan luokkiin tuotua haluttuja lisäominaisuuksia, esimerkkinä Volley-kirjastoon uudelleenyrittämiset virheen sattuessa.

4.3.1 Näkymien luokat

Kuten jo aiemmin mainittiin, niin jokaiselle näkymälle on oma Java-luokkansa. Nämä ovat tärkeitä Android-sovelluksen toiminnan kannalta, sillä luokissa käsitellään kaikki tiedot. Sovelluksessa jokaisella näkymän luokalla on pääpiirtein sama tehtävä, hakea tiedot palvelimelta ja näyttää ne käyttäjälle. Tietojen hakeminen tapahtuu Volley-kirjastoa käyttäen. Koska palvelin palauttaa tiedot JSON-

muotoisena ja Javassa on oma kirjastonsa JSON-käsittelyyn, tapahtuu tietojen käsittely käyttäen Java-kirjastoja.

Käyttäjän klikatessa jotain painiketta näkymässä lähettää näkymä Java-koodille onClick-signaalin. Jotta Java puolella voidaan havaita tämä signaali, täytyy sovelluksen luoda signaalille kuuntelija. Yksinkertaisimmillaan kuuntelija luodaan lisäämällä XML-koodiin rivi android:onclick=”funktionnimi”. Tämän jälkeen Java-koodiin lisätään julkinen funktio samalla nimellä, joka ottaa parametrikseen View-luokan olion.

4.3.2 Muut luokat

Työssä tehtävässä sovelluksessa on myös muita kuin näkymiin sidottuja luokkia. Näitä on työssä kahdenlaisia, tietoluokkia ja valmiita luokkia laajentava luokka. Tietoluokkia on vain yksi, Users, jota käytetään palvelimelta tulevien käyttäjätietojen hallitsemiseen. Users-luokassa on jokaiselle tietokentälle oma asettaja ja lukija funktionsa. Näiden avulla Users-luokasta voidaan luoda useita olioita ja asettaa niille arvoja, joita luetaan tarvittaessa.

Sovelluksessa laajennetaan kolmea eri valmista luokkaa, BaseExpandableListAdapter, ArrayAdapter ja StringRequest. Luokkien laajentamista tarvitaan työssä, sillä kyseisiin luokkiin halutaan tuoda uusia ominaisuuksia, joita ne eivät tarjoa. BaseExpandableListAdapter-luokkaa käytetään tarjoamaan tiedot ja näkymät laajennettavaan listaan (14). Laajennettava lista on sellainen listaus, josta normaalisti näkyy vain otsikko ja sitä klikkaamalla saadaan näytettyä enemmän tietoa. Tämä on hyödyllinen tapauksissa, jossa on paljon tietoa, joka voidaan sisällyttää yhden otsikon alle. Sovelluksessa kyseisen tyylistä listaa käytetään päänäkymässä joukkueiden näyttämisessä. Työssä luokkaan ei lisätä ominaisuuksia vaan sille kerrotaan, minkälaisena sille annettu tieto näytetään käyttäjälle.

ArrayAdapter-luokkaa laajennetaan työssä kahdessa eri luokassa. ArrayAdapter-luokka on samanlainen kuin BaseExpandableListAdapter, mutta ArrayAdapteria käytetään ei laajennettavissa oleviin listoihin. Käyttötarkoitus molemmissa

luokissa on sama, kertoa vain, millaisena annettu tieto esitetään visuaalisesti käyttäjälle. Teknisesti tämän toteuttaminen molemmissa listatyypeissä on samanlainen. Luodaan LayoutInflater-luokasta olio ja käytetään juuri luotua oliota täyttämään haluttu näkymä halutulla tiedolla. Lopuksi palautetaan tiedoilla täytetty näkymä. Kuvassa 14 nähdään, miten tämä toteutuu ohjelmakoodissa.

```
@Override
public View getGroupView(int i, boolean b, View view, ViewGroup viewGroup) {
    System.out.println("getGroupView");
    String headerTitle = (String) getGroup(i);
    if(view == null) {
        LayoutInflater inflater = (LayoutInflater) this._context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
        view = inflater.inflate(R.layout.list_group, viewGroup, false);
    }

    TextView lblListHeader = (TextView) view.findViewById(R.id.lblListHeader);
    lblListHeader.setTypeface(null, Typeface.BOLD);
    lblListHeader.setText(headerTitle);
    System.out.println("Before return groupview");
    return view;
}
```

KUVA 14. Listan ulkonäkömäärittelyä

Viimeinen laajennettava luokka on StringRequest. Sovelluksessa StringRequest-luokkaa käytetään Volley-kirjaston kanssa kertomaan, mitä tehdä tiedolla, joka saadaan palvelimelta. Koska StringRequest-luokka lopettaa ajamisensa virheen sattuessa tai loppuun päästyään, on sitä laajennettava. Sovelluksessa StringRequest luokkaan lisätään uudelleenyritys, mikäli ensimmäinen tietohakuyritys epäonnistuu. Volley-kirjasto tarjoaa valmiin metodin, jolla uudelleenyritys voidaan helposti toteuttaa. Kutsutaan vain setRetryPolicy-metodia StringRequest-olion luonnin jälkeen. setRetryPolicy ottaa parametreikseen aikakatkaisuviiveen millisekunteina, uudelleenyrityksien määrän ja vetäytymiskertoimen. Kuvassa 15 nähdään, miltä ominaisuuden lisääminen toteutetaan ohjelmallisesti.

```
class MyStringRequest extends StringRequest {

    private final int timeoutms = 10000;
    private final int maxretries = 10;
    private final float backoffmultiplier = 1.0f;

    MyStringRequest(String url, Response.Listener<String> listener, Response.ErrorListener errorListener) {
        super(url, listener, errorListener);
        setRetryPolicy(new DefaultRetryPolicy(timeoutms, maxretries, backoffmultiplier));
    }

    MyStringRequest(int method, String url, Response.Listener<String> listener, Response.ErrorListener errorListener) {
        super(method, url, listener, errorListener);
        setRetryPolicy(new DefaultRetryPolicy(timeoutms, maxretries, backoffmultiplier));
    }
}
```

KUVA 15. StringRequest-luokan laajentaminen

4.4 Jatkokehittäminen

Sovellusta kehittäessä on keskitytty lähinnä toiminnollisuuden tekemiseen. Tästä syystä sovelluksen ulkoasu on todella yksinkertainen ja keskeneräinen. Nykyisellä ulkoasulla sovellusta voidaan testata, mutta ennen varsinaista julkaisua ulkoasua tulee parantaa käyttäjäystävällisemmäksi. Myös eri näkymien välisiin siirtymiin voitaisiin lisätä animaatioita, jotta sovellus näyttäisi ammattimaiselta.

Jotta sovellus olisi kaikkien saatavilla, pitää sovelluksesta tehdä versiot eri alustoille. Sovelluksen Android-version julkaisemisen jälkeen on vuorossa nettiselaimessa toimiva versio. Toiminnollisuus on valmiina muita alustoja varten, joten uusille alustoille sovelluksen kehittäminen on vain päätelaitteen ohjelmiston työstämistä. Nettiselaimessa toimiva versio on monialustainen, koska jokaisessa älypuhelimessa on jonkin asteinen nettiselain. Kuitenkin olisi hyvä, että sovelluksesta tehtäisiin myös oma sovellus iOS-laitteille. Monen sovelluksen ylläpitäminen on hieman työläämpää, mutta hyvin ohjelmoituna omat sovellukset ovat vakaampia ja suorituskykyisempiä (15).

Sovelluksesta saatiin toimiva ja toiminnollisuudesta hyvä. Täydellinen toiminnollisuus kuitenkaan ei ole, vaan siinä löytyy parannettavaa ja jatkokehitettävää. Käytetyt kirjastot ja ohjelmointikielet päivittyvät jatkuvasti ja joitain ominaisuuksia saatetaan poistaa, jolloin ohjelmistoihin tulee tehdä muutoksia. Android-

käyttöjärjestelmälle julkaistaan päivityksiä APIin, joten muutoksia täytyy tehdä myös päätepuolelle.

Tulevaisuudessa sovellukseen voidaan lisätä paljon uusia ominaisuuksia. Tällä hetkellä suunnitelmissa on muiden harjoituksien kommentointi, profiilikuvan lisääminen ja viikoittaiset harjoitteluvinkit. Suunnitelluista ominaisuuksista harjoituksien kommentointiin on palvelinohjelmistossa valmiina tuki. Tämän vuoksi kommentointimahdollisuuden lisääminen ei ole työlästä. Uusien ominaisuuksien avulla sovelluksesta saadaan aikaan ehjä kokonaisuus, jota on mukava ja helppo käyttää.

5 YHTEENVETO

Työn tarkoituksena oli tehdä urheilijoille suunnattu treenipäiväkirja-sovellus Android-käyttöjärjestelmälle. Samalla tutustuttiin Slim-viitekehykseen, PHP-ohjelmointikieleen ja Android-kehitykseen. Sovelluksen toiminnollisuus saatiin valmiiksi, joten sitä voidaan alkaa testata käytännössä. Asetetut tavoitteet saavutettiin hyvin. HUHPlayn jatkokehitysmahdollisuudet ovat laajat, sillä työssä tehtiin vain toiminnollisuus pääpiirteittään. Käyttöliittymää voidaan parantaa sekä uusia ominaisuuksia tullaan lisäämään. Yksi lisättävistä ominaisuuksista on kommentointimahdollisuus.

Android-käyttöjärjestelmälle sovelluksen kehittäminen on helposti lähestyttävää sen kattavan dokumentaation ansiota. Palvelimen puolelle HTTP-pyyntöjen käsittelijöiden tekeminen olisi enemmän aikaa vievää, mikäli ulkopuolisia viitekehyksiä ei käytettäisi. Myös muita viitekehyksiä, jotka tekevät saman asian, on olemassa, mutta työssä käytettiin Slimiä sen keveyden ja yksinkertaisuuden vuoksi.

Jatkossa HUHPlay tulee olemaan työn tilaavan urheiluseuran käytössä. Aluksi tuotetta testataan urheilijoilla. Myöhemmin sovelluksesta pyritään saamaan urheilijoille yksi varainkeruutapa ja tukijoille uusi tapa seurata urheilijoita. Tulevaisuudessa sovellusta tullaan kehittämään vielä paljon ja ulkoasuun tehdään parannuksia.

LÄHTEET

1. Kangas, Seppo 2016. Re: HUHPlay kysymyksiä. Sähköpostiviesti. Vastaanottaja: Tommi Vanhala. 30.10.2016.

2. What is social media. 2016. WebFinance Inc. Saatavissa: http://www.businessdictionary.com/definition/social-media.html. Hakupäivä 27.9.2016.

3. Dave Chaffey. 8.8.2016. Global social media research summary 2016. Saatavissa: http://www.smartinsights.com/social-media-marketing/social-media-strategy/new-global-social-media-research/. Hakupäivä 23.10.2016.

4. HTTP/1.1: Method definitions. World Wide Web Consortium. Saatavissa: https://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html. Hakupäivä 8.10.2016.

5. Slim Framework. Slim Framework Team. Saatavissa: http://www.slimframework.com/. Hakupäivä 8.10.2016.

6. Application. Slim Framework Team. Saatavissa: http://www.slimframework.com/docs/objects/application.html. Hakupäivä 12.10.2016.

7. PHP: What is PHP?. 2016. The PHP Group. Saatavissa: http://php.net/manual/en/intro-whatis.php. Hakupäivä 15.10.2016.

8. PHP: PHP Tags. 2016. The PHP Group. Saatavissa: http://php.net/manual/en/language.basic-syntax.phptags.php. Hakupäivä 15.10.2016.

9. What is MySQL?. 2016. Oracle Corporation. Saatavissa: http://dev.mysql.com/doc/refman/5.7/en/what-is-mysql.html. Hakupäivä 16.10.2016.

10. Android, the world's most popular mobile platform. Android Developers. Saatavissa: https://developer.android.com/about/android.html. Hakupäivä 22.10.2016.

11. Transmitting Network Data Using Volley. Android Developers. Saatavissa: https://developer.android.com/training/volley/index.html. Hakupäivä 16.10.2016.

12. Sending a Simple Request. Android Developers. Saatavissa: https://developer.android.com/training/volley/simple.html. Hakupäivä 16.10.2016

13. Activity. Android Developers. Saatavissa: https://developer.android.com/reference/android/app/Activity.html. Hakupäivä 16.10.2016.

14. BaseExpandableListAdapter. Android Developers. Saatavissa: https://developer.android.com/reference/android/widget/BaseExpandableListAdapter.html. Hakupäivä 16.10.2016.

15. Native Apps vs. Web Apps – What is the Better Choice. 24.08.2016. Viswanathan, Priya. Saatavissa: https://www.lifewire.com/native-apps-vs-web-apps-2373133. Hakupäivä 17.11.2016.