TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

MATTI PANULA

OPPIMISPÄIVÄKIRJA

Mobiiliohjelmointi

SISÄLTÖ

LYHENTEET JA MERKINNÄT

LGPL	GNU Lesser General Public License
NDK	Native Development Kit

1. AIKATAULU JA JOHDANTO

Tunnit		Kuvaus
10. - 20.10.2017	10h	Andoid-ohjelmointiin ja kurssin materiaaleihin tutustumista
10.10.-26.11.2017	30h	Developing Android Apps -MOOC -kurssi
20. -30.10.2017	10h	Android Studion asennus Linux-koneeseen. Hello World pyörimään emulaattoriin ja omaan Huawei Honor 8 -laitteeseen.
31.10.2017	10h	Opintopäiväkirjan aloitus. Latexiin tutustumista. Git-repon perustus
20.-25.11.2017	10h	Harjoitus 4, Laskukone
25.-26.11.2017	10h	Harjoitus 5 ja 6, Tietokanta
26.11.-27.12.2017	50h	Laaja harjoitustyö
27.-29.12.2017	5h	Opintopäiväkirjan puhtaaksikirjoitus

Aloitin kurssin asentamalla kehitysympäristöt ja seuraamalla MOOC-kurssia. Minulla ei ole aiempaa Android-ohjelmointi kokemusta joten käytin MOOC-kurssia pääsääntöisenä tiedon lähteenäni. Valitettavasti MOOC-kurssin aikaiset muistiinpanot jäivät varsin rajallisiksi ja se heijastuu myös tämän opintopäiväkirjan MOOC-harjoitusta käsittelevässä osiossa.

Ajanpuutteen vuoksi jätin Firebase- ja Kartta-harjoitukset tekemättä. Osin tähän on syynä myös se että tehtävät vaikuttivat varsin triviaaleilta ja uskon että se että keskitin aikani Androidin perusteiden opettelun jälkeen laajaan harjoitustyöhön, palveli paremmin itselleni asettamia oppimistavoitteita.

Olen toteuttanut työssäni viimeiset vajaa 3 vuotta kosketuskäyttöä tukevia sovelluksia JavaFX-ympäristössä. JavaFX:n XML-pohjainen UI-kehitys on lähellä Androidin vastaavaa ja UI-rakentaminen olikin minulle helppoa. Koska UI:n ”layout”-koodaus on minusta niin suoraviivaista, en ole tässä dokumentissa kuvannut UI-elementtien asemointia XML:llä lainkaan.

2. HARJOITUS 1, LAITTEEN KUVAUS

Valitsin kurssin kohdelaitteeksi henkilökohtaisen päivittäisessä käytössäni olevan Huawei Honor 8 -puhelimen. Siinä on Android 7 -käyttöjärjestelmä. Tarkemmat tiedot löytyvät mm. GSM-Arena sivustolta [2]. Honor 8 tukee yleisimpiä älypuhelinten ominaisuuksia kuten paikannus, kamera ja kiihtyvyysanturi.

Android-laitteiden sovelluksia kehitetään tyypillisesti JetBrainsin InteliJ Ideaan pohjautuvan Android Studio IDE:n avulla. Kehitys onnistuu Windows-, Mac- ja Linux-ympäristössä. Itse asensin kehitysympäristön Arch-Linux:iin pohjautuvaan Antergos -Linux-jakeluun. Android Studio osaa asentaa Android Developer Tools-, Android SDK- ja Android emulaattorit sisäisesti. Itse asensin ne erikseen jolloin mm. Android Debug Bridgen ja emulaattorin käyttö onnistuu tarvittaessa helposti myös ilman Android Studioa.

Android-sovelluksia kehitetään Googlen implementaatiolla Java-standardista. Vaativammat sovellukset, kuten pelit, hyödyntävät usein natiivikoodia (C,C++) Androidin Native Development Kitin (NDK) kautta käynnistettynä. Android-sovellus voidaan toteuttaa myös web-teknologioilla ja se voidaan paketoida itsenäiseksi ajettavaksi sovellukseksi. Tällöin pystytään hyödyntämään samaa koodia useilla laitealustoilla mutta usein performanssin ja ulkonäön (native look and feel) kustannuksella.

3. HARJOITUS 2, GIT

Git on minulle hyvin tuttu työstäni jo n. kymmenen vuoden ajalta.

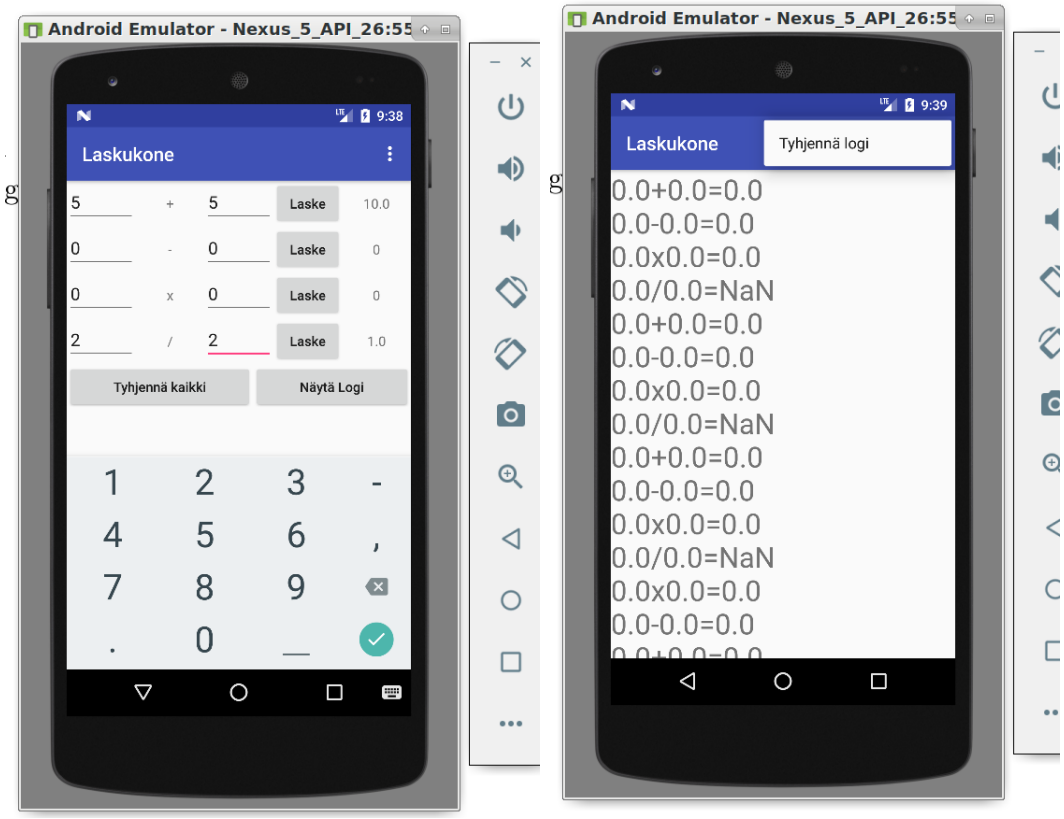
https://github.com/mpanu/tut-mobiili

4. HARJOITUS 4, PERUSKÄYTTÖLIITTYMÄ

https://github.com/mpanu/tut-mobiili/tree/master/Laskukone

Tässä tehtävässä toteutettiin kuvana annetun määritelmän mukainen laskukone.

- Laskee laskut
- Logittaa laskut
- Tyhjennä-nappi
- ”Näytä logi” -nappi avaa intentin kautta uuden activityn (skrollattava logi)
- Pää- ja logi-activityissä overflow-menu jolla tyhjätään logi (filestä ja näkyvistä)



4.1 MainActivity.java

Laskimen UI on toteutettu XML:llä. XML:n copy-paste on ihan kätevää mutta näkymien luonti suoraan javasta voisi olla tällaisissa tapauksissa joustavampaa.

MainActivityn onCreate -metodissa annetaan käsittelijät operaatio-napeille:

```
addLaskeButtonHandler((Button)findViewById(R.id.plussabutton), "+");
addLaskeButtonHandler((Button)findViewById(R.id.miinusbutton), "-");
addLaskeButtonHandler((Button)findViewById(R.id.kertobutton), "x");
addLaskeButtonHandler((Button)findViewById(R.id.jakobutton), "/");
```

Metodin addLaskeButtonHandler(Button btn, final String operation) toiminta nojaa oletukseen että syöttö- ja tulostus-komponentit ovat aina samassa järjestyksessä joten niihin voi voi viitata lapsen indeksillä.

```
LinearLayout buttonParentLayout = (LinearLayout) btn.getParent();
EditText ekaText = (EditText) buttonParentLayout.getChildAt(0);
EditText tokaText = (EditText) buttonParentLayout.getChildAt(2);
TextView tulosText = (TextView) buttonParentLayout.getChildAt(4);
```

Parametrina saatavalle napille annetaan click-käsittelijä.

```
btn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        double eka = Double.parseDouble(ekaText.getText().toString());
        double toka = Double.parseDouble(tokaText.getText().toString());
        String tulos = laske(eka,toka,operation)+"";
        tulosText.setText(tulos);
        //kirjoita logi
        FileUtils.kirjoitaTiedostoon(MainActivity.this,
                                     eka+operation+toka+"="+tulos+"\n");
    }
});
```

Login tyhjennys tapahtuu oletustyylisen MenuItemin kautta seuraavasti.

```
public boolean onOptionsItemSelected(MenuItem item) {
    FileUtils.clearLog(MainActivity.this);
    TextView tv = (TextView) findViewById(R.id.tv_display);
    if(tv != null)
        tv.setText("");
    return true;
}
```

”Tyhjennä Kaikki” -napin vaikutuspiiriin kuuluvat näkymät etsitään addLaskeButtonHandler-metodia vastaavalla tekniikalla loopissa indeksin perusteella. Napille annetaan käsittelijä seuraavasti.

```
tyhjennaBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        for(TextView clearableView : clearableViews){
            clearableView.setText("0");
        }
    }
});
```

”Näytä Logi” -nappi avaa uuden näkymän joka koostuu yhdestä TextView-komponentista. Se lukee sisältönsä tiedostosta auetessaan ja myös sisältää menun joka mahdollistaa login tyhjentämisen. Login toiminnallisuus on MathLogActivity -luokassa. Logi avataan seuraavalla tavalla.

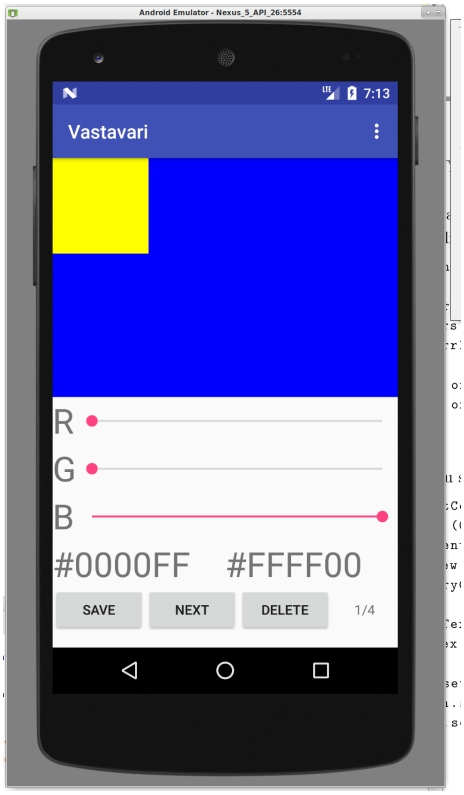
```
final Button logButton = findViewById(R.id.logi);
logButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Class destinationActivity = MathLogActivity.class;
        Intent startChildActivityIntent = new Intent(MainActivity.this,
                destinationActivity);
        startActivity(startChildActivityIntent);
    }
});
```

Tidoston käsittely on erotettu omaan FileUtils -luokkaan joka sisältää metodit kirjoitaTiedostoon, lueTiedostosta ja clearLog.

5. HARJOITUS 5 JA 6, TIETOKANTASOVELLUS

https://github.com/mpanu/tut-mobiili/tree/master/Vastavari

Tehtävänantona oli toteuttaa tietokantaa hyödyntävä sovellus. Toteutettu sovellus mahdollistaa väritietojen tallennuksen SQLite -kantaan sekä tallennettujen värien selaamisen ja poistamisen. Sovellus esittää slidereilla määritellyn värin, sen RGB-vastavärin sekä niiden hex-arvot (leikepöytä-tuella). Edellemainittujen sekä nappuloiden lisäksi on vielä kuvassa UI-elementti luku ”1/4” joka kertoo että kantaan on tallennettu 4 väriä joista ensimmäinen on valittuna.



Tietokantakäsittely kopioitiin melko suoraan kurssin AkuSovellus-esimerkistä sillä erotuksella että kantakyselyn tuloksia ei mäpätä suoraan olioiksi. Mielenkiintoinen huomio oli että Androidin developer-dokumentaatiot suosittelevat vahvasti

Room-kirjaston käyttämistä SQLite-kommunikaatioon. En tosin lähtenyt tutustumaan Roomiin syvemmin kun kyse on niin yksinkertaisesta sovelluksesta. Data Acces (DAO) ja Data Transfer Objectien (DAO) hyödyntäminen kantakommunikaatiossa on tosin minulle jo ennestään tuttua mm. Spring- ja JavaEE-ympäristöistä. Tietokantakäsittely blokkaa UI:n ja tästä tehtävästä saisi luultavasti enemmän pisteitä jos sen olisi toteuttanut asynkronisesti esim. AsyncTask:in avulla. Tietokantakommunikaatio toimii luotettavasti mutta indeksin mäppäyksessä kannan columnId-kenttään on jotain häikkää joten sovelluksessa on bugi.

Alkuperäisenä ajatuksena oli toteuttaa yleinen vastavärien etsimiseen soveltuva appi mutta minulle selvisi että kattava ratkaisu tarjoaisi useita komplementtivärejä mm. HSV- ja CMY-väriavaruuksiin sekä kontrastiin ja valoisuuteen perustuen joten tämä sovellus on vain tynkä demo. Jatkokehitysideana tähän voisi integroida kameran niin voisi poimia värimaailmoja oikean maailman olioista. Väriryhmien tallentaminen nimellä erilliseen tauluun olisi myös hyvä ominaisuus.

5.1 MainActivity.java

Jokaiselle seekBarille asetetaan kuuntelija seuraavasti. SeekBarit on alustettu tarjoamaan kokonaislukuarvoja väliltä 0-255.

```
seekBarRed.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
    public void onProgressChanged(SeekBar seekBar, int progress, boolean fro
        red = progress;
        setColors();
        clearCurrIdx();
    }
    public void onStartTrackingTouch(SeekBar seekBar) {}
    public void onStopTrackingTouch(SeekBar seekBar) {}
});
```

Värien asetus tapahtuu seuraavasti.

```
private void setColors(){
    int color = (0xff) << 24 | (red & 0xff) << 16 | (green & 0xff) << 8 | (b
    int complementaryColor = (0xff) << 24 | ((255-red) & 0xff) << 16 | ((255
    mainColorView.setBackgroundColor(color);
    complementaryColorView.setBackgroundColor(complementaryColor);

    mainHex.setText(String.format("#%06X", (0xFFFFFF & color)));
    complementHex.setText(String.format("#%06X", (0xFFFFFF & complementaryCol

    seekBarRed.setProgress(red);
    seekBarGreen.setProgress(green);
```

```
        seekBarBlue.setProgress(blue);
    }
```

Värin tallennus, tai indeksin päivitys jos väri on jo tallennettu.

```
    saveBtn = findViewById(R.id.savebtn);
    saveBtn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            int[] color = getMatchFromList();
            if(color == null) {
                datasource.open();
                datasource.store(red, green, blue);
                colorsList = datasource.findAll();
                currentIdx = colorsList.size() - 1;
            } else {
                currentIdx = colorsList.indexOf(color);
            }
            colIdTxt.setText((currentIdx + 1) + "/" + colorsList.size());
        }
    });
```

Haetaan värit kannasta, tai jos on jo haettu eikä välissä tallennettu, asetetaan näkyviin seuraava väri muistista.

```
    nextBtn = findViewById(R.id.nextbtn);
    nextBtn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            int testi = -1;
            if(colorsList == null || currentIdx == -1){
                datasource.open();
                colorsList = datasource.findAll();
                if(colorsList.size() >= 1) {
                    currentIdx = 0;
                    Log.d(TAG, "indeksi nollaan");
                }
            } else {
                incrementIdx();
            }
            if(colorsList.size() == 0) return;
            setcolorByRowId(colorsList.get(currentIdx)[3]);
            colIdTxt.setText((currentIdx + 1) + "/" + colorsList.size());

        }
    });
```

Valitun poisto kannasta columnId-kentän perusteella.

```
deleteBtn = findViewById(R.id.deletebtn);
deleteBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

        if(currentIdx == -1) return;

        datasource.open();
        datasource.delete(colorsList.get(currentIdx)[3]);
        colorsList = datasource.findAll();
        if(colorsList.size() == 0){
            clearCurrIdx();
        } else {
            setColor(colorsList.get(currentIdx));
        }
    }
});
```

5.2 ColorDataSource.java

```
public void store(int r, int g, int b){
    ContentValues values = new ContentValues();
    values.put(MySQLiteHelper.C_RED, r);
    values.put(MySQLiteHelper.C_GREEN, g);
    values.put(MySQLiteHelper.C_BLUE, b);
    database.insert(MySQLiteHelper.T_COLOR, null, values);
}

public List<int[]> findAll(){
    List<int[]> colors = new ArrayList<>();
    Cursor cursor = database.query(MySQLiteHelper.T_COLOR,
            new String[]{MySQLiteHelper.C_RED,
                         MySQLiteHelper.C_GREEN,
                         MySQLiteHelper.C_BLUE,
                         MySQLiteHelper.COLUMN_ID },
            null, null, null, null, null);
    while (cursor.moveToNext()) {
        int[] intArr = new int[4];
        intArr[0] = cursor.getInt(0);
        intArr[1] = cursor.getInt(1);
        intArr[2] = cursor.getInt(2);
        intArr[3] = cursor.getInt(3);
        colors.add(intArr);
        cursor.moveToNext();
```

```
        }
        cursor.close();
        return colors;
    }

    public void delete(int id){
        database.delete(MySQLiteHelper.T_COLOR,
                                    MySQLiteHelper.COLUMN_ID + " = " + id,
                                    null);
    }
```

6. HARJOITUS 11-19, DEVELOPING ANDROID APPS MOOC

https://eu.udacity.com/course/new-android-fundamentals–ud851

Tämä MOOC-kurssi oli pääasiallinen tiedonlähde opetellessani Android-koodausta ja käytin kurssin läpikäymiseen ja esimerkkikoodien testailuun varsin paljon aikaa. Kurssi oli hyvin toteutettu ja mielenkiintoinen. Aikomukseni oli dokumentoida tehtäviä kurssin edetessä mutta muistiinpanot jäivät varsin tyngäksi.

6.1 FavouriteToys

Opeteltiin androidin perusteet, miten xml- ja java-koodit ovat yhteydessä (xml:n viittataminen R:n kautta yms.). Tehtiin yksinkertainen tekstilista ja lisättiin se ScrollViewiin

6.2 GitHub repo

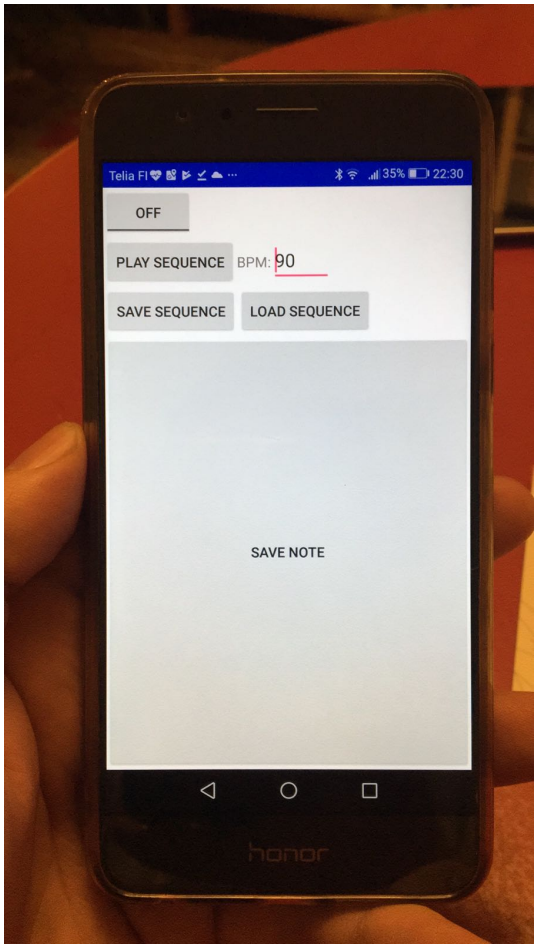
Android-platform ei salli verkkohakujen tekemistä UI-threadissä koska se saattaisi "jäädyttää"UI:n. Tässä harjoituksessa perehdyttiin verkkohaun tekemiseen AsyncTask:n avulla. Tutustuttiin myös permissioihin. AndroidManifest.xml:n tulee lisätä <uses-permission android:name="android.permission.INTERNET"/> jotta sovellus kysyy asenettaessa käyttäjältä luvan verkon käyttämiseen.

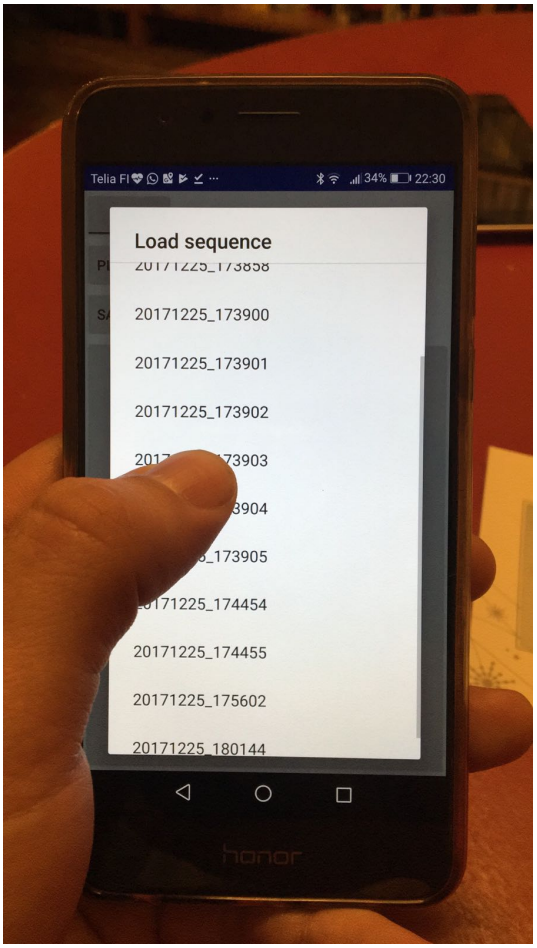
6.3 RecyclerView

RecyclerView mahdollistaa UI-komponenttien uudelleenkäyton. Tästä on hyötyä silloin kun halutaan tarjota "ikkuna"isompaan datamäärään. RecyclerView "kierrättää"UI-komponentit, joten skrollatessa paljastuvia komponentteja ei tarvitse kutsua findById:llä joka kerta kun ne ilmestyvät näkyviin.

7. LAAJA HARJOITUSTYÖ, THEREMIN-SEKVENSSERI

https://github.com/mpanu/tut-mobiili/tree/master/ThereminSeq
./app/src/main/AndroidManifest.xml
./app/src/main/res/layout/activity_main.xml
./app/src/main/java/fi/tut/matti/thereminseq/MainActivity.java
./app/src/main/java/fi/tut/matti/thereminseq/PitchBinding.java
./app/src/main/res/raw/accel_osc.csd
./app/src/main/res/raw/score_osc.csd





Tämän harjoitustyön alkuperäinen idea oli toteuttaa Theremin-tyylinen syntetisaattori jonka parametreja ohjataan puhelimen kiihtyvyysantureiden tietojen perusteella. Tavoiteeseen päästiin ja lisäominaisuutena toteutin yksinkertaisen sekvensserin joka tukee tiedostoon tallennusta. Minun Honor 8 -laitteella soittamisessa on arviolta yli 20ms viive joka on liikaa hyvän soittotuntuman saavuttamiseen. Ilmeisesti joillain Sonyn ja Googlen laitteilla olisi mahdollista saavuttaa iOS-tason audiolatenssi koska ne ilmoittavat tukevansa Androidin FEATURE_AUDIO_PRO -määritystä [3].

7.1 Käyttöliittymä

Kun sovelluksen vasemman ylälaidan toggle-napin enabloi, alkaa puhelin toistamaan ääniaaltoa jonka korkeus määräytyy puhelimen X- ja Y-akseleihin kohdistuvien kiihtyvyyksien summan perusteella. Äänenkorkeuden määrää siis käytännössä maan painovoiman suunta suhteessa puhelimeen, mutta mukana on myös puhelimen liikuttamisesta aiheutuvat kiihtyvyydet jotka toisaalta tuovat soittotuntumaan mukavan lisän. Akselit valikoituivat kokeilun kautta, X- ja Y-akseleita käytettäessä muodostuu ”soittamiseen” luonnollinen liikerata.

UI:n vapaan alueen täyttää iso ”SAVE NOTE” -nappi joka lisää senhetkisen, syntetisaattorilta kysytyn, äänenkorkeuden hertseinä List<Double> -tyyppiseen tietorakenteeseen. ”SAVE SEQUENCE”-nappi tallentaa listan arvot tiedostoon. ”LOAD SEQUENCE” -avaa AlertDialog-tyyppisen näkymän joka mahdollistaa tallennetun tiedoston lataamisen. Tiedostonnimenä on käytetty päivämäärää ja aikaa sekunnin tarkkuudella. ”PLAY SEQUENCE”-nappi soittaa muistissa olevan sekvenssin käyttäjän ”BPM”-EditText -kentän määräämällä nopeudella (Beats Per Minute).

7.2 Csound

Tämä hajoitustyö käyttää äänen tuottamiseen Csound-kirjastoa. Csound on Barry Vercoen MIT Media Labissä vuonna 1985 kehittämä musiikin ja äänen käsittelyyn keskittyvä ohjelmointiympäristö [1]. Sitä kehitetään aktiivisesti avoimen lähdekoodin periaattella ja se on lisensoitu GNU Lesser General Public License (LGPL) -lisenssillä joka mahdollistaa sen linkittämisen myös kaupalliseen suljetun lähdekoodin sovellukseen. Csound on käännetty useimmille laitealustoille ja siitä löytyy valmiiksi käännetty versio myös ARM-pohjaiselle Androidille. Android-käännös tarjoaa valmiina myös Androidin Native Development Kit (NDK) -rajapinnalla toteututetut ”koukut” yleisimpiin Csound-API:n ominaisuuksiin.

Csound on kokoelma C/C++ -kielillä toteutettuja syntetisaattorimoduuleja sekä ympäristö joka mahdollistaa moduuleiden kytkemisen toisiinsa sekä niiden parametrien manipuloinnin aikajanalla. Csound ”ohjelmat” ovat csd-päätteisiä tekstitiedostoja joka koostuvat instrumentteja sisältävästä orchestra-osiosta ja score-osiosta joka säätää instrumenttien parametreja aikajanalla (vrt. soittimet ja soittajat). Perinteinen käyttötapaus Csoundilla on ollut että muusikko ohjelmoi kappaleen csd-tiedostoon ja ”kääntää” tai renderöi sen Csound-kääntäjällä wav-tiedostoksi. Nykykoneet pystyvät toistamaan isojakin orkestereja reaaliajassa ja uskon että nykyään yleisin käyttötapaus on ohjata Csoundin ”käännös” suoraan äänikortille tai esim. raituriohjelmalle ja tarvittaessa manipuloida parametreja reaaliajassa.

7.3 Csoundin käyttö harjoitustyössä

Tämä sovellus hyödyntää kahta eri csd-tiedostoa (Csound-ohjelmaa). Csound käynnistetään ajamaan accel_osc.csd -tiedoston määritelemää ohjelmaa kun vasemman yläkulman toggle asetetaan on-tilaan. Tiedostossa accel_osc.csd on määritetty että instrumentti 1 soi voimakkuudella 0 360000 sekuntia. Tämä mahdollistaa sen että Csound ”jää kuuntelemaan” kiihtyvyysantureilta ”software bus”-väylän kautta tulevia tietoja. Csound sammuu oletuksena jos kaikki tiedossa olevat score-eventit on käsitelty. NDK-kirjastoja hyödyntäessä pitää olla tarkkana ettei tehdä turhaa laskentaa joka kuluttaa akkua joten kaikki UI:n muut nappulat sulkevat tämän Csound-instanssin.

Toinen Csound-ohjelma score_osc.csd jää vastaavasti kuuntelemaan 360000 sekunnin ajaksi, mutta suoraan instrumenttiin kytketyn ”reaaliaikaisen” väylän asemesta syötetäänkin Csoundille score-tietoa. Kun ”PLAY SEQUENCE” -nappia painetaan, generoidaan muistissa olevista pitch-arvoista score siten että ensimmäisen nuotin alkuaika on nolla ja seuraavien alkuaika lasketaan BPM-arvon perusteella. Tämä mahdollistaa sen että csound-pystyy hoitamaan nuottien ajoituksen tarkasti eikä Android-puolella tarvitse toteuttaa nuottien ajoituksia.

Csound-Andoid-kirjasto tarjoaa CsoundAccelerometerBinding -oletustoteutuksen kiihtyvyysanturien hyödyntämiseen. Se hyödyntää Androidin TYPE_ACCELEROMETER-rajapinnan kautta saatavia tietoja. Vastaava toteutus olisi helposti tehtävissä myös TYPE_GRAVITY-rajapinnan anturitiedoille joissa on mukana ainoastaan painovoima eikä muita kiihtyvyyksiä ja joka siten sopipisi mahdollisesti paremmin tarkkaan melodiseen soittamiseen.

CsoundAccelerometerBinding ottaa sensorit käyttöön seuraavalla tavalla.

```
sensorManager = (SensorManager) context.
        getSystemService(Context.SENSOR_SERVICE);
    List<Sensor> sensors = sensorManager.getSensorList(
        Sensor.TYPE_ACCELEROMETER);
```

CsoundAccelerometerBinding-luokka implementoi Androidin SensorEventListener -rajapinnan ja se kytketään SensorManageriin seuraavasti.

```
sensorManager.registerListener(this, sensor,
    SensorManager.SENSOR_DELAY_FASTEST);
```

7.4 MainActivity.java

Soitto-toiminta käynnistyy vasemman yläkulman toggle-buttonilla. Resursseista luetaan accel_osc.csd -tiedosto ja se tallennetaan cache-hakemistoon jotta Csound pystyy lukemaan sen. Se myös pitää näytön päällä soiton ajan jotta nuottien tallennus soiton yhteydessä on mahdollista. Koodissa näkyvä CsoundMotion on Csoundin mukana tuleva wrapper-luokka yllämainitulle CsoundAccelerometerBinding -luokalle joka välittää kiihtyvyysanturin tiedot Csoundille.

```
startStopButton = findViewById(R.id.mute);
startStopButton.setOnCheckedChangeListener(
        new CompoundButton.OnCheckedChangeListener() {
    public void onCheckedChanged(CompoundButton buttonView,
                                 boolean isChecked) {
        if (isChecked) {
            String csd = getResourceFileAsString(R.raw.accel_osc);
            File f = createTempFile(csd);
            CsoundMotion csoundMotion = new CsoundMotion(csoundObj);
            csoundMotion.enableAccelerometer(MainActivity.this);
            csoundObj.startCsound(f);
            keepScreenOn(true);
        } else {
            csoundObj.stop();
            keepScreenOn(false);
        }
    }
});
```

Play-nappi sammuttaa soittotilan kytkemällä ON-OFF-nappulan offille ja käynnistää uuden Csound-instanssin score_osc.csd-tiedostolla. Score generoidaan BPM-arvon ja listaan tallennettujen äänenkorkeuksien perusteella ja lähetetään Csoundille soitettavaksi sendScore-metodin avulla. Generoitu score esitetään käyttäjälle

Toast-dialogin avulla.

```
playBtn = findViewById(R.id.playBtn);
playBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        if(startStopButton.isChecked()){
            startStopButton.setChecked(false);
        }

        try {
            bpm = Double.parseDouble(bpmTxt.getText().toString());
        } catch (NumberFormatException e) {
            bpmTxt.setText(bpm + "");
        }

        String csd = getResourceFileAsString(R.raw.score_osc);
        File f = createTempFile(csd);
        csoundObj.startCsound(f);

        String score = "";
        double beatLength = 15.0 / (double)bpm; // 4 steps per beat
        int i = 0;
        for(double pitchD : seqList){
            String pitchStr = String.format(Locale.US,"%.2f", pitchD);
            // instNbr startBeat durBeats ... instParams=pitch
            score += "i 1 " + i * beatLength + " "
                    + (beatLength+0.1) + " " + pitchStr + "\n";
            i++;
        }
        csoundObj.sendScore(score);
        Toast.makeText(MainActivity.this, score, Toast.LENGTH_LONG).show();

    }
});
```

Aktiivisen nuotin tallennukseen käytetty ”SAVE NOTE” -nappi lukee PitchBinding -luokasta double-tyyppisen arvon. Kyseinen luokka toteuttaa CsoundBinding -rajapinnan ja se on kytketty Csoundiin CsoundObj-luokan addBinding-metodin avulla. PitchBinding on itse toteutettu luokka josta löytyy seuraava metodi jota Csound kutsuu säännöllisesti. Csoundilta pyydetään ”pitch”-niminen arvo joka tallennetaan julkiseen ”value”-nimiseen double-kenttään.

```
public void updateValuesFromCsound() {
    Csound csound = csoundObj.getCsound();
    value = csound.GetChannel("pitch");
}
```

Vastaavasti accel_osc.csd tiedostosta löytyy seuraava rivi joka kertoo että äänenkorkeutta ohjaavan kpch-muuttujan arvoa tarjoillaan ”software bus”-väylään ”pitch” nimellä.

```
chnset kpch, "pitch"
```

7.5 Johtopäätökset

Csound on varsin varteenotettava vaihtoehto Android-musiikkisovellusten toteuttamiseen. Kehittäjälle joka tuntee analogisynteesin yleisimmät rakennuspalikat on Csoundin hyödyntäminen, Csoundin vanhahkon tyylisestä syntaksista ja muusta historian painolastista huolimatta, kohtuullisen vaivatonta. Itse kulutin paljon aikaa siihen että käänsin Csound-Androidin itse ARM-arkkitehtuurille ja vasta jälkeenpäin huomasin että ladattavana olisi ollut myös valmis käännös. Csound-Android-projektin importtaaminen omaan Andoid-projektiin onnistui vaivattomasti ja esimerkkisovelluksia tutkimalla sai äänen soimaan omassa laitteessa varsin vaivattomasti. Koska Csound-Android oli käännetty ARM-arkkitehtuurille ei x86-emulaattoria voinut käyttää testaamiseen. Tämä lisäsi tiettyä vaivalloisuutta kehitystyöhön. Jos Csound-Androidilla tekisi vakavaa kehitystä olisi syytä harkita sen kääntämistä x86-arkkitehtuurille jotta se toimisi myös emulaattorissa.

Alkuoletukseni oli että Androidilla ei pysty toteuttamaan vakavasti otettavia musiikkisovelluksia koska Andoid-laitteet eivät tue lyhyitä audiolatensseja. Projekti antoi oletukselle vahvistusta, mutta projektin aikana löysin myös viitteitä siitä että on olemassa Android-laitteita jotka mahdollisesti kykenevät ammattitason audioon. Silti, verrattuna iOS-ekosysteemin, pro-audio Androidilla on merkityksetön markkina joka ei ole kehittäjälle houkutteleva. Android-ekosysteemi on audiokehityksen kannalta ongelmallinen myös siksi että oletuskieli Java ei virtuaalisen luonteensa vuoksi sovellu reaaliaikasovelluksiin. Oletan että Android-pelitkin kehitetään pääsääntöisesti jollain natiivikielellä kuten C++. Tässä harjoitustyössä kiihtyvyysanturin tieto reititetaan javan kautta Csoundille. Teknisesti olisi mahdollista välittää anturitiedot suoraan natiivirajapinnan kautta ja ohittaa Java. Tämä tuskin onnistuisi ilman C/C++-koodausta mutta olisi mielenkiintoista tietää onko sillä merkitystä latenssiin.

LÄHTEET

[1] csounds.com. csounds.com. [Online]. Available: csounds.com

[2] GSMArena. Honor 8 - full phone specifications. [Online]. Available: https://www.gsmarena.com/huawei_honor_8-8195.php

[3] juce.com forum. juce.com forum. [Online]. Available: https://forum.juce.com/t/list-of-android-pro-audio-devices/21165/3