KAUNAS UNIVERSITY OF TECHNOLOGY

FACULTY OF INFORMATICS

COMPUTER DEPARTMENT

App Development for Smart Mobile Systems Individual work

„Morse code flashlight translator"/"Morsify"

Assignment completed by:
IFF 6/8 group student
Tadas Laurinaitis

Assignment evaluated by:
Prof. Rytis Maskeliūnas

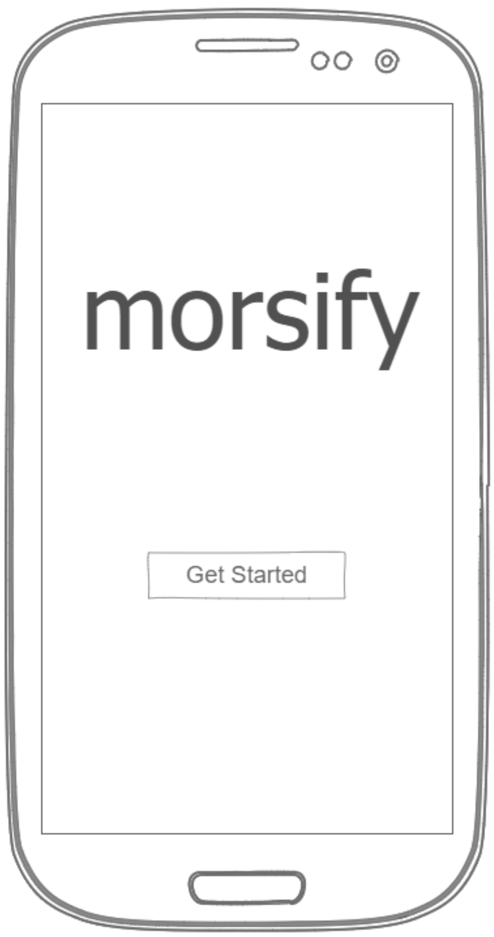
Table of contents:

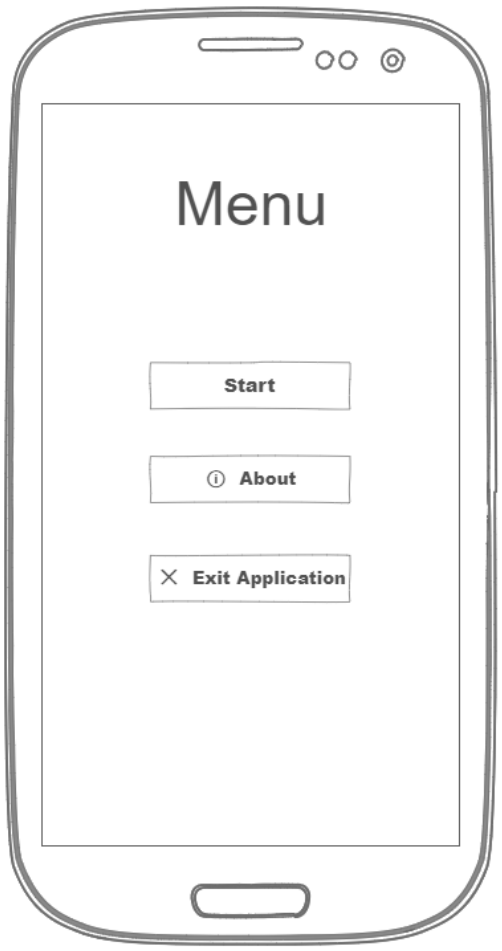
Contents

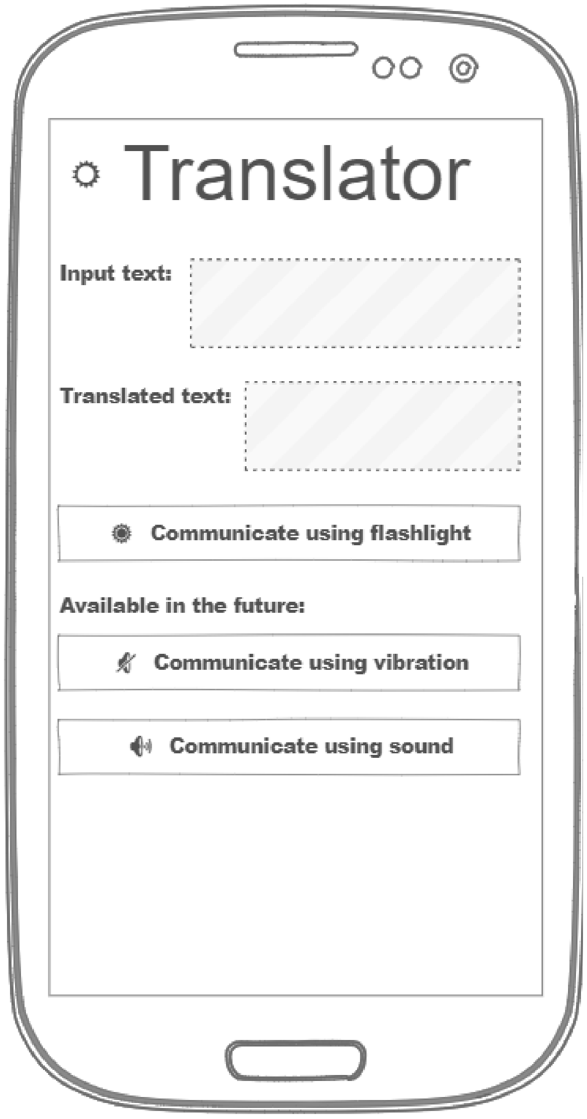
Description of app:

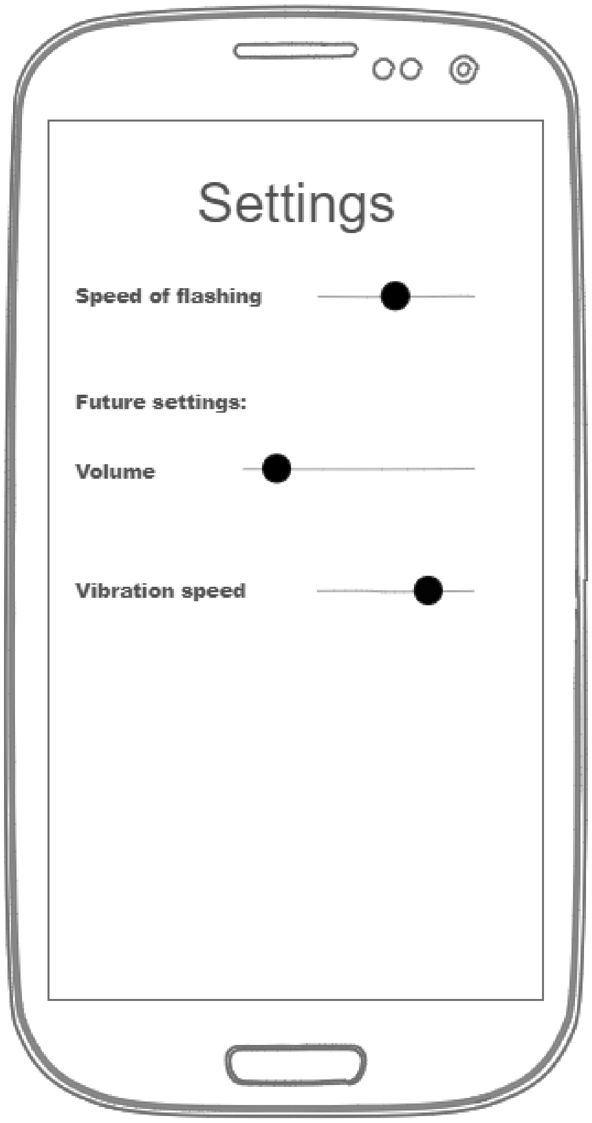
The application which I am creating is a tool used to translate written text to morse code and then show it using flashlight. User will be able to write a sentence, which then will be translated into morse code (dots and dashes) and will have an option to communicate that code using camera's flashlight. User will also have an option to see information about application, change application's settings and will be able to close application using „Exit Application" button.

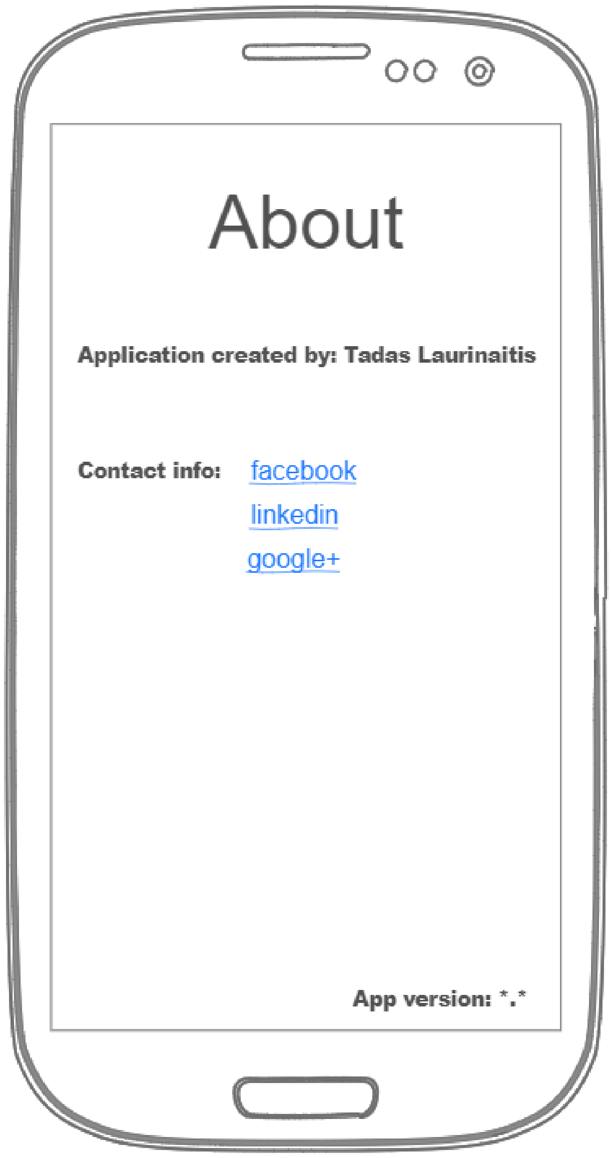
App mockup:






Translator
Input text:
Translated text:
Communicate using flashlight
Available in the future:
Communicate using vibration
Communicate using sound


Settings
Speed of flashing
Future settings:
Volume
Vibration speed



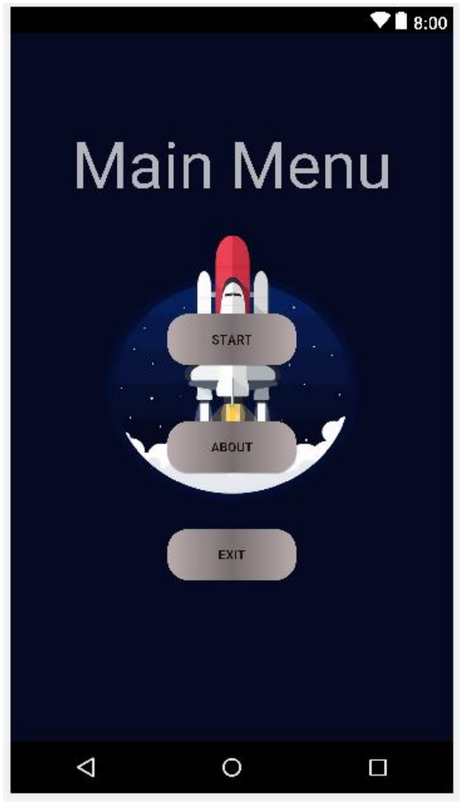
About

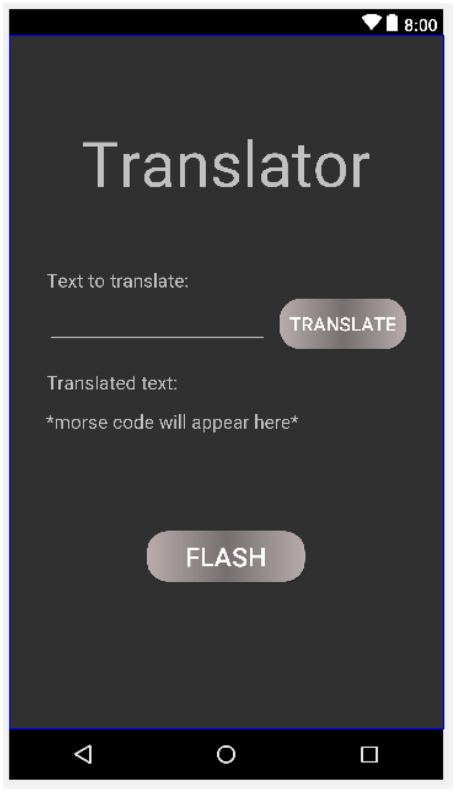
Application created by: Tadas Laurinaitis

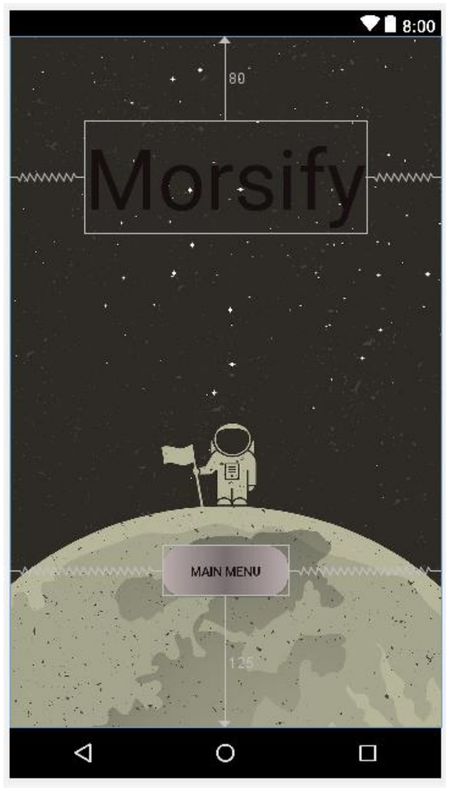
Contact info: facebook
linkedin
google+

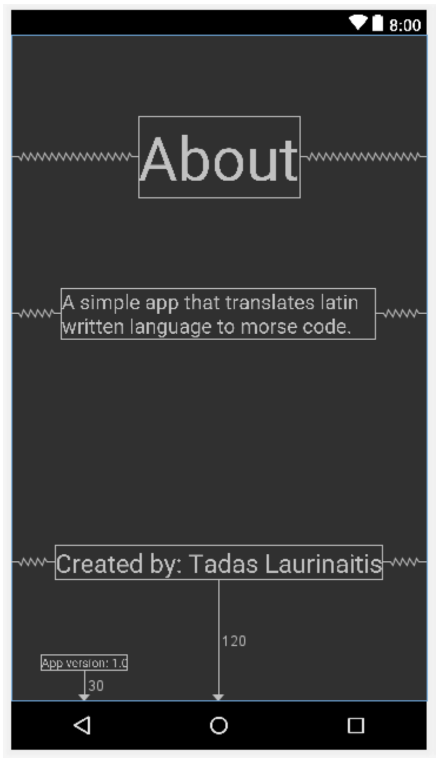
App version: *.*

User interface:





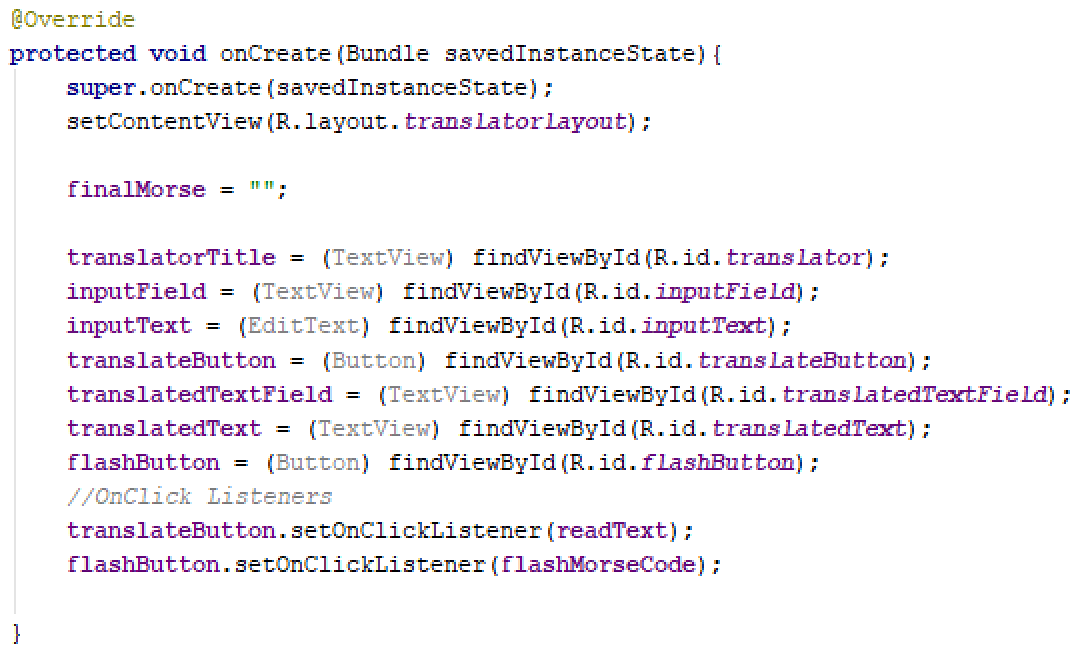



8:00
About
A simple app that translates latin written language to morse code.
Created by: Tadas Laurinaitis
120
App version: 1.0
30

Main functionality:

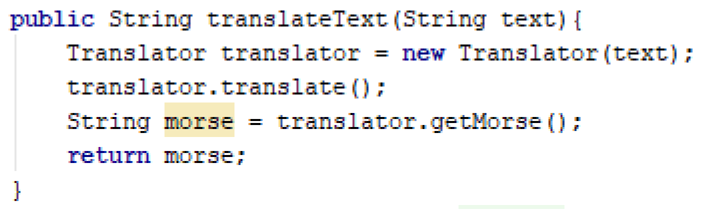
To translate the written text, I created a class named „Translator“ which is only responsible for the conversion of text to morse code in a form of string. This string is then passed on to a „for“ cycle which goes through every symbol of that same string. Inside the „for“ cycle there are 4 if statements which check if symbol meets certain requirements, for example: if symbol is equal to „-“ then it accesses the camera and turns it on and after a certain delay it turns it off. Thats the basic principle on which my app is built.

Main functions:



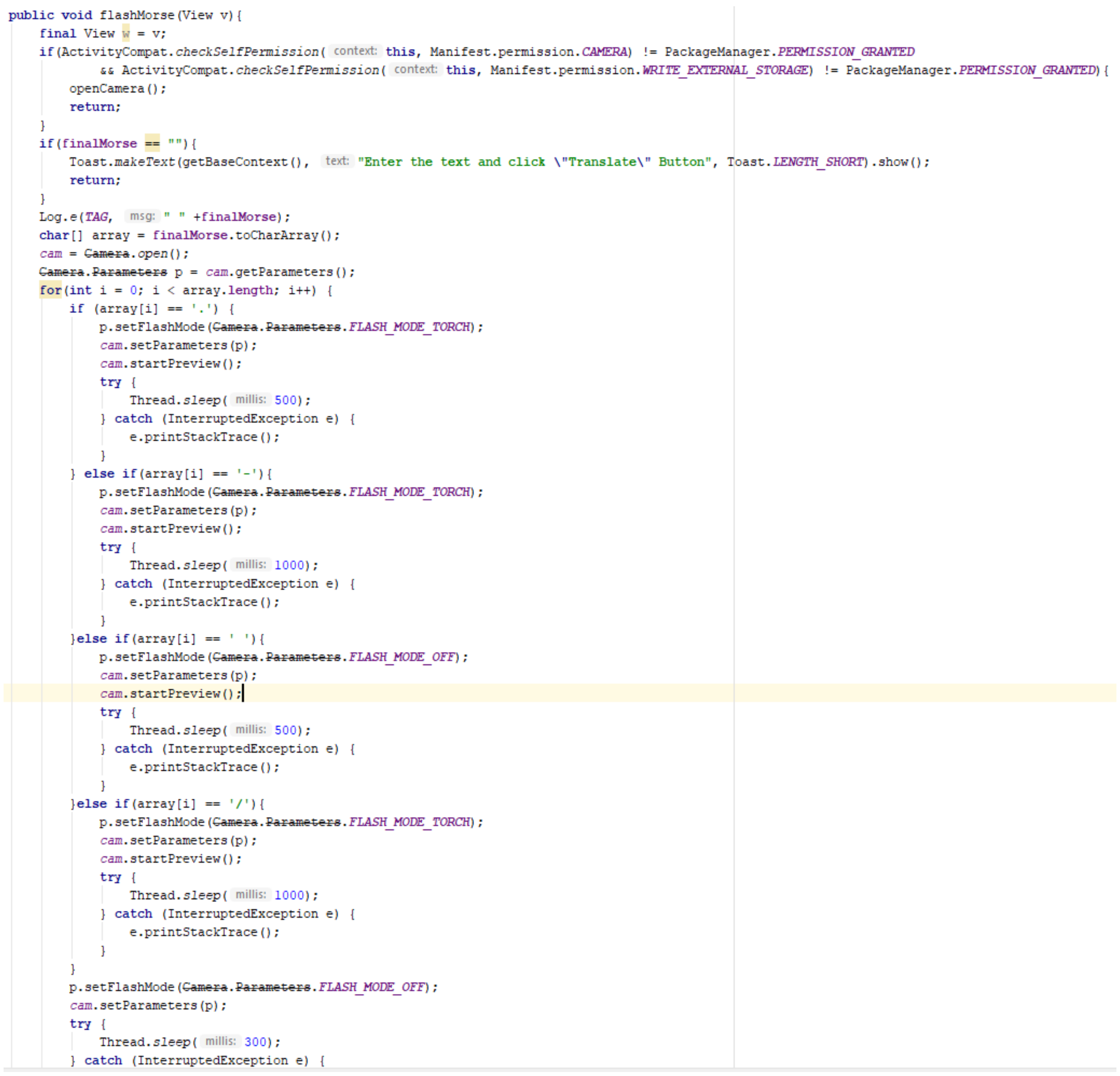
```
@Override
protected void onCreate(Bundle savedInstanceState){
    super.onCreate(savedInstanceState);
    setContentView(R.layout.translatorlayout);

    finalMorse = "";

    translatorTitle = (TextView) findViewById(R.id.translator);
    inputField = (TextView) findViewById(R.id.inputField);
    inputText = (EditText) findViewById(R.id.inputText);
    translateButton = (Button) findViewById(R.id.translateButton);
    translatedTextField = (TextView) findViewById(R.id.translatedTextField);
    translatedText = (TextView) findViewById(R.id.translatedText);
    flashButton = (Button) findViewById(R.id.flashButton);
    //OnClick Listeners
    translateButton.setOnClickListener(readText);
    flashButton.setOnClickListener(flashMorseCode);

}
```

Picture #1 onCreate() method which is responsible for initialization of values

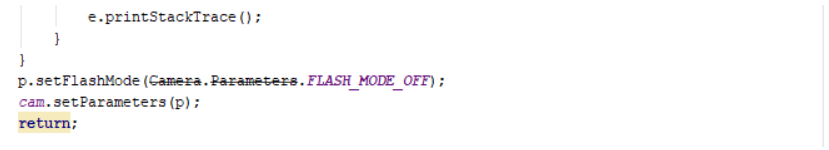


```
public String translateText(String text){
    Translator translator = new Translator(text);
    translator.translate();
    String morse = translator.getMorse();
    return morse;
}
```

Picture #2 translateText() method which is responsible for the translation of text which is passed through the parameters.

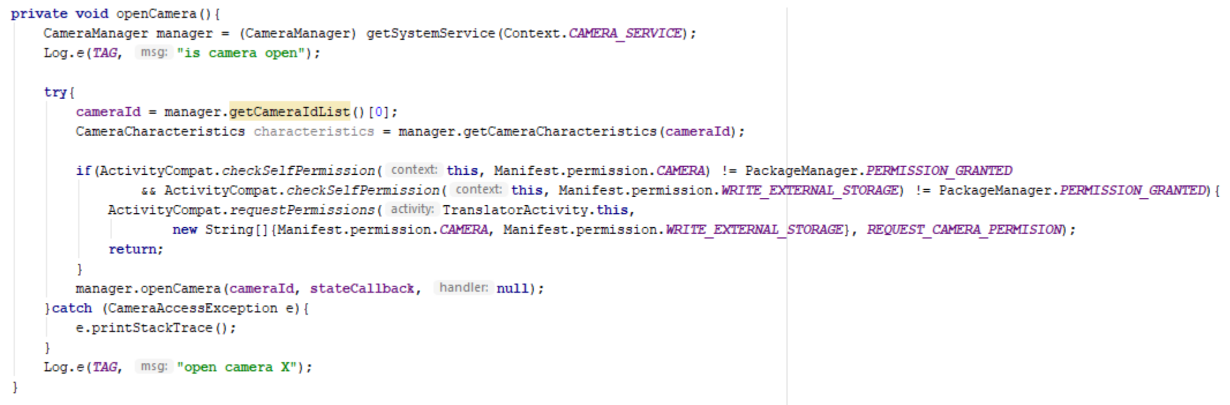


```
public void flashMorse(View v){
    final View w = v;
    if(ActivityCompat.checkSelfPermission( context: this, Manifest.permission.CAMERA) != PackageManager.PERMISSION_GRANTED
            && ActivityCompat.checkSelfPermission( context: this, Manifest.permission.WRITE_EXTERNAL_STORAGE) != PackageManager.PERMISSION_GRANTED){
        openCamera();
        return;
    }
    if(finalMorse == ""){
        Toast.makeText(getBaseContext(),  text: "Enter the text and click \"Translate\" Button", Toast.LENGTH_SHORT).show();
        return;
    }
    Log.e(TAG,  msg: " " +finalMorse);
    char[] array = finalMorse.toCharArray();
    cam = ~~Camera~~.open();
    ~~Camera.Parameters~~ p = cam.getParameters();
    for(int i = 0; i < array.length; i++) {
        if (array[i] == '.') {
            p.setFlashMode(~~Camera.Parameters~~.FLASH_MODE_TORCH);
            cam.setParameters(p);
            cam.startPreview();
            try {
                Thread.sleep( millis: 500);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        } else if(array[i] == '-'){
            p.setFlashMode(~~Camera.Parameters~~.FLASH_MODE_TORCH);
            cam.setParameters(p);
            cam.startPreview();
            try {
                Thread.sleep( millis: 1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }else if(array[i] == ' '){
            p.setFlashMode(~~Camera.Parameters~~.FLASH_MODE_OFF);
            cam.setParameters(p);
            cam.startPreview();
            try {
                Thread.sleep( millis: 500);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }else if(array[i] == '/'){
            p.setFlashMode(~~Camera.Parameters~~.FLASH_MODE_TORCH);
            cam.setParameters(p);
            cam.startPreview();
            try {
                Thread.sleep( millis: 1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
        p.setFlashMode(~~Camera.Parameters~~.FLASH_MODE_OFF);
        cam.setParameters(p);
        try {
            Thread.sleep( millis: 300);
        } catch (InterruptedException e) {
```



```
            e.printStackTrace();
        }
    }
    p.setFlashMode(~~Camera.Parameters~~.FLASH_MODE_OFF);
    cam.setParameters(p);
    return;
```

Picture #3 flashMorse() method which is responsible for the translation of morse code into camera flashes



```java
private void openCamera(){
    CameraManager manager = (CameraManager) getSystemService(Context.CAMERA_SERVICE);
    Log.e(TAG,  msg: "is camera open");

    try{
        cameraId = manager.getCameraIdList()[0];
        CameraCharacteristics characteristics = manager.getCameraCharacteristics(cameraId);

        if(ActivityCompat.checkSelfPermission( context: this, Manifest.permission.CAMERA) != PackageManager.PERMISSION_GRANTED
                && ActivityCompat.checkSelfPermission( context: this, Manifest.permission.WRITE_EXTERNAL_STORAGE) != PackageManager.PERMISSION_GRANTED){
            ActivityCompat.requestPermissions( activity: TranslatorActivity.this,
                    new String[]{Manifest.permission.CAMERA, Manifest.permission.WRITE_EXTERNAL_STORAGE}, REQUEST_CAMERA_PERMISION);
            return;
        }
        manager.openCamera(cameraId, stateCallback,  handler: null);
    }catch (CameraAccessException e){
        e.printStackTrace();
    }
    Log.e(TAG,  msg: "open camera X");
}
```

Picture #4 openCamera() method which is mainly responsible for checking the permissions to access camera and external storage.

Literature list:

https://stackoverflow.com/questions/6068803/how-to-turn-on-front-flash-light-programmatically-in-android

https://stackoverflow.com/questions/32929790/android-permission-denial-cant-use-the-camera

https://www.syncios.com/android/how-to-debug-samsung-galaxy-s8.html

https://www.tutorialspoint.com/java/util/timer_schedule_period.htm

https://stackoverflow.com/questions/18185384/blinking-flash-according-to-morse-code-android-how-to-avoid-anrs-due-to-sleepi

https://www.google.com/search?rlz=1C1CHBF_enLT815LT815&biw=1920&bih=1058&tbm=isch&sa=1&ei=cp4VXOChCMmTsgGAyY6IDQ&q=app+icon&oq=app+icon&gs_l=img.3..0i67l2j0j0i67j0j0i67j0l3j0i67.2933.3927..4108...0.0..0.71.537.8......1....1..gws-wiz-img.......35i39.ocVJUeBQYfI#imgrc=6c-SmPQcXpfQ3M:

Source code:

```java
package com.example.destroyer.morsecodeflashlighttranslator;

import android.app.Activity;
import android.content.Context;
import android.os.Bundle;
import android.widget.Button;
import android.widget.TextView;

public class AboutActivity extends Activity {

    private TextView aboutField;
    private TextView createdByField;
    private TextView appVersionField;
    private TextView summaryField;
    private Context context = this;

    @Override
    protected void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        setContentView(R.layout.aboutlayout);

        aboutField = (TextView) findViewById(R.id.aboutField);
        createdByField = (TextView) findViewById(R.id.createdByField);
        appVersionField = (TextView) findViewById(R.id.appVersionField);
        summaryField = (TextView) findViewById(R.id.summaryField);
    }

}



package com.example.destroyer.morsecodeflashlighttranslator;

import android.app.Activity;
import android.content.Intent;
import android.content.Context;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

public class GettingStartedActivity extends Activity{

    private TextView title;
    private Button toMainMenu;
    private Context context = this;

    @Override
    protected void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        setContentView(R.layout.gettingstartedlayout);

        title = (TextView) findViewById(R.id.title);
        toMainMenu = (Button) findViewById(R.id.toMainMenuButton);
        toMainMenu.setOnClickListener(startMainMenuActivity);
    }
    public void runMainMenuActivity(){
        Intent intent = new Intent(context, MainMenuActivity.class);
        context.startActivity(intent);
    }
    View.OnClickListener startMainMenuActivity = new View.OnClickListener(){
        @Override
        public void onClick(View v){
            runMainMenuActivity();
        }
    };
}
```

```
package com.example.destroyer.morsecodeflashlighttranslator;

import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

public class MainMenuActivity extends Activity{

    private TextView menuField;
    private Button startButton;
    private Button aboutButton;
    private Button exitButton;
    private Context context = this;

    @Override
    protected void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        setContentView(R.layout.mainmenulayout);

        menuField = (TextView) findViewById(R.id.menuField);
        startButton = (Button) findViewById(R.id.startButton);
        aboutButton = (Button) findViewById(R.id.aboutButton);
        exitButton = (Button) findViewById(R.id.exitButton);

        startButton.setOnClickListener(startTranslatorActivity);
        aboutButton.setOnClickListener(startAboutActivity);
        exitButton.setOnClickListener(exitListener);
    }

    public void runTranslatorActivity(){
        Intent intent = new Intent(context, TranslatorActivity.class);
        context.startActivity(intent);
    }
    View.OnClickListener startTranslatorActivity = new View.OnClickListener(){
        @Override
        public void onClick(View v){
            runTranslatorActivity();
        }
    };

    public void runAboutActivity(){
        Intent intent = new Intent(context, AboutActivity.class);
        context.startActivity(intent);
    }
    View.OnClickListener startAboutActivity = new View.OnClickListener(){
        @Override
        public void onClick(View v){
            runAboutActivity();
        }
    };
```

```java
    View.OnClickListener exitListener = new View.OnClickListener(){
        @Override
        public void onClick(View v){
            finish();
            System.exit(0);
        }
    };
}
```

```java
package com.example.destroyer.morsecodeflashlighttranslator;

import android.app.Activity;
import android.content.Context;
import android.os.Bundle;

public class SettingsActivity extends Activity {

    Context context = this;

    @Override
    protected void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        setContentView(R.layout.aboutlayout);
    }
}
```

```java
package com.example.destroyer.morsecodeflashlighttranslator;

public class Translator {

    String primaryText;
    String morseCode;

    public Translator(String primaryText){
        this.primaryText = primaryText;
        morseCode = "";
    }

    public void translate(){
        String text = primaryText.toLowerCase();
        char[] array = text.toCharArray();
        String morse = "";
        String current = "";
        for(int i = 0; i < array.length; i++){
            current = "";
            if(array[i] == 'a'){
                current = ".-";
                morse += current;
                morse += " ";
            }
            else if(array[i] == 'b'){
                current = "-...";
                morse += current;
                morse += " ";
            }
            else if(array[i] == 'c'){
                current = "-.-.";
                morse += current;
                morse += " ";
            }
            else if(array[i] == 'd'){
                current = "-..";
                morse += current;
                morse += " ";
            }
            else if(array[i] == 'e'){
                current = ".";
                morse += current;
                morse += " ";
            }
            else if(array[i] == 'f'){
                current = "..-.";
                morse += current;
                morse += " ";
            }
            else if(array[i] == 'g'){
                current = "--.";
                morse += current;
                morse += " ";
            }
            else if(array[i] == 'h'){
                current = "....";
                morse += current;
                morse += " ";
            }
            else if(array[i] == 'i'){
                current = "..";
                morse += current;
                morse += " ";
            }
            else if(array[i] == 'j'){
                current = ".---";
                morse += current;
                morse += " ";
            }
            else if(array[i] == 'k'){
                current = "-.-";
                morse += current;
                morse += " ";
            }
            else if(array[i] == 'l'){
                current = ".-..";
                morse += current;
                morse += " ";
            }
            else if(array[i] == 'm'){
                current = "--";
                morse += current;
                morse += " ";
            }
            else if(array[i] == 'n'){
                current = "-.";
                morse += current;
                morse += " ";
            }
            else if(array[i] == 'o'){
                current = "---";
                morse += current;
                morse += " ";
            }
            else if(array[i] == 'p'){
```

```
            current = ".--.";
            morse += current;
            morse += " ";
        }
        else if(array[i] == 'q'){
            current = "--.-";
            morse += current;
            morse += " ";
        }
        else if(array[i] == 'r'){
            current = ".-.";
            morse += current;
            morse += " ";
        }
        else if(array[i] == 's'){
            current = "...";
            morse += current;
            morse += " ";
        }
        else if(array[i] == 't'){
            current = "-";
            morse += current;
            morse += " ";
        }
        else if(array[i] == 'u'){
            current = "..-";
            morse += current;
            morse += " ";
        }
        else if(array[i] == 'v'){
            current = "...-";
            morse += current;
            morse += " ";
        }
        else if(array[i] == 'w'){
            current = ".--";
            morse += current;
            morse += " ";
        }
        else if(array[i] == 'x'){
            current = "-..-";
            morse += current;
            morse += " ";
        }
        else if(array[i] == 'y'){
            current = "-.--";
            morse += current;
            morse += " ";
        }
        else if(array[i] == 'z'){
            current = "--..";
            morse += current;
            morse += " ";
        }
        else if(array[i] == ' '){
            current = "/";
            morse += current;
            morse += " ";
        }
        else
            continue;
    }
    morseCode = morse;
```

```java
    }

    public String getMorse(){
        return morseCode;
    }
}



package com.example.destroyer.morsecodeflashlighttranslator;

import android.Manifest;
import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.graphics.SurfaceTexture;
import android.hardware.camera2.CameraAccessException;
import android.hardware.camera2.CameraCaptureSession;
import android.hardware.camera2.CameraCharacteristics;
import android.hardware.camera2.CameraDevice;
import android.hardware.camera2.CameraManager;
import android.hardware.camera2.CaptureRequest;
import android.hardware.camera2.params.StreamConfigurationMap;
import android.media.ImageReader;
import android.os.Bundle;
import android.os.Handler;
import android.os.HandlerThread;
import android.support.annotation.NonNull;
import android.support.v4.app.ActivityCompat;
import android.util.Log;
import android.util.Size;
import android.util.SparseIntArray;
import android.view.Surface;
import android.view.TextureView;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.hardware.Camera;
import android.widget.Toast;

import java.io.File;
import java.util.Timer;
import java.util.TimerTask;

public class TranslatorActivity extends Activity {

    Context context = this;
    public static Camera cam = null;
    TextView translatorTitle;
    String finalMorse;

    TextView inputField;
    EditText inputText;
    Button translateButton;

    TextView translatedTextField;
    TextView translatedText;
    Button flashButton;

    //Part 3 - Camera
    private static final String TAG = "AndroidCameraApi";
    private static final SparseIntArray ORIENTATIONS = new SparseIntArray();
    static{
        ORIENTATIONS.append(Surface.ROTATION_0, 90);
        ORIENTATIONS.append(Surface.ROTATION_90, 0);
        ORIENTATIONS.append(Surface.ROTATION_180, 270);
        ORIENTATIONS.append(Surface.ROTATION_270, 180);
    }
    private String cameraId;
    protected CameraDevice cameraDevice;
    private static final int REQUEST_CAMERA_PERMISION = 200;
    private Handler mBackgroundHandler;
    private HandlerThread mBackgroundThread;

    @Override
    protected void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        setContentView(R.layout.translatorlayout);

        finalMorse = "";

        translatorTitle = (TextView) findViewById(R.id.translator);
        inputField = (TextView) findViewById(R.id.inputField);
        inputText = (EditText) findViewById(R.id.inputText);
        translateButton = (Button) findViewById(R.id.translateButton);
        translatedTextField = (TextView) findViewById(R.id.translatedTextField);
        translatedText = (TextView) findViewById(R.id.translatedText);
        flashButton = (Button) findViewById(R.id.flashButton);
        //OnClick Listeners
        translateButton.setOnClickListener(readText);
        flashButton.setOnClickListener(flashMorseCode);

    }

    public String translateText(String text){
        Translator translator = new Translator(text);
        translator.translate();
        String morse = translator.getMorse();
        return morse;
    }
    View.OnClickListener readText = new View.OnClickListener(){
        @Override
        public void onClick(View v){
            String text = inputText.getText().toString();
            Log.e(TAG, "got string from box: " +text);
            finalMorse = translateText(text);
            Log.e(TAG, "translated string from box: " +finalMorse);
            //System.out.println(morse);
            translatedText.setText(finalMorse);
        }
    };

    public void flashMorse(View v){
        final View w = v;
        if(ActivityCompat.checkSelfPermission(this, Manifest.permission.CAMERA) !=
PackageManager.PERMISSION_GRANTED
                && ActivityCompat.checkSelfPermission(this,
Manifest.permission.WRITE_EXTERNAL_STORAGE) != PackageManager.PERMISSION_GRANTED){
            openCamera();
            return;
        }
        if(finalMorse == ""){
            Toast.makeText(getBaseContext(), "Enter the text and click \"Translate\"
Button", Toast.LENGTH_SHORT).show();
```

```
            return;
        }
        Log.e(TAG, " " +finalMorse);
        char[] array = finalMorse.toCharArray();
        cam = Camera.open();
        Camera.Parameters p = cam.getParameters();
        for(int i = 0; i < array.length; i++) {
            if (array[i] == '.') {
                p.setFlashMode(Camera.Parameters.FLASH_MODE_TORCH);
                cam.setParameters(p);
                cam.startPreview();
                try {
                    Thread.sleep(500);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            } else if(array[i] == '-'){
                p.setFlashMode(Camera.Parameters.FLASH_MODE_TORCH);
                cam.setParameters(p);
                cam.startPreview();
                try {
                    Thread.sleep(1000);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }else if(array[i] == ' '){
                p.setFlashMode(Camera.Parameters.FLASH_MODE_OFF);
                cam.setParameters(p);
                cam.startPreview();
                try {
                    Thread.sleep(500);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }else if(array[i] == '/'){
                p.setFlashMode(Camera.Parameters.FLASH_MODE_TORCH);
                cam.setParameters(p);
                cam.startPreview();
                try {
                    Thread.sleep(1000);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
            p.setFlashMode(Camera.Parameters.FLASH_MODE_OFF);
            cam.setParameters(p);
            try {
                Thread.sleep(300);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
        p.setFlashMode(Camera.Parameters.FLASH_MODE_OFF);
        cam.setParameters(p);
        return;
    }

    View.OnClickListener flashMorseCode = new View.OnClickListener(){
        @Override
        public void onClick(View v){
            flashMorse(v);
        }
    };
    public void flashLightOn(View view) {
        try {
            if (getPackageManager().hasSystemFeature(
                    PackageManager.FEATURE_CAMERA_FLASH)) {
                cam = Camera.open();
                Camera.Parameters p = cam.getParameters();
                p.setFlashMode(Camera.Parameters.FLASH_MODE_TORCH);
                cam.setParameters(p);
                cam.startPreview();
            }
        } catch (Exception e) {
            e.printStackTrace();
            Toast.makeText(getBaseContext(), "Exception flashLightOn()",
                    Toast.LENGTH_SHORT).show();
        }
    }

    public void flashLightOff(View view) {
        try {
            if (getPackageManager().hasSystemFeature(
                    PackageManager.FEATURE_CAMERA_FLASH)) {
                cam.stopPreview();
                cam.release();
                cam = null;
            }
        } catch (Exception e) {
            e.printStackTrace();
            Toast.makeText(getBaseContext(), "Exception flashLightOff",
                    Toast.LENGTH_SHORT).show();
        }
    }

    //Stuff from labaratory work
    private void openCamera(){
        CameraManager manager = (CameraManager)
getSystemService(Context.CAMERA_SERVICE);
        Log.e(TAG, "is camera open");

        try{
            cameraId = manager.getCameraIdList()[0];
            CameraCharacteristics characteristics =
manager.getCameraCharacteristics(cameraId);

            if(ActivityCompat.checkSelfPermission(this, Manifest.permission.CAMERA) !=
PackageManager.PERMISSION_GRANTED
                    && ActivityCompat.checkSelfPermission(this,
Manifest.permission.WRITE_EXTERNAL_STORAGE) != PackageManager.PERMISSION_GRANTED){
                ActivityCompat.requestPermissions(TranslatorActivity.this,
                        new String[]{Manifest.permission.CAMERA,
Manifest.permission.WRITE_EXTERNAL_STORAGE}, REQUEST_CAMERA_PERMISION);
                return;
            }
            manager.openCamera(cameraId, stateCallback, null);
        }catch (CameraAccessException e){
            e.printStackTrace();
        }
        Log.e(TAG, "open camera X");
    }

    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull String[]
permissions, @NonNull int[] grantResults){
        if(requestCode == REQUEST_CAMERA_PERMISION){
            if(grantResults[0] == PackageManager.PERMISSION_DENIED){
                Toast.makeText(TranslatorActivity.this, "You cant use this app without 
granting permission", Toast.LENGTH_SHORT).show();
                finish();
            }
        }
    }

    private final CameraDevice.StateCallback stateCallback = new 
CameraDevice.StateCallback() {
        @Override
        public void onOpened(CameraDevice camera) {
            Log.e(TAG, "onOpened");
            cameraDevice = camera;
        }

        @Override
        public void onDisconnected(@NonNull CameraDevice camera) {
            cameraDevice.close();
        }

        @Override
        public void onError(@NonNull CameraDevice camera, int error) {
            cameraDevice.close();
            cameraDevice = null;
        }
    };
/*
    public void flashMorse(View v){
        final View w = v;
        openCamera();
        if(finalMorse == ""){
            Toast.makeText(getBaseContext(), "Enter the text and click \"Translate\" 
Button", Toast.LENGTH_SHORT).show();
            return;
        }
        Log.e(TAG, " " +finalMorse);
        char[] array = finalMorse.toCharArray();

        for(int i = 0; i < array.length; i++){
            Log.e(TAG, " " +i +" raide: " +array[i]);
            Timer timer = new Timer();
            if(array[i] == '.'){
                flashLightOn(w);
                TimerTask timer_task = new TimerTask() {
                    public void run() {
                        flashLightOff(w);
                    }
                };
                timer.schedule(timer_task, 5000);
                timer_task.cancel();
            }else if(array[i] == '-'){
                flashLightOn(w);
                TimerTask timer_task = new TimerTask() {
                    public void run() {
                        flashLightOff(w);
                    }
                };
                timer.schedule(timer_task, 8000);
                timer_task.cancel();
            }else if(array[i] == ' '){
                TimerTask timer_task = new TimerTask() {
                    public void run() {
```

```
                    //flashLightOff(w);
                }
            };
            timer.schedule(timer_task, 8000);
            timer_task.cancel();
        }else if(array[i] == '/'){
            TimerTask timer_task = new TimerTask() {
                public void run() {
                    //flashLightOff(w);
                    int sass = 0;
                }
            };
            timer.schedule(timer_task, 2000);
            timer_task.cancel();
        }
        timer.cancel();
    }
    flashLightOff(w);
  }
*/
}
```