Sensor Based Music Player

Srushti Desai  
Department of Computer Science  
Binghamton University  
sdesai9@binghamton.edu

Roja Eswaran  
Department of Computer Science  
Binghamton University  
reswara1@binghamton.edu

ABSTRACT:

There are many sensors present in the android phone that we do not use such as light sensor, proximity sensor, accelerometer, humidity etc. There is not even a single main stream Music app which makes use of these sensors. For fun, we decided to make a Music app which can work as a Sensor Based Music player. If a person does not want to play the Music player based on sensors, it can work as a normal music player also. We made the entire Music Player from scratch using Android studio.

1. INTRODUCTION:

Our motto is to create a Music player app from scratch. We will give the user two options, if they want to play it as a normal music player or if they want to play it as a Sensor Based Music player. The user should click on an option and decide for themselves. Once the user decides to play a Sensor-based Music player the song starts playing, if the user satisfies the condition of proximity sensor and puts an obstacle at 2-5 cm from the phone the song pauses, once the obstacle is removed it starts playing again. We will also use the light sensor, if the light intensity is higher than 50 the song will play or else the song will pause. The third sensor which we are going to use is the accelerometer sensor. If we shake the phone at a force of 4 m/s or greater on the x-axis it plays the next song. If the user decides to use the option of a normal music player, the app will work as a normal music player. There will also be buttons to fast forward the song, pause the song, play the song.

2. DESIGN:

First, we designed the basic layout 1. Once the app is opened it opens a layout with a Pikachu photo asking you to click it.

Layout 2: After clicking on Pikachu, you get two options either to play it as an ordinary music player or a sensor-based music player.

Layout 3: Let’s say you decide to use the sensor-based music player first, it will give you two options, to either use it with an accelerometer or with light/proximity sensor.

Layout 4: If you choose any one of these, it will show you the song which is being played. If you have chosen light/proximity sensor you will be able to play/pause the song by satisfying the conditions of the sensors mentioned in the implementations. You can also change the song by shaking the phone(satisfying the condition of an accelerometer).
Let's say you selected the normal music player instead of the sensor-based music player. It will show the following layout which works as a normal music player. It allows you to play, pause the song and fast forward the song by 5 sec.:

3. IMPLEMENTATION:

3.1 Songs:

There is a raw folder in our API, in which we have loaded the songs. In the Main Activity of our android Project, we are adding all the songs of the raw folder into an array list. We added 6 songs for our mobile app.

3.2 Normal Music Player:

Once the user decides to use the normal Music Player instead of Sensor-based music player layout 5 is displayed. The normal music player was implemented to give options of playing, pausing and fast forwarding the song. The basic logic code is:

```java
b1.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        int temp = (int)startTime;
        if((temp+forwardTime)<=finalTime){
            startTime = startTime + forwardTime;
            mediaPlayer.seekTo((int)startTime);
            Toast.makeText(getApplicationContext(),"You have Jumped forward 5 seconds",Toast.LENGTH_SHORT).show();
        }
        else{
            Toast.makeText(getApplicationContext(),"Cannot jump forward 5 seconds",Toast.LENGTH_SHORT).show();
        }
    }
});
```

Once a user presses the forward button, this logic checks the current time of the song. Then it checks if there are more 5 secs left for the song to get over? If more 5 secs are left it adds the 5 sec or else refuses from fast forwarding. The basic logic to start and stop the song is MediaPlayer.start() and MediaPlayer.stop().

3.3 Light Sensor:

The light sensor value was set to 50, such that if the intensity of light is less than 50 it paused the song and shows light sensor activated stop and if it more than 50 the song played and shows light sensor is activated. The basic logic code of the light sensor implementation is as follows:

```java
if(event.sensor.getType()==Sensor.TYPE_LIGHT){
    if(value<50)
    {
        Toast.makeText(getApplicationContext(),"LightSensorActivated_STOP",Toast.LENGTH_SHORT).show();
        if (mp.isPlaying()) {
            mp.pause();
        }
    }
    else{
        Toast.makeText(getApplicationContext(),"LightSensorActivated_Start",Toast.LENGTH_SHORT).show();
        if (!mp.isPlaying()) {
            mp.setLooping(true);
            mp.start();
            event.values[0]=0;
        }
    }
}
```

3.4 Proximity Sensor:

The Proximity sensor value was set to 5, such that if there is an obstacle at a distance 5 cm or less than 5 from the proximity sensor, the song paused and if it more than 5 the song plays. The basic logic code of the proximity sensor implementation is as follows:

```java
if (event.sensor.getType() == Sensor.TYPE_PROXIMITY) {
    if (!mp.isPlaying()) {
        if (event.values[0] < 5) {
            if (!mp.isPlaying()) {
                mp.setLooping(true);
            }
            mp.start();
            event.values[0]=0;
        }
    }
    else{
        Toast.makeText(getApplicationContext(),"ProximitySensorActivated",Toast.LENGTH_SHORT).show();
        if (!mp.isPlaying()) {
            mp.setLooping(true);
            mp.start();
            event.values[0]=0;
        }
    }
}
```
mp.start();
}
} else {
    if (mp.isPlaying()) {
        mp.pause();
    }
}

3.5 Accelerometer Sensor:

The formula used for accelerometer is as follows \(\sqrt{x^2+y^2+z^2}\). The condition set by us was that \(\sqrt{x^2+y^2+z^2} > 30\) and \(x > 4\) that is force particularly in the x direction should be more than 4 m/s, if these conditions are satisfied then it played the next song. Since we have added 6 songs to our project file, after 6th song it will again start playing the first song. The basic logic of the accelerometer is as follows:

```java
if (event.sensor.getType() == Sensor.TYPE_ACCELEROMETER) {
    float x = event.values[0];
    float y = event.values[1];
    float z = event.values[2];
    if (Math.sqrt(Math.pow(x, 2.0) + Math.pow(y, 2.0) + Math.pow(z, 2.0)) > 30 && x > 4) {
        // playNext();
        if (i <= 6) {
            mediaPlayer.reset();
            i++;
            i = i % 6;
            mediaPlayer = MediaPlayer.create(accelerometer.this, play_list.get(i));
            mediaPlayer.start();
        }
    }
}
```

4. EVALUATION:

The Music player was implemented exactly the way mentioned in the implementation. We were able to change the songs by shaking the phone (in case we had selected the option to use accelerometer). We were also able to play/pause the song by keeping an obstacle at 2-5 cm and removing it or by decreasing and increasing the light intensity respectively.

5. CONCLUSION:

Our Music player works properly but there are chances that the user might want the music to pause at a different light intensity or if an obstacle is placed at a distance more than 5 cm from the proximity sensor. For this the user will have to make changes in the code itself. The Music Player can be upgraded by adding code such that when the phone is shaken on x-axis it plays pop songs, when shaken on y-axis it plays jazz and when shaken on z-axis it plays country music.

ACKNOWLEDGEMENT:

We would like to thank the professor for his support.

REFERENCES:

https://www.youtube.com/watch?v=8rNFZdQbYY8
https://www.youtube.com/watch?v=2VJ1zeEVL8A