# MainActivity.java

package ca.brocku.imajadio.app;

import android.app.Activity;

import android.content.Intent;

import android.database.Cursor;

import android.graphics.Bitmap;

import android.graphics.BitmapFactory;

import android.graphics.Matrix;

import android.media.AudioTrack;

import android.media.ExifInterface;

import android.net.Uri;

import android.os.AsyncTask;

import android.os.Bundle;

import android.os.Environment;

import android.os.Handler;

import android.os.Message;

import android.provider.MediaStore;

import android.util.Log;

import android.view.Menu;

import android.view.MenuItem;

import android.view.View;

import android.widget.Button;

import android.widget.ImageView;

import android.widget.ProgressBar;

import android.widget.SeekBar;

import android.widget.TextView;

import android.widget.Toast;

import java.io.File;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.OutputStream;

import java.text.SimpleDateFormat;

import java.util.Arrays;

import java.util.Comparator;

import java.util.Date;

import java.util.Locale;

import imajadio.Imajadio;

import imajadio.WaveHeader;

public class MainActivity extends Activity {

Imajadio imajadio;

// Activity request codes

private static final int CAMERA\_CAPTURE\_IMAGE\_REQUEST\_CODE = 100;

public static final int MEDIA\_TYPE\_IMAGE = 1;

private static int RESULT\_LOAD\_IMAGE = 2;

// directory name to store captured images

private static final String IMAGE\_DIRECTORY\_NAME = "Imajadio";

private Uri fileUri; // file url to store image

private ImageView imgPreview;

private ImageView loadingBlack;

private Button playButton;

private View progressBar;

private SeekBar grainDurationSeekBar;

private TextView progressText;

private TextView convertingText;

private ProgressBar loadingSpinner;

private float grainDuration;

private float realGrainDuration;

private boolean readyToPlay = false;

private boolean isPlaying = false;

private AudioTrack.OnPlaybackPositionUpdateListener listener;

public final static String APP\_PATH\_SD\_CARD = "/Imajadio/"; //directory to store images

private Bitmap image; //the image to be converted

private StatusBarThread t;

private Handler handie = new Handler() {

@Override

public void handleMessage(Message msg) {

Bundle b = msg.getData();

int position = b.getInt("PROGRESS\_POSITION");

if (position == -1) {

progressBar.setVisibility(progressBar.getVisibility() == View.GONE ? View.VISIBLE : View.GONE);

} else {

progressBar.setX(position);

}

}

};

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

BitmapFactory.Options options = new BitmapFactory.Options();

options.inDensity = getResources().getDisplayMetrics().densityDpi;

image = BitmapFactory.decodeResource(this.getResources(), R.drawable.keys, options);

progressBar = (View) findViewById(R.id.progressBar);

grainDurationSeekBar = (SeekBar) findViewById(R.id.seekBarGrainDuration);

grainDurationSeekBar.setOnSeekBarChangeListener(new SeekBarHandler());

grainDuration = 0.025f;

progressText = (TextView) findViewById(R.id.progressText);

convertingText = (TextView) findViewById(R.id.convertingText);

imgPreview = (ImageView) findViewById(R.id.imgPreview);

imgPreview.setImageBitmap(image);

playButton = (Button) findViewById(R.id.playButton);

playButton.setOnClickListener(new PlayButtonHandler());

loadingSpinner = (ProgressBar) findViewById(R.id.progressSpinner);

loadingBlack = (ImageView) findViewById(R.id.loadingBlack);

playButton.setText("CONVERT");

loadingSpinner.setVisibility(View.GONE);

convertingText.setVisibility(View.GONE);

loadingBlack.setVisibility(View.GONE);

listener = new AudioTrack.OnPlaybackPositionUpdateListener() {

@Override

public void onMarkerReached(AudioTrack audioTrack) {

playButton.setText("PLAY");

grainDurationSeekBar.setEnabled(true);

}

@Override

public void onPeriodicNotification(AudioTrack audioTrack) {}

};

}

@Override

protected void onActivityResult(int requestCode, int resultCode, Intent data) {

// if the result is capturing Image

if (requestCode == CAMERA\_CAPTURE\_IMAGE\_REQUEST\_CODE) {

if (resultCode == RESULT\_OK) {

// successfully captured the image

// display it in image view

previewCapturedImage();

//deletes the file after it has been displayed

//this is the uncropped image getting deleted

File file = new File(fileUri.getPath());

boolean deleted = file.delete();

readyToPlay = false;

playButton.setText("CONVERT");

} else if (resultCode == RESULT\_CANCELED) {

// user cancelled Image capture

} else {

// failed to capture image

}

} else if (requestCode == RESULT\_LOAD\_IMAGE && resultCode == RESULT\_OK && null != data) {

//This is to LOAD IMAGE FROM SDCARD

Uri selectedImage = data.getData();

String[] filePathColumn = {MediaStore.Images.Media.DATA};

Cursor cursor = getContentResolver().query(selectedImage,

filePathColumn, null, null, null);

cursor.moveToFirst();

int columnIndex = cursor.getColumnIndex(filePathColumn[0]);

String picturePath = cursor.getString(columnIndex); //contains the path of the selected image

cursor.close();

BitmapFactory.Options options = new BitmapFactory.Options();

options.inDensity = getResources().getDisplayMetrics().densityDpi;

image = BitmapFactory.decodeFile(picturePath, options);

//Bitmap test = BitmapFactory.decodeResource(this.getResources(), R.drawable.decreasing\_freq, options); //used to test images in drawable/

imgPreview.setImageBitmap(image);

// image = ((BitmapDrawable)imgPreview.getDrawable()).getBitmap();

imgPreview.setDrawingCacheEnabled(false);

imgPreview.setDrawingCacheEnabled(true);

image = imgPreview.getDrawingCache();

// image=((BitmapDrawable)imgPreview.getDrawable()).getBitmap();

Log.e("impreveiw height", String.valueOf(imgPreview.getHeight()));

Log.e("impreveiw weight", String.valueOf(imgPreview.getWidth()));

readyToPlay = false;

playButton.setText("CONVERT");

}

}//onActivityResult

/\*\*

\* Here we store the file url as it will be null after returning from camera

\* app

\*/

@Override

protected void onSaveInstanceState(Bundle outState) {

super.onSaveInstanceState(outState);

// save file url in bundle as it will be null on screen orientation changes

outState.putParcelable("file\_uri", fileUri);

}

/\*\*

\* Restore the fileUri again

\*/

@Override

protected void onRestoreInstanceState(Bundle savedInstanceState) {

super.onRestoreInstanceState(savedInstanceState);

// get the file url

fileUri = savedInstanceState.getParcelable("file\_uri");

}

@Override

public boolean onCreateOptionsMenu(Menu menu) {

// Inflate the menu; this adds items to the action bar if it is present.

getMenuInflater().inflate(R.menu.main, menu);

return true;

}

@Override

public boolean onOptionsItemSelected(MenuItem item) {

// Handle action bar item clicks here. The action bar will

// automatically handle clicks on the Home/Up button, so long

// as you specify a parent activity in AndroidManifest.xml.

int id = item.getItemId();

switch (id) {

case R.id.action\_exportAudio:

try {

saveWav();

} catch (FileNotFoundException e) {

e.printStackTrace();

}

return true;

case R.id.action\_newImage:

captureImage();

return true;

case R.id.action\_loadImage:

//load image from sdcard

Intent i = new Intent(Intent.ACTION\_PICK, android.provider.MediaStore.Images.Media.EXTERNAL\_CONTENT\_URI);

startActivityForResult(i, RESULT\_LOAD\_IMAGE);

return true;

case R.id.action\_saveImage:

//export image to sdcard

imgPreview.setDrawingCacheEnabled(false);

imgPreview.setDrawingCacheEnabled(true);

Bitmap bitmap = imgPreview.getDrawingCache();

saveImageToExternalStorage(bitmap);

deleteLastFromDCIM();

return true;

case R.id.action\_exit:

Intent intent = new Intent(Intent.ACTION\_MAIN);

intent.addCategory(Intent.CATEGORY\_HOME);

intent.setFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK);

startActivity(intent);

return true;

}

return super.onOptionsItemSelected(item);

}//onOptionsItemSelected

public boolean saveImageToExternalStorage(Bitmap image) {

String fullPath = Environment.getExternalStorageDirectory().getAbsolutePath() + APP\_PATH\_SD\_CARD;

String timeStamp = new SimpleDateFormat("yyyyMMdd\_HHmmss", Locale.getDefault()).format(new Date());

try {

File dir = new File(fullPath);

if (!dir.exists()) {

dir.mkdirs();

}

OutputStream fOut = null;

File file = new File(fullPath, "IMG\_" + timeStamp + ".png");

file.createNewFile();

fOut = new FileOutputStream(file);

// 100 means no compression, the lower you go, the stronger the compression

image.compress(Bitmap.CompressFormat.PNG, 100, fOut);

fOut.flush();

fOut.close();

//This is important. It is used to let the scanner know a new file was added.

sendBroadcast(new Intent(Intent.ACTION\_MEDIA\_SCANNER\_SCAN\_FILE, Uri.fromFile(file)));

MediaStore.Images.Media.insertImage(getApplicationContext().getContentResolver(), file.getAbsolutePath(), file.getName(), file.getName());

Toast.makeText(getApplicationContext(), "Saved to Imajadio folder", Toast.LENGTH\_SHORT).show();

return true;

} catch (Exception e) {

Log.e("saveToExternalStorage()", e.getMessage());

return false;

}

}//saveImageToExternalStorage

//Start Camera Helper Methods

/\*

\* Capturing Camera Image will launch camera app request image capture

\*/

private void captureImage() {

Intent intent = new Intent(MediaStore.ACTION\_IMAGE\_CAPTURE);

fileUri = getOutputMediaFileUri(MEDIA\_TYPE\_IMAGE);

intent.putExtra(MediaStore.EXTRA\_OUTPUT, fileUri);

// start the image capture Intent

startActivityForResult(intent, CAMERA\_CAPTURE\_IMAGE\_REQUEST\_CODE);

}

/\*\*

\* Receiving activity result method will be called after closing the camera

\*/

/\*

\* Display image from a path to ImageView

\*/

private void previewCapturedImage() {

try {

imgPreview.setVisibility(View.VISIBLE);

// bitmap factory

BitmapFactory.Options options = new BitmapFactory.Options();

Bitmap bitmap = BitmapFactory.decodeFile(fileUri.getPath(),

options);

//fix for orientation

ExifInterface exif = new ExifInterface(fileUri.getPath());

int orientation = exif.getAttributeInt(ExifInterface.TAG\_ORIENTATION, 1);

Log.e("ORITENTATION", String.valueOf(orientation));

Matrix matrix = new Matrix();

if (orientation == 8) {

matrix.postRotate(270);

} else if (orientation == 3) {

matrix.postRotate(180);

} else if (orientation == 6) {

matrix.postRotate(90);

}

bitmap = Bitmap.createBitmap(bitmap, 0, 0, bitmap.getWidth(), bitmap.getHeight(), matrix, true);

//end fix for orientation

//fix for huge resolution

if (bitmap.getWidth() > 4096 || bitmap.getHeight() > 4096) {

bitmap = Bitmap.createScaledBitmap(bitmap, (int) (bitmap.getWidth() \* .5), (int) (bitmap.getHeight() \* .5), false);

}

//end fix for huge resolution

imgPreview.setImageBitmap(bitmap);

imgPreview.setDrawingCacheEnabled(false);

imgPreview.setDrawingCacheEnabled(true);

image = imgPreview.getDrawingCache();

//image = ((BitmapDrawable)imgPreview.getDrawable()).getBitmap();

} catch (NullPointerException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

}

/\*\*

\* Creating file uri to store image/video

\*/

public Uri getOutputMediaFileUri(int type) {

return Uri.fromFile(getOutputMediaFile(type));

}//getOutputMediaFileUri

/\*

\* returning image / video

\*/

private static File getOutputMediaFile(int type) {

// External sdcard location

File mediaStorageDir = new File(

Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY\_PICTURES),

IMAGE\_DIRECTORY\_NAME);

// Create the storage directory if it does not exist

if (!mediaStorageDir.exists()) {

if (!mediaStorageDir.mkdirs()) {

Log.d(IMAGE\_DIRECTORY\_NAME, "Oops! Failed create "

+ IMAGE\_DIRECTORY\_NAME + " directory");

return null;

}

}

// Create a media file name

String timeStamp = new SimpleDateFormat("yyyyMMdd\_HHmmss",

Locale.getDefault()).format(new Date());

File mediaFile;

if (type == MEDIA\_TYPE\_IMAGE) {

mediaFile = new File(mediaStorageDir.getPath() + File.separator

+ "IMG\_" + timeStamp + ".jpg");

} else {

return null;

}

return mediaFile;

}

//End Camera Helper Methods

private void deleteLastFromDCIM() {

File f = new File(Environment.getExternalStorageDirectory() + "/DCIM/Camera");

Log.i("Log", "file name in delete folder : " + f.toString());

File[] files = f.listFiles();

//Log.i("Log", "List of files is: " +files.toString());

Arrays.sort(files, new Comparator<Object>() {

public int compare(Object o1, Object o2) {

if (((File) o1).lastModified() > ((File) o2).lastModified()) {

// Log.i("Log", "Going -1");

return -1;

} else if (((File) o1).lastModified() < ((File) o2).lastModified()) {

// Log.i("Log", "Going +1");

return 1;

} else {

// Log.i("Log", "Going 0");

return 0;

}

}

});

//Log.i("Log", "Count of the FILES AFTER DELETING ::"+files[0].length());

files[0].delete();

}//deleteLastFromDCIM

public boolean saveWav() throws FileNotFoundException {

if (readyToPlay) {

String timeStamp = new SimpleDateFormat("yyyyMMdd\_HHmmss",

Locale.getDefault()).format(new Date());

String fullPath = Environment.getExternalStorageDirectory().getAbsolutePath() + "/Imajadio/";

try {

File dir = new File(fullPath);

if (!dir.exists()) {

dir.mkdirs();

}

} catch (Exception e) {

Log.e("saveWav()", e.getMessage());

return false;

}

FileOutputStream all = new FileOutputStream(fullPath + "IMAJADIO\_" + timeStamp + ".wav");

//This creates the header for the wav file.

WaveHeader w = new WaveHeader((short) 1, imajadio.getAudioChannelCount(), imajadio.getAudioSampleRate(), (short) 16, imajadio.getDATA().length);

Log.e("HEADERINFO", w.toString());

try {

//write header

w.write(all);

//write the data

all.write(imajadio.getDATA());

all.close();

Toast.makeText(getApplicationContext(), "Saved audio to Imajadio folder", Toast.LENGTH\_SHORT).show();

} catch (IOException e) {

e.printStackTrace();

}

return true;

}

Toast.makeText(getApplicationContext(), "Audio NOT converted yet", Toast.LENGTH\_SHORT).show();

return false;

}//save

public void onUpdateProgressBar(int i) {

Message message = new Message();

Bundle b = new Bundle();

b.putInt("PROGRESS\_POSITION", i);

message.setData(b);

handie.sendMessage(message);

}

private class PlayButtonHandler implements View.OnClickListener {

@Override

public void onClick(View view) {

Button b = (Button) view;

String buttonText = b.getText().toString();

if (buttonText.equals("PLAY")) {

if (readyToPlay == true) {

imajadio.play();

playButton.setText("STOP");

grainDurationSeekBar.setEnabled(false);

t = new StatusBarThread(isPlaying, MainActivity.this, imgPreview, realGrainDuration);

t.start();

}

} else if (buttonText.equals("CONVERT")) {

new AsyncTaskImajadio().execute();

} else if (buttonText.equals("STOP")) {

t.requestStop();

imajadio.stop();

playButton.setText("PLAY");

grainDurationSeekBar.setEnabled(true);

}

}

}//PlayButtonHandler

private class SeekBarHandler implements SeekBar.OnSeekBarChangeListener {

@Override

public void onProgressChanged(SeekBar seekBar, int progress, boolean b) {

grainDuration = (float) ((progress + 1) \* .001);

progressText.setText(String.valueOf((int) (grainDuration \* 1000)) + " ms");

readyToPlay = false;

playButton.setText("CONVERT");

}

@Override

public void onStartTrackingTouch(SeekBar seekBar) {

}

@Override

public void onStopTrackingTouch(SeekBar seekBar) {

}

}

private class AsyncTaskImajadio extends AsyncTask<Void, Void, Void> {

@Override

protected void onPreExecute() {

realGrainDuration = grainDuration;

loadingSpinner.setVisibility(View.VISIBLE);

convertingText.setVisibility(View.VISIBLE);

loadingBlack.setVisibility(View.VISIBLE);

playButton.setEnabled(false);

grainDurationSeekBar.setEnabled(false);

}

@Override

protected Void doInBackground(Void... voids) {

imajadio = new Imajadio(image, 16, grainDuration);

Log.e("IMAGE DIMENS (H/W)", image.getHeight() + "; " + image.getWidth());

imajadio.bitmapToAudio();

imajadio.onPlaybackStopped(listener);

imajadio.normalizeAudio();

readyToPlay = true;

return null;

}

@Override

protected void onPostExecute(Void result) {

playButton.setText("PLAY");

loadingSpinner.setVisibility(View.GONE);

convertingText.setVisibility(View.GONE);

loadingBlack.setVisibility(View.GONE);

playButton.setEnabled(true);

grainDurationSeekBar.setEnabled(true);

}

}//AsyncTaskImajadio

}

# StatusBarThread.java

package ca.brocku.imajadio.app;

import android.widget.ImageView;

public class StatusBarThread extends Thread {

public boolean shouldContinue = true;

boolean isPlaying;

MainActivity activity;

ImageView imgPreview;

float realGrainDuration;

StatusBarThread(boolean isPlaying, MainActivity activity, ImageView imgPreview, float realGrainDuration) {

this.isPlaying = isPlaying;

this.activity = activity;

this.imgPreview = imgPreview;

this.realGrainDuration = realGrainDuration;

}

@Override

public void run() {

isPlaying = true;

activity.onUpdateProgressBar(-1);

for (int i = 0; i < imgPreview.getWidth() && shouldContinue; i++) {

try {

activity.onUpdateProgressBar(i);

Thread.sleep((long) (realGrainDuration \* 1000));

} catch (InterruptedException e) {

e.printStackTrace();

}

}

activity.onUpdateProgressBar(-1);

isPlaying = true;

}

public void requestStop() {

shouldContinue = false;

}

}

# Imajadio.java

package imajadio;

import android.graphics.Bitmap;

import android.graphics.Color;

import android.media.AudioFormat;

import android.media.AudioManager;

import android.media.AudioTrack;

import android.util.Log;

public class Imajadio {

private final int NUMBER\_OF\_FREQUENCIES = 4000 - 100; //frequency range from 100-10000 Hz

private final int SAMPLE\_RATE = 8000;

private final Bitmap IMAGE;

private final int IMAGE\_HEIGHT;

private final int IMAGE\_WIDTH;

private final int MAX\_AMPLITUDE;

private byte[] DATA;

private double highestAmplitude;

private double grainDuration;

private AudioTrack audio; //contains the audio to play

public Imajadio(Bitmap image) {

this(image, 16, 1);

}

public Imajadio(Bitmap image, int bitDepth) {

this(image, bitDepth, 1);

}

public Imajadio(Bitmap image, int bitDepth, double grainDuration) {

this.IMAGE = image;

this.IMAGE\_HEIGHT = IMAGE.getHeight();

this.IMAGE\_WIDTH = IMAGE.getWidth();

this.MAX\_AMPLITUDE = (int) Math.pow(2, bitDepth - 1); //max amplitude is based on bit depth

this.grainDuration = grainDuration;

this.highestAmplitude = 0;

}

public double getGrainDuration() {

return grainDuration;

}

public void setGrainDuration(double grainDuration) {

this.grainDuration = grainDuration;

}

/\*\*

\* Converts the entire bitmap image to audio.

\*

\* It writes out the samples to the AudioTrack so that the returned AudioTrack just needs to be

\* played.

\*

\* @return AudioTrack an instance which has the audio written out to it

\*/

public void bitmapToAudio() {

double[] samples = new double[(int) (grainDuration \* SAMPLE\_RATE)];

byte[] generatedSnd = new byte[IMAGE\_WIDTH \* 2 \* samples.length];

int idx = 0;

audio = new AudioTrack(AudioManager.STREAM\_MUSIC,

SAMPLE\_RATE, AudioFormat.CHANNEL\_OUT\_MONO,

AudioFormat.ENCODING\_PCM\_16BIT, 2 \* samples.length \* IMAGE\_WIDTH,

AudioTrack.MODE\_STATIC);

for (int column = 1; column <= IMAGE\_WIDTH; column++) { //for each column in the image

//Convert a column to samples

int[] pixels = new int[IMAGE\_HEIGHT];

IMAGE.getPixels(pixels, 0, 1, column - 1, 0, 1, IMAGE\_HEIGHT); //extract a column of pixels

Log.e("COLUMN #", String.valueOf(column));

samples = columnToSamples(pixels, column - 1);

// convert to 16 bit pcm sound array

// assumes the sample buffer is normalised.

for (double dVal : samples) {

short val = (short) ((dVal));

// in 16 bit wav PCM, first byte is the low order byte

generatedSnd[idx++] = (byte) (val & 0x00ff);

generatedSnd[idx++] = (byte) ((val & 0xff00) >>> 8);

}

DATA = generatedSnd;

}

audio.write(generatedSnd, 0, generatedSnd.length);

}

private double[] columnToSamples(int[] column, int columnIndex) {

Harmonic[] harmonics = new Harmonic[column.length];

//converts each pixel in a column to its associated harmonic

for (int verticalOffset = 0; verticalOffset < column.length; verticalOffset++) {

harmonics[verticalOffset] = pixelToHarmonic(column[verticalOffset], verticalOffset);

//Log.e("Harm #"+String.valueOf(verticalOffset), "freq: "+ String.valueOf(harmonics[verticalOffset].getFrequency()) + "; amp: " + String.valueOf(harmonics[verticalOffset].getAmplitude()));

}

int numSamples = (int) (grainDuration \* SAMPLE\_RATE);

double[] samples = new double[numSamples];

double amplitude;

for (int sampleIndex = 0; sampleIndex < numSamples; sampleIndex++) { //for each sample in the output table

amplitude = 0;

int skipper = 0;

for (Harmonic h : harmonics) { //add the amplitude of each harmonic

if(skipper%20 == 0 && h.getAmplitude() != 0) {

double samplesPerPeriod = SAMPLE\_RATE/h.getFrequency();

amplitude += h.getAmplitude() \* Math.sin(2 \* Math.PI \* (numSamples\*(columnIndex-1)+sampleIndex)/samplesPerPeriod);

//amplitude += h.getAmplitude() \* Math.sin(2 \* Math.PI \* sampleIndex/samplesPerPeriod);

}

skipper++;

}

samples[sampleIndex] = amplitude;

//highest amplitude is stored for later normalization

if (Math.abs(samples[sampleIndex]) > highestAmplitude) {

highestAmplitude = Math.abs(samples[sampleIndex]);

}

}

return samples;

}

/\*\*

\* Converts a pixel into a harmonic of the column it is in.

\* <p/>

\* Calculates the frequency of the harmonic using the pixels vertical offset in the bitmap

\* column.

\* <p/>

\* Calculates the amplitude of the harmonic using the intensity of the color of the pixel.

\*

\* @param pixel the Bitmap pixel to be converted

\* @param verticalOffset the pixel's position in the column

\* @return Harmonic the harmonic representation of the pixel

\*/

private Harmonic pixelToHarmonic(int pixel, int verticalOffset) {

double frequency = (double) (IMAGE\_HEIGHT - verticalOffset) \*

NUMBER\_OF\_FREQUENCIES / (IMAGE\_HEIGHT + 1);

double amplitude = ((double) (Color.red(pixel) + Color.green(pixel) + Color.blue(pixel))) / 765 \*

MAX\_AMPLITUDE / IMAGE\_HEIGHT;

return new Harmonic(frequency, amplitude);

}

public void play() {

if (audio.getPlayState() == 3) {

audio.stop();

audio.reloadStaticData();

}

audio.play();

}

public void stop(){

audio.stop();

audio.reloadStaticData();

}

public void onPlaybackStopped(AudioTrack.OnPlaybackPositionUpdateListener listener){

audio.setNotificationMarkerPosition((int) (SAMPLE\_RATE \* IMAGE\_WIDTH\*grainDuration));

audio.setPlaybackPositionUpdateListener(listener);

}

public byte[] getDATA() {

return DATA;

}

public short getAudioChannelCount() {

return (short) audio.getChannelCount();

}

public int getAudioSampleRate() {

return audio.getSampleRate();

}

public void normalizeAudio() {

double multiplier = MAX\_AMPLITUDE / highestAmplitude; //what to multiply every sample by.

for (int i = 0; i < DATA.length; i = i + 2) {

int k = (int) ((twoBytesToAmplitude(DATA[i], DATA[i + 1])) \* multiplier);

DATA[i] = (byte) (k & 0x00ff);

DATA[i + 1] = (byte) ((k & 0xff00) >>> 8);

}

audio.write(DATA, 0, DATA.length);

}//normalizeAudio

private static double twoBytesToAmplitude(byte b1, byte b2) {

return ((b2 << 8) | (b1 & 0xFF));

}

}

# Harmonic.java

package imajadio;

public class Harmonic {

private final double frequency; //from 100Hz-10000Hz

private final double amplitude;

public Harmonic(double frequency, double amplitude) {

this.frequency = frequency;

this.amplitude = amplitude;

}

public double getFrequency() {

return frequency;

}

public double getAmplitude() {

return amplitude;

}

}

# activity\_main.xml

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:orientation="vertical"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent"

android:layout\_alignParentLeft="true"

android:layout\_alignParentStart="true">

<FrameLayout

android:layout\_width="match\_parent"

android:layout\_height="0dp"

android:layout\_weight="1">

<ImageView

android:id="@+id/imgPreview"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:visibility="visible"

android:layout\_gravity="center\_vertical|right"

android:scaleType="centerCrop"

/>

<ImageView

android:id="@+id/loadingBlack"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:visibility="visible"

android:layout\_gravity="center\_vertical|right"

android:scaleType="centerCrop"

android:background="@drawable/convertingblock"/>

<View

android:id="@+id/progressBar"

android:layout\_width="1dp"

android:layout\_height="match\_parent"

android:background="@drawable/statusbar"

android:visibility="gone"

/>

<ProgressBar

android:id="@+id/progressSpinner"

style="?android:attr/progressBarStyleLarge"

android:layout\_width="150dp"

android:layout\_height="150dp"

android:layout\_gravity="center"

android:visibility="gone"

/>

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:textAppearance="?android:attr/textAppearanceSmall"

android:text="Converting..."

android:textStyle="bold"

android:id="@+id/convertingText"

android:layout\_gravity="center"

android:visibility="visible"

android:singleLine="false"

android:textColor="#ffffff" />

</FrameLayout>

<LinearLayout

android:orientation="vertical"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:padding="10dp">

<RelativeLayout

android:layout\_width="match\_parent"

android:layout\_height="56dp">

<Button

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:id="@+id/playButton"

android:layout\_centerHorizontal="true" />

</RelativeLayout>

<LinearLayout

android:orientation="horizontal"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

>

<TextView

android:layout\_width="0dp"

android:layout\_height="match\_parent"

android:text="Column Duration"

android:gravity="center\_vertical"

android:id="@+id/Param1Text"

android:layout\_weight="2.5"/>

<SeekBar

android:layout\_width="0dp"

android:layout\_height="wrap\_content"

android:id="@+id/seekBarGrainDuration"

android:progress="25"

android:max="49"

android:progressDrawable="@drawable/seekbarline"

android:indeterminate="false"

android:layout\_weight="4"

android:maxHeight="2dp" />

<TextView

android:layout\_weight="1"

android:layout\_width="0dp"

android:layout\_height="match\_parent"

android:text="25 ms"

android:gravity="center\_vertical"

android:id="@+id/progressText"

/>

</LinearLayout>

</LinearLayout>

</LinearLayout>

# AndroidManifest.xml

<?xml version="1.0" encoding="utf-8"?>

<manifest xmlns:android="http://schemas.android.com/apk/res/android"

package="ca.brocku.imajadio.app" >

<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE"/>

<uses-permission android:name="android.permission.CAMERA"/>

<application

android:allowBackup="true"

android:icon="@drawable/imajadio"

android:label="@string/app\_name"

android:theme="@style/AppTheme" >

<activity

android:name="ca.brocku.imajadio.app.MainActivity"

android:label="@string/app\_name"

android:screenOrientation="portrait"

android:configChanges="orientation|keyboardHidden">

<intent-filter>

<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />

</intent-filter>

</activity>

</application>

</manifest>