



# **DJ Tool: A Mobile Phone Audio Player Application**

Oriol Boix Sancho

Supervisor: Roberto Bresin

Degree Project in Music Acoustics  
KTH – School of Computer Science and Communication (CSC)  
Department of Speech, Music and Hearing  
100 44 Stockholm



# Abstract

Since listening to music using mobile phones has become a normal situation, companies have started to think about developing new tools within the music field. Thereby, one of the fields in what they are working on is to enable users to put effects on their music while listening to it. One of the most popular techniques for manipulating recorded sounds is scratching. To scratch is just a variation in speed and direction of a sound file. These techniques are the presentation card of DJs, who implement them using a turntable and an audio mixer.

The purpose of this project is to design and implement a system which enables users to use their handset for manipulating the sound by using different tools, being one of them adding DJ scratching sounds on top of their music while listening to it from their mobile phones. The application is developed by using Python, and designed to be used on Nokia mobile phones.

# Acknowledgements

In the first place I would like to thank my supervisor Roberto Bresin for giving me the opportunity to do my Degree project at KTH and for his invaluable guidance and help at any time during its realization.

I am also in debt with Kjetil Falkenberg and Gäel Dubus who have been always ready to give me a hand with many aspects of the work whenever I asked for it.

I have to thank as well all the people in the department of Speech Music and Hearing whose warm hospitality really made me feel at home.

Finally, I express my warmest gratitude to my family for the continuous support and encouragement they gave me to accomplish this job.

# List of Abbreviations

**DJ** -- Disc-Jockey  
**SMS** -- Short Message Service  
**PD** -- Pure Data  
**GRiPD** -- Graphical Interface for Pure Data  
**WAV** -- Waveform Audio Format  
**PyS60** -- Python for S60  
**PyGTK** -- Python for GTK



# Table of Contents

1. INTRODUCTION.....	9
2. DJs THEORY .....	11
2.1 Brief Historical Introduction .....	11
2.2 Equipment.....	12
2.2.1 <i>Turntable</i> .....	13
2.2.2 <i>Audio Mixer</i> .....	14
2.3 Scratching .....	16
2.4 Techniques .....	16
2.4.1 <i>Baby</i> .....	17
2.4.2 <i>Tear</i> .....	17
2.4.3 <i>Rolltear</i> .....	18
2.4.4 <i>Chopshort</i> .....	18
2.4.5 <i>Forward</i> .....	19
2.4.6 <i>Silent Backdraw</i> .....	19
2.4.7 <i>Scribble</i> .....	20
2.4.8 <i>Uzi</i> .....	20
2.4.9 <i>Chirps</i> .....	21
2.4.10 <i>Flare</i> .....	21
2.4.11 <i>Crab</i> .....	22
2.4.12 <i>Twiddle</i> .....	22
3. SOFTWARE.....	23
3.1 Skipproof.....	23

3.2 Python.....	24
3.3 Nokia S60 Emulator .....	25
4. IMPLEMENTATION .....	27
4.1 Functions.....	27
4.2 Interface .....	29
4.3 Manipulation of the sound .....	32
5. DISCUSSION .....	35
6. FURTHER RESEARCH .....	37
References.....	39
Appendices.....	41
Appendix A. Audio mobile phone player script in Python .....	41
Appendix B. Keys' functions .....	52

# 1 Introduction

Mobile phones technology is growing quickly, as in almost all electronic devices. Nowadays, a mobile phone is not only a device used to call or send short messages (SMS), but it is also used to send e-mails, take pictures, record videos, listening to music and a wide variety of other tools which turn this instrument into a kind of small laptop that we can carry in our pocket. Accordingly, when we are going to an electronic store to buy a mobile phone, we don't ask about calling or messaging tools, but we ask about the capacity to store data (i.e. pictures, music), the quality of the camera and the new tools that this mobile phone offers us. Thus, companies are working on new tools to make people feel attraction to its new devices.

One of the most important concepts in this field is music. Music, in some way, represents our way of being, our character and even our mood in some moments. For this reason, it is important for us to carry with our favorite music and be able to listen to it anywhere, anyway and in any moment. Obviously, we are able to do it since many years ago. Hence, the question for the companies is now: What can we offer new to our costumers within the music listening tool?

There is an answer to this question. It seems that the next step to be followed by the companies is to develop tools which permit users adding effects to the music while listening to it. The number of effects that can be added to a sound file are infinite. Exist an important variety of ways to manipulate a sound file, and one of the most popular and seldom studied is the scratching effect. To scratch means "to drag a vinyl record forth and back against the needle on an ordinary turntable along the grooves" [1].

This technique is developed and used by DJs. The introduction of digital means for scratching introduced by electronic instrument manufacturers and computer programmers during last years stems from the popularity that scratching achieved in the 1990's [2]. Nevertheless, most of these new products have failed.

The aim of this project is to design and implement a system which enables users to use their handset for manipulating the sound they are listening to on their mobile phones by using different tools, being one of them adding DJ scratching sounds.

## 2 DJs Theory

Though some people think that “DJing<sup>1</sup>” is something that was born few years ago, this way to create and play music is not precisely something new.

In the next paragraphs, a brief historical introduction is presented. We will see how “DJs world” was born and how has evolved during the pass of the years. Furthermore, it is explained the equipment used by a DJ to perform a session, to record an album, etc...Finally, it is described in a wide way what scratching consists in and its different techniques.

### 2.1 Brief Historical Introduction

Since a DJ is that person who plays recorded sound, it is possible to say that the first stone was put in 1877, when Thomas Alva Edison invented the phonograph cylinder [3]. This device was the first one capable to play back recorded sound. However, was in 1935 when first appeared the term Disc-Jockey. The American commentator, Walter Winchell, called the radio announcer Martin Block that way because he “was the first who created the illusion that he was broadcasting from a ballroom while that nation’s top dance bands performing live” [4]. Six years later, the Disc-Jockey term appeared in the American weekly magazine Variety [5]. Since then, discotheques began to appear in the Unites States and Europe with the consequent appearance of DJs around the world. Another important name was Jimmy Savile. He became the first DJ who started using two turntables for continuous

---

<sup>1</sup> For *DJing* is understood the action that a DJ carries out on his/her profession.

play [5]. But this phenomenon did not only take place in the discotheques of Europe or the U.S.A. In the late 1950s appeared in Kingston (Jamaica) the so-called “sound systems”. It was a new way of entertainment and partying on the streets of the city. The DJ, also called “selector”, played music for hours using a rhythmic changing style. The success of these parties was absolute as business quickly received the call and the promoters started to sell admissions, food and alcohol. In fact, one of those “selectors” is known as the creator of Hip-Hop music and the one who invented the scratching technique. His name is Clive Campbell, also known as DJ Kool Herc. Clive, born in Kingston, moved when he was 15 years old to Bronx borough in NY. Thereby, he started to give parties on the streets as he had done in his natal city. He realized that people became crazy when the rhythmical part (brake) of James Brown’s songs sounded. Hence, to enlarge this part, he decided to take the needle of the turntable back where the break started to repeat that part of the song as many times as he wanted [5].

Since that time, the figure of the DJ was extended around new countries all around the world. During the last decades equipment has been improved which has allowed DJs create new techniques and sounds.

Nowadays, new digital interfaces are being designed and developed, fact that “old school” DJs don’t agree due to these new interfaces don’t allow to have and show the skills that the contact with the vinyl record and the needle permit. Nevertheless, these new technologies can help to develop and experiment with new techniques in order to find new sounds and effects.

## **2.2 Equipment**

Although nowadays new DJ interfaces are going out to the market (i.e. compact disc players or computer media players) in order to develop and experiment new techniques and effects, there are two devices which have always been used in a DJ performance. This “traditional” equipment consists of the turntable and the audio mixer. The first one is

where the vinyl record is put on. The audio mixer incorporates several features, the crossfader being one of the most important and used by DJs.

Below, both the turntable and the audio mixer are described in a detailed way, paying special attention to the vinyl record and the crossfader since their importance in a scratching performance.

## 2.2.1 Turntable

The turntable (*Figure 1*) is the part of the DJ equipment used to reproduce recorded sound. As has been said above, the main function of a turntable is to reproduce music, so it is where the vinyl disc record (*Figure 2*) is placed. There are two types of turntable; the direct-drive turntable and the belt-drive turntable. The difference between them is in the model of motor used. Here, the first model is the one described since it is the most used one by DJs. This type of turntable has the platter mounted on the motor. This has been the most significant drawback that direct-drive turntables have always had on account of the vibration of the motor on the vinyl record. However, the development of a shock-absorbing material during the last years placed between the motor and the platter has reduced the vibrations.



Figure 1. Turntable

The most important parts of a turntable besides the platter and the vinyl record are: the needle, the pick up mechanism (*Figure 3*) and the pitch controller.

The needle is the part which contacts the vinyl record to read the signal and makes possible the reproduction of the sound. It incorporates a little device known as the pick-up mechanism specially designed “for scratching to prevent the diamond stylus (needle) from skipping from one groove to another” [6]. Finally, the pitch controller is used by DJs “to “tune” the sound sample to the tonality of the piece to which they are scratching “[7].



Figure 2. Vinyl record



Figure 3. Shure M44-7 cartridge

The functioning of a turntable is relatively easy. First, the motor drives at a continuous speed the platter where the vinyl record is placed. Then, the needle converts the mechanical movement into an electric signal which presents the same variations of the groove. This electric signal is then first sent to the audio mixer and to the loudspeaker later.

### **2.2.2 Audio Mixer**

The audio mixer (*Figure 4*) is placed between the turntable and the loudspeaker. As stated, the turntable sends the audio signal to the audio mixer, which deals the signal by using its different controllers. Once the signal has been dealt, is sent to the loudspeaker and reproduced for the audience.

An audio mixer has several controllers in order to deal with the signal and obtain the desired result. Among these controllers, the most important are the crossfader, a line/phono switch, a volume fader and one or more knobs for equalization [7].

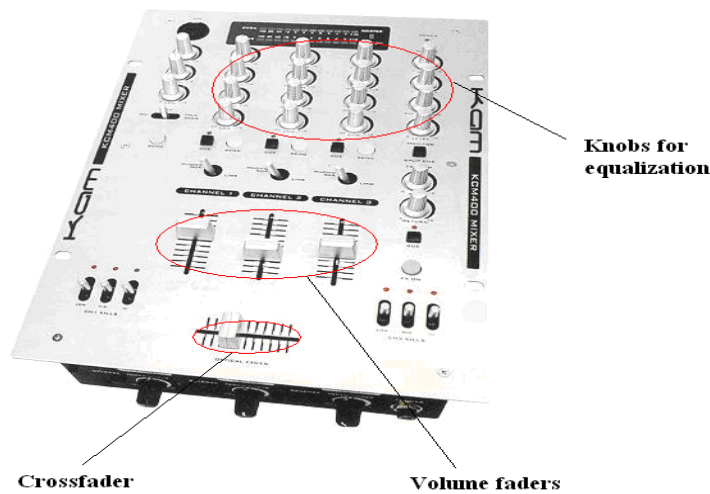


Figure 4. Audio mixer

As shown in *Figure 4*, the controllers are placed strategically to make things easier to the DJ. Since the crossfader is the most used controller by DJs, it is also the nearest one to their hands. The crossfader makes the DJ able to control the volume of both turntables (right and left) at the same time. If the controller is completely on the left end, only the sound of the left turntable is reproduced by the loudspeaker. Accordingly, if it is completely on the right end, the sound of the left turntable is silenced and only the sound of the right turntable is heard. Thus, when the controller is anywhere in the middle of the fader, one of the two turntables will sound louder than the other unless the crossfader is absolutely in the middle. For example, in *Figure 4*, as the crossfader is slightly placed on the left, it is possible to approximate that the left turntable has a 70% of the total volume and the right turntable a 30%. Nevertheless, as it is explained in more detail in 2.3, the crossfader is mainly used to silence completely one of the two turntables.

To control with more accuracy the volume of each turntable, the volume faders are used. Normally, an audio mixer has two or three volume faders, one for each turntable. The one

placed on the left (labeled channel 1), controls the left turntable volume and the one on the right (labeled channel 2), controls the right one.

An audio mixer includes some knobs for equalization. There are many kinds of equalization, and it is feasible to find audio mixers with different equalization controllers. DJs use these controls to control and change the pitch shifting of the sound. This can be done by adjusting the level (gain), amplitude, bandwidth and center frequency [8].

## **2.3 Scratching**

Scratching is a technique used by DJs, also called turntablists, which consists in moving back and forth a vinyl disc on the turntable. From an audio processing point of view, it just means to vary the speed and the direction of an audio file. By using this technique, DJs create an effect similar to scratch a disc, which well done, allows to create rhythms and melodic phrases.

As it has been mentioned above, DJs use one or two turntables and an audio mixer to perform their sessions. The most important in scratching is the physic contact between the DJ hand and the vinyl record. Using his/her fingers in different positions, they are able to create different scratching techniques.

The other important point is controlling the crossfader while scratching. By changing the position of the crossfader, it is possible to create new and different scratching techniques. For example, some effects are done by silencing the sound (moving the crossfader from the left end to the right end) several times in a quick way. The crossfader is therefore used as a switch used to turn the sound quickly on and off [7].

## **2.4 Techniques**

There are lots of scratching techniques, which differ in the movement of the DJ's hand on the vinyl disc and the position of the crossfader. Thus, by moving back and forth the vinyl disc at different speeds while opening and closing the crossfader (opened means that the sound is audible and closed that the sound is inaudible) different effects are generated.

Below, twelve of the most important techniques are described and represented. In the “record” figures is shown the distance that the vinyl disc is moved back or forth (y axis) and the number of samples (x axis). In the “crossfader” figures the position of the crossfader (y axis) and the number of samples (x axis) are shown. All figures are adapted from Skiproof [9] software.

### 2.4.1 Baby

Baby scratch is the simplest scratch technique. It is performed by moving the vinyl disc back and forth repeatedly while the crossfader is in the open position. In the picture is only shown one back and forth movement.

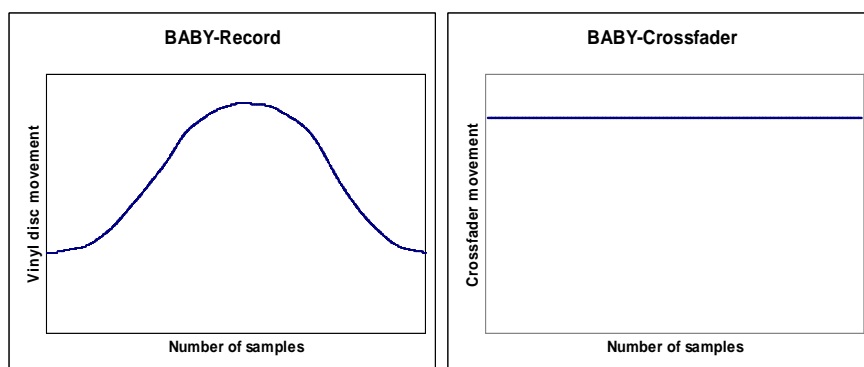


Figure 5. Baby scratch

### 2.4.2 Tear

Tear scratch is another fundamental technique. It is similar to baby scratch, but introduces a quick back and forth movement during the main back and forth movement.

Usually, as *Figure 6* shows, it is introduced in the backstroke. In this technique the crossfader is also in open position.

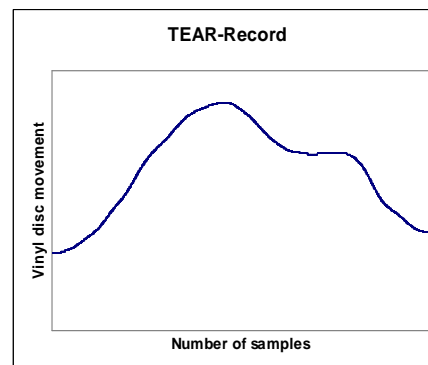


Figure 6. Tear scratch

### 2.4.3 Rolltear

Rolltear scratch combines repeatedly back and forth movements of different distance. While performing them, the crossfader is drastically moved from the closed position to the open position and after some back and forth movements is turned to the closed position again, usually when the last back and forth movement is in the backstroke.

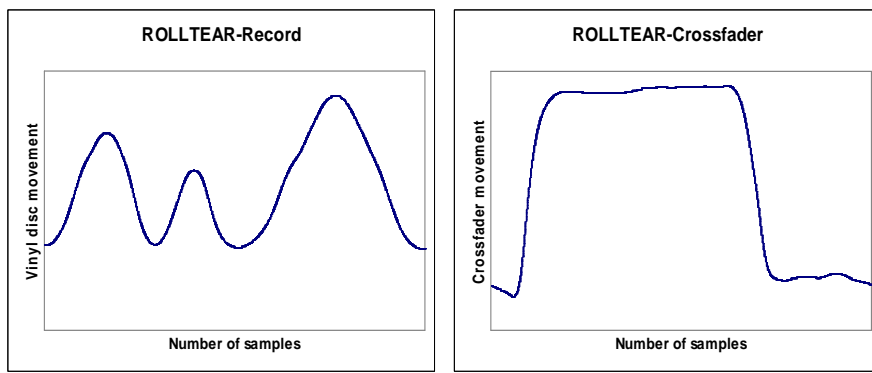


Figure 7. Rolltear scratch

### 2.4.4 Chopshort

Chopshort scratch is a simple back and forth movement like baby scratch. However, in this case, the crossfader is closed during the backstroke.

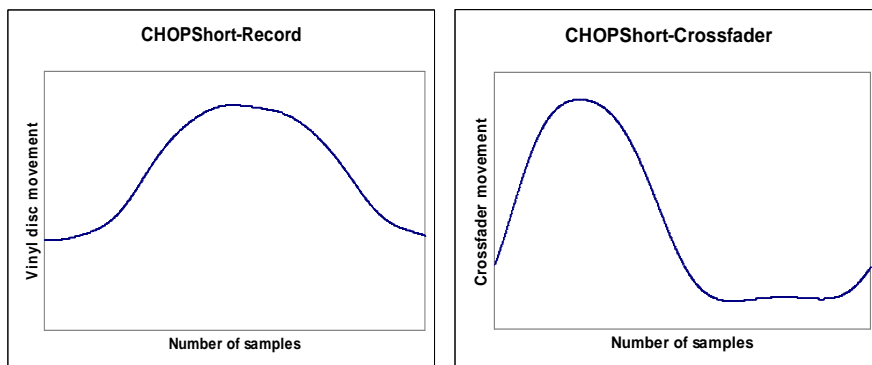


Figure 8. Chopshort scratch

## 2.4.5 Forward

Forward scratch is a baby scratch in which the crossfader is closed during the back movement of the vinyl disc. When the record is moved forth, the crossfader is instantly moved to the open position.

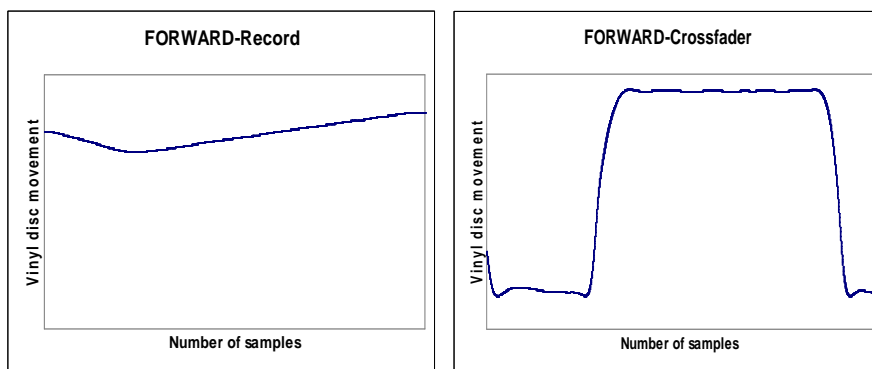


Figure 9. Forward scratch

## 2.4.6 Silent Backdraw

As its name indicates, in silent backdraw scratch the back movement of the record is silenced. The picture shows how in the first samples the crossfader is in the open position and how it is closed when the backward pull is being done.

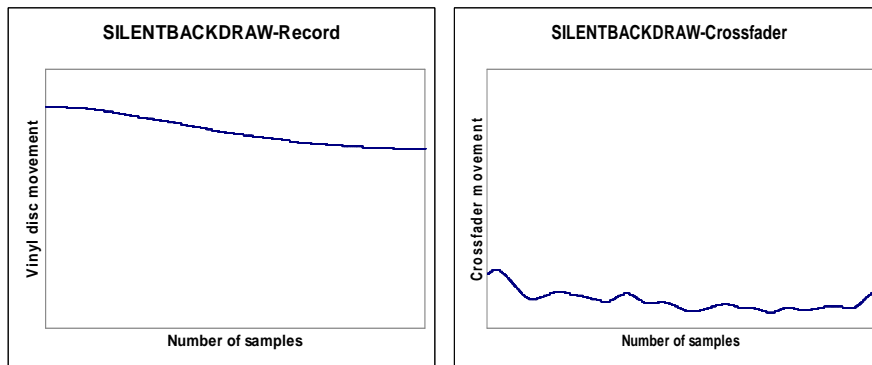


Figure 10. Silent Backdraw scratch

## 2.4.7 Scribble

Scribble scratch is another technique in which the crossfader is not used (open position all the time). It is similar to the baby scratch technique, but in this case the back and forth movements are shorter and quicker. DJs are capable to achieve these quick movements by tensing the muscles of the scratching hand [10].

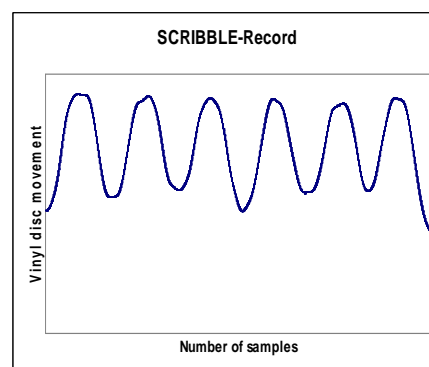


Figure 11. Scribble scratch

## 2.4.8 Uzi

Uzi scratch is one of the most dynamics techniques, where the DJ has both scratching and crossfader hands in constant movement. The movement on the record is a repeatedly back and forth pull and push, similar to the baby scratch. On the contrary in this technique the crossfader plays an important role. While DJ is performing the back movement the crossfader is instantly moved to the closed position and when is doing the forth movement the crossfader is instantly moved to the open position.

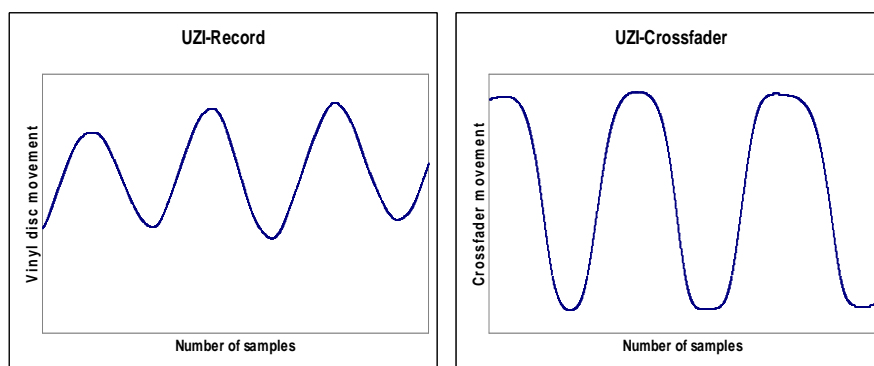


Figure 12. Uzi scratch

## 2.4.9 Chirps

Chirps scratch is a difficult technique to perform due to it takes a good amount of coordination. Firstly the record is moved forward while the crossfader is in open position. Then, in the back movement of the vinyl disc the crossfader is moved to the close position, getting there before the back movement is ending. By doing this in quick sequence DJs achieve a sound similar to a chirp [11].

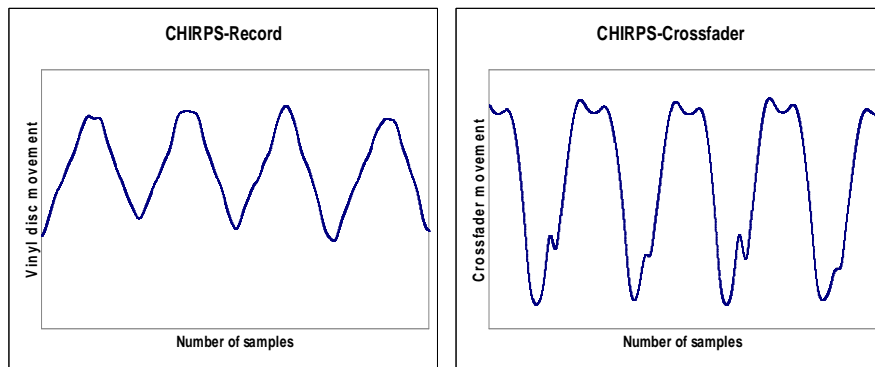


Figure 13. Chirps scratch

## 2.4.10 Flare

Flare scratch begins with the crossfader in open position. Then, the DJ starts moving forward the record, and until the direction has changed to backwards, the crossfader has been moved several times from the open to the closed position, ending in the open position. The same when the record is moved back, ending again in the open position.

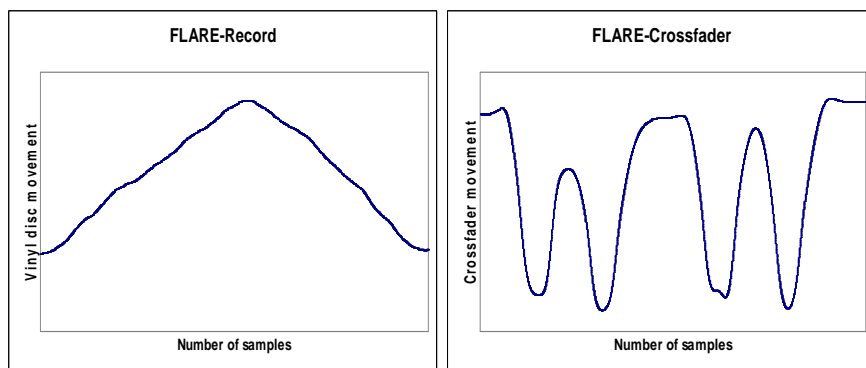


Figure 14. Flare scratch

### 2.4.11 Crab

This technique requires the fastest crossfader movement. While a simple forth and back movement is being done on the record, the crossfader is moved from the open to the close position even four times, standing on the initial position when the DJ is changing the direction of the record.

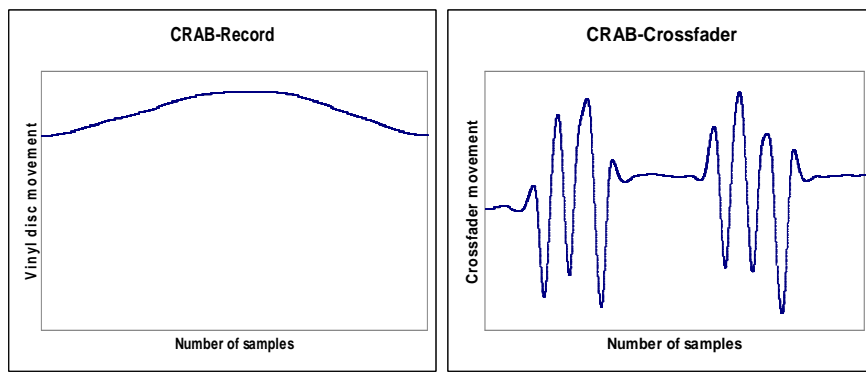


Figure 15. Crab scratch

### 2.4.12 Twiddle

This technique is very similar to Flare scratch. It only varies on the movement of the crossfader.

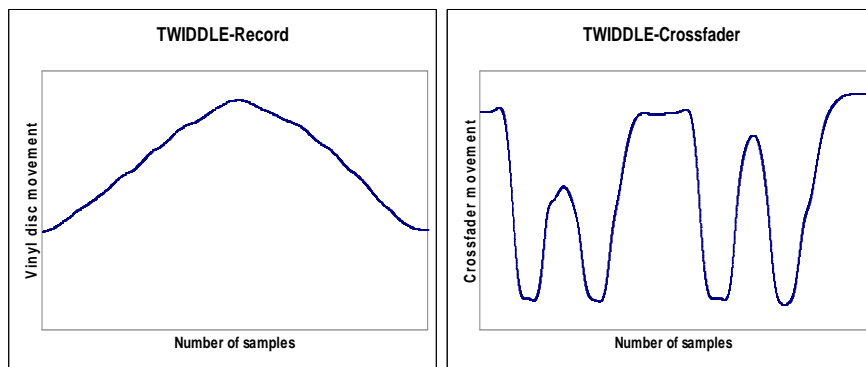


Figure 16. Twiddle scratch

## 3 Software

Different software tools have been used in order to obtain the desired result. In this chapter the different tools are described.

### 3.1 Skipproof and Pure data

Skipproof<sup>2</sup> is a software tool developed using PD<sup>3</sup> (Pure Data, Puckette 1996) and GrIPD<sup>4</sup> (Graphical Interface for Pure Data, Sarlo 2003), which allows the user to simulate around 20 different scratch techniques on the computer. Pure Data (*Figure 17*) is a graphic programming language for sound and music computing applications. Its graphic appearance, flexibility and simplicity convert it in a software fast of prototyping.

The main feature of Skipproof is to simulate the turntable and the audio mixer in only one graphical interface (*Figure 18*). Thereby, Skipproof becomes a starting point and an important tool to this project due to the aim of the project is to develop a tool capable of doing the same as Skipproof but using other tools and being implemented in a mobile phone.



Figure 17. Script in PD ( adapted from <http://upload.wikimedia.org/wikipedia/en/0/0b/Pd-objects.png>)

---

<sup>2</sup> <http://www.speech.kth.se/~kjetil/software.html>

<sup>3</sup> [www.puredata.org](http://www.puredata.org)

<sup>4</sup> <http://crca.ucsd.edu/~jsarlo/gripd/>

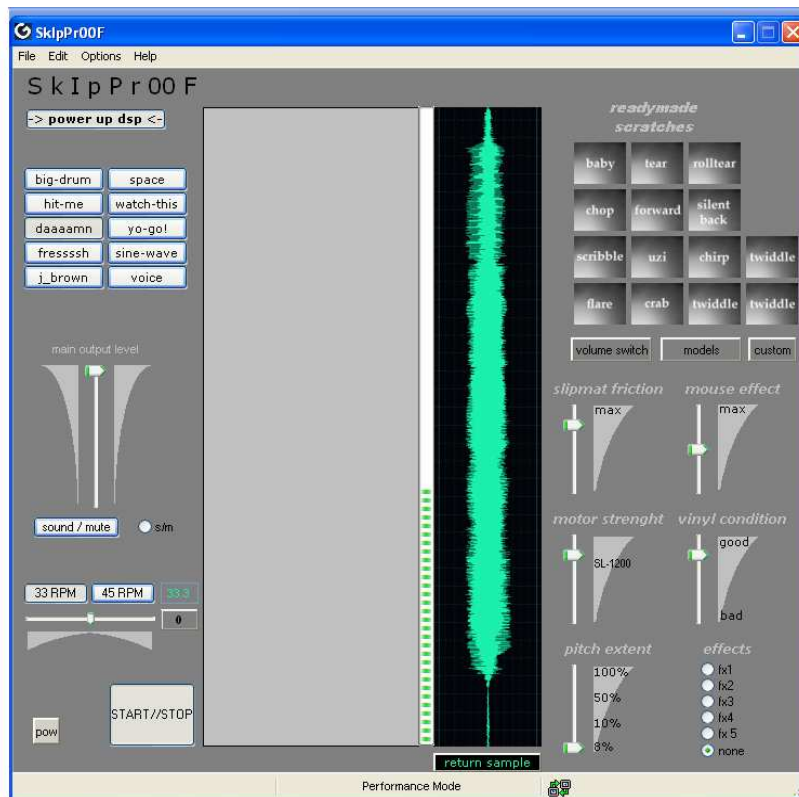


Figure 18. Skipproof interface

### 3.2 Python

The programming language that has been used in this project has been Python<sup>5</sup> (Guido van Rossum 1991). Python is an interpreted programming language that permits the user to understand easily and quickly the written code because of its minimalist syntax and semantics. Since it is mainly used to develop new tools and applications for mobile technology, it has been the programming language elected to work with. Indeed, the large standard library and the great number of modules designed allow the programmer to develop a high number of applications within audio, video and multimedia field (e.g. Games).

Each script in Python does not need to be compiled. The software is available on the official Python website and also on Nokia mobiles phones as an application.

<sup>5</sup> [www.python.org](http://www.python.org)

Once the script is finished using a text editor (e.g. Microsoft Word), the user just have to save the file as “namefile.py” and run the script on his/her mobile phone. It is also possible to use a Nokia emulator to run a script on the computer as it is explained in the next point.

### 3.3 Nokia S60 Emulator (5th Edition)

The main function of the Nokia S60 Emulator<sup>6</sup> is to simulate a Nokia mobile phone on the computer, so the user is able to try his scripts written in Python without having a Nokia mobile phone. As the *Figure 19* shows, Python is installed on the applications menu and ready to read a script.

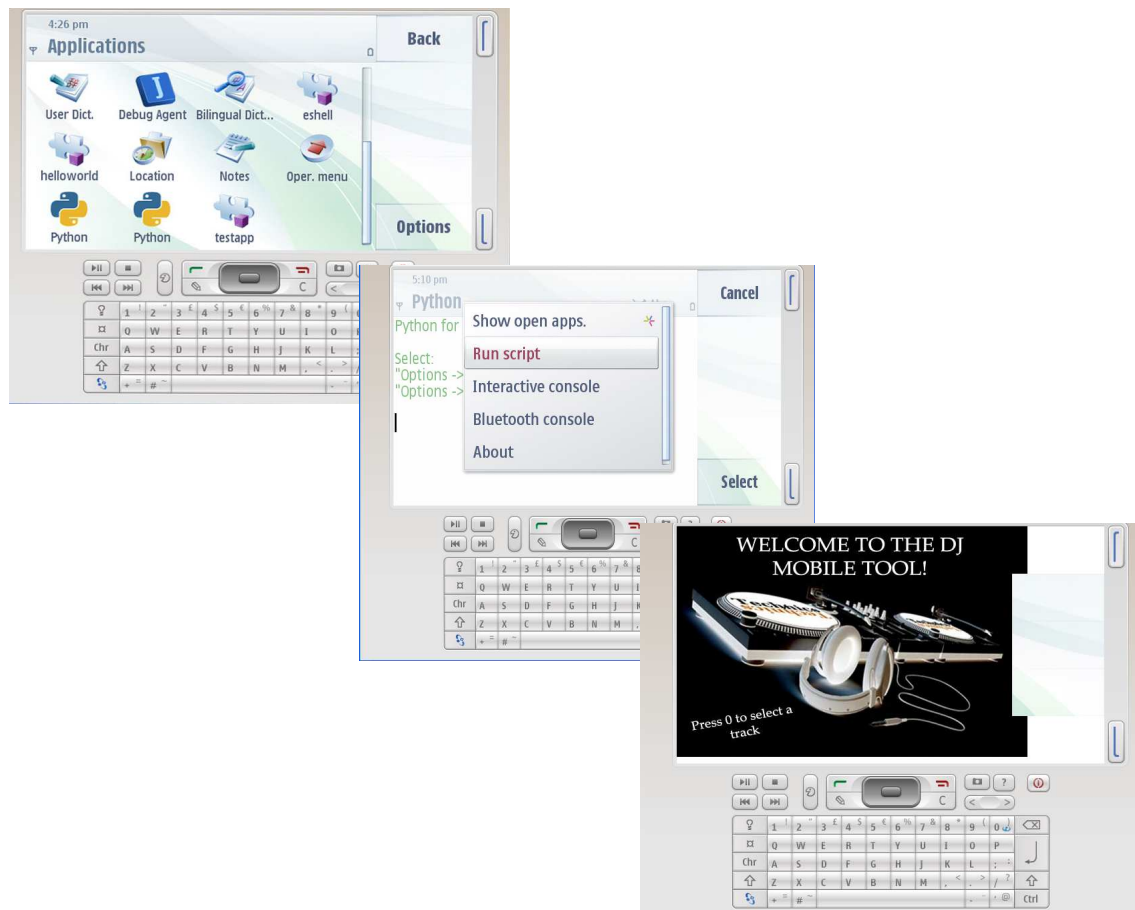


Figure 19. Nokia S60 Emulator interface

<sup>6</sup> www.s60.com



# 4 Implementation

In this chapter the implementation of the project is described. It has been divided into three main sections. Firstly, the different functions of the program are presented. Then it is shown the design of the interface, and after that, it is explained how the sound has been manipulated in order to acquire the desired effects. The code written in Python to develop the application can be found on the “Appendix A” of this work.

## 4.1 Functions

As stated in previous chapters, the aim of this project is to develop an application which enables the user of a mobile phone to simulate different effects being on of them different scratching techniques while listening to the music. However, as a music reproducer, some other basic tools and functionalities have to be implemented in order to be able to control some necessary aspects of a sound reproduction.

The extra functions designed are the next:

- *Play a sound file*

The application is able to reproduce any sound file (WAV format). From the option “Select file” in the menu, the user has access to the music folder of his mobile phone and can select any audio file from there. Once the file is selected is instantly played.

- *Stop a sound file*

The audio file that is being played can be stopped whenever the user wants from the options menu.

- *Control the volume*

The user is able to control the volume of the sound file. To turn the volume up is used the right key and to turn it down is used the left key.

- *Mute*

The audio file can be automatically silenced by using the mute option. If the audio file is being played, it can be muted by pressing the “sharp” key (#). Then, if the same key is pressed again, the sound is back.

- *Switch*

This function allows the user to silence the audio during 0.1 seconds by pressing the “star” key (\*). After this time the volume is re-established. This tool acts like a switch or a crossfader moved quickly from the open to the close position.

- *Control of the sample rate*

The sample rate of the audio file can be changed. The standard sample rate of every audio file on the playlist is 44100Hz. By pressing the indicated keys, the sample rate adopts the indicated values on the next table.

Fast - 60000 Hz (key 9)
<b>Normal - 44100 Hz (key 8)</b>
Slow - 22000 Hz (key 7)

Since the changing of the sample rate is not in real time, once this function was developed every time the sample rate was changed the audio file started from the

beginning, fact that was not desired. To solve this setback, a function that keep the position of the song before changing its sample rate in order to start there once it has been changed was programmed (see Appendix A).

In this case these values can also be changed by the user by varying the script. Not all the frequencies may work; it depends on the sound card of each device.

- *Move the audio file forward*

This functionality allows the user to move the audio file forward. By clicking the up arrow on the keyboard, the song is moved a preset time (one second) forward.

- *Move the audio file backward*

In this case, the audio file is rewind a preset time (one second) every time the down arrow is pressed.

## 4.2 Interface

As the mobile phone application that it is, the interface of this project has been an important point to have on count. It is essential that the software works correctly, but it is also required, or should always be, to accompany a nice software application with an attractive interface design. Appearance is always important in multimedia applications. Python has some tools used to deal only with the graphical part of any application. One of the most used is PyGTK<sup>7</sup>, which allows the programmer to create graphical interfaces with Python language. However, this application is not able to work neither on the emulator nor the mobile phones. Thus, the interface has been created by using the simplest tools and options that PyS60 allows. Accordingly, the efforts have been put on to design an attractive background and an easy and understandable handling of the options and functionalities.

---

<sup>7</sup> <http://www.pygtk.org/>

In this chapter a written tutorial of the functioning of the program is described, showing its different menus and backgrounds.

### Starting the application

The first touch with the application, once the script has been ran, it is a screen giving the welcome to the application (*Figure 20*), which indicates to press the “green” key to jump to the options menu.



Figure 20. Welcoming screen

Once the “green” key has been pressed appears the menu that allows the user to select the audio file to be played. From this menu the user can also leave the application by choosing the “ESC-Exit” option. (*Figures 21 and 22*)



Figure 21. Options menu



Figure 22. Selection track menu

## Main menu

Once a track has been selected, the application jumps to the main menu. From this menu the user is able to use the different tools implemented to manipulate the audio (Figure 23). This menu also offers the possibility to jump to an “Options” menu (Figure 24) by pressing the green right key (see Appendix B. Keys’ function). From there the user can select a new file, stop the sound, go back to the main menu or exit the application.



Figure 23. Main menu



Figure 24. Options menu



Figure 25. Exit menu

## 4.3 Manipulation of the sound

Once the basic audio player tools had been implemented, the main concern was how to deal directly with the audio file. The manipulation of an audio file is always a difficult issue, with mathematics and audio processing aspects behind. Python offers a wide number of music, multimedia and audio modules which have a great number of functions that can allow the programmer to manipulate the sound file. Unfortunately, almost all of them can't be used neither on the emulator nor the mobile phone. This has been an important obstacle during the work, due to the inability to manipulate the signal in real time that it has meant. Below the way how the sample rate is manipulated and how the scratching effects are added when the application is used are described. In both cases, the code developed in python to create these functions is shown (see all code in *Appendix A. Audio mobile phone player script in Python*).

### Sample Rate

An important parameter of an audio file is its sample rate. By changing the sample rate the speed reproduction of the file is changed, which gives an interesting and even a funny effect. For this reason, this has been the first aspect of manipulating the audio file that has been dealt. All the audio files in this project are in "WAV" format, which presents the following structure.

<b>WAV FILE FORMAT</b>			
<b>File offset (bytes)</b>	<b>Field name</b>	<b>Field Size (bytes)</b>	
0	<b>Chunk ID</b>	4	<b>The "RIFF" chunk descriptor</b> The Format of concern is "WAVE", which requires two sub-chunks: "fmt" and "data"
4	<b>Chunk Size</b>	4	
8	<b>Format</b>	4	

12	<b>Subchunk1 ID</b>	4	<b>The "fmt" sub-chunk</b> Describes the format of the sound information in the data sub-chunk
16	<b>Subchunk1 Size</b>	4	
20	<b>Audio Format</b>	2	
22	<b>Num. Channels</b>	2	
24	<b>Sample Rate</b>	4	
28	<b>Byte Rate</b>	4	
32	<b>Block Align</b>	2	
34	<b>Bits Per Sample</b>	2	
36	<b>Subchunk2 ID</b>	4	<b>The "data" sub-chunk</b> Indicates the size of the sound information and contains the raw sound data
40	<b>Subchunk2 Size</b>	4	
44	<b>Data</b>		

Figure. 26. WAV file format (adapted from [http://69.10.233.10/KB/audio-video/Concatenation\\_Wave\\_Files/conca1.JPG](http://69.10.233.10/KB/audio-video/Concatenation_Wave_Files/conca1.JPG))

As shown in the *Figure 26*, between the bytes 24 and 27 the sample rate is stored. Then, by changing this parameter, has been possible to develop a tool in the application which allows the user to change the reproduction speed (see 4.1 “*Control of the sample rate*”).

Code:

#### # Increase the speed reproduction of the audio file #

def increase\_speed(): #Name of the function (In this case, the function that increases the sample rate)

global music, soundfile

b = music.current\_volume() #The volume of the file before changing the sample rate is saved

x = music.current\_position() #The position of the file before changing the sample rate is saved

newsamp = 60000 #New sample rate

f = open(soundfile, 'r+') #The file that is sounding is opened as a file

w = wave.open(soundfile, "r") #The file that is sounding is opened as a wave

a = w.getframerate() #To know the sample rate of the file before changing it

c = struct.pack('l', newsamp) #Converts newsap into binary

f.seek(24, 0) #The byte 24 of the file is searched

f.write(c) #The new sample rate is written in the searched position

f.flush()

```

music = audio.Sound.open(soundfile) #The new audio file with the new sample rate
is opened
music.play(audio.KMdaRepeatForever) #The new audio file with the new sample
rate is played
#Below, if the sample rate before the changing is the normal (44100 Hz) the
position is set using a different parameter than if the previous sample rate was the
slow (22000 Hz).
if a == 44100:
    music.set_position(x/1.36)
    music.set_volume(b)
else:
    music.set_position(x/2.74)
    music.set_volume(b)
f.close()

```

### **Scratching Effects**

As stated in previous chapters, scratching means to change the reproduction speed and the direction of an audio file. Since manipulate the audio directly has not been permitted by the emulator and the mobile phone, other ways to simulate the scratching effects have had to be thought.

Then, the idea has been to add different recorded scratching techniques while the audio file is sounding.

Code:

#### **# Add different scratching techniques #**

```

def effect1():

    global music, L

    a = music.current_volume() #The volume of the file sounding is saved
    music.set_volume(a-2) #The volume of the file sounding is turned down
    L = audio.Sound.open("e:\python\DJTools\baby.wav") #The scratching technique
    is opened
    L.set_volume(a) #The scratching technique volume is set to parameter a
    L.play() #The scratching technique is played
    e32.ao_sleep(2)
    music.set_volume(a) #Once the scratching technique is finished, the file volume is
    set to a again
    L.stop()

```

## 5 Discussion

This has been the first attempt in developing an audio player for Nokia mobile phones with the scratching tool. Nowadays, this application already exists in two devices; the iPhone and the Samsung M7600 Beat DJ. In both cases, the scratching techniques are simulated by dragging the fingers on the touch-screen. From my point of view, and after having tested both devices, simulating scratching by using touch-screen it is not the best way to develop this application.

During the project many problems have arose. Among them, the most important one has been not being able to manipulate the audio file in real time. Anyway, this problem can be solved in the future by creating new audio modules able to work on the emulator and the mobile phone. Another important problem during the work has been not having a mobile phone to be used only for the project. It has meant that I have not been able to try the script on mobile phones as many times as I would have wanted. It would have been really important due that in some cases the same script works different depending if it is tested on the emulator or on the mobile phone. Because of that, many changes have had to be done after trying the scripts on the mobile phone and realizing that the application did not worked as it did on the emulator. This drawback has slowed down the work in some moments.

In conclusion I consider that developing applications which enable users to manipulate the sound they are listening to is a really interesting field to study and to work with. This work, as a first attempt, can help further researches within this field.



## 6 Further Research

For what to the further work respects in the DJ audio mobile phone applications, in first place would be important to find out the way to manipulate the sound file in real time. This could be done by creating new audio, music or multimedia modules which would make the programmer able to use them on the emulator, and so on Nokia mobile phones. Another option would be the use of others programming software, and consequently, to develop the application for others mobile phones brands. Once this application would be designed and implemented satisfactorily, the next step would be how to use it and how to simulate the scratching techniques. In the present work, each of the six techniques are simulated by pressing a different key on the keyboard. In the future, and in order to make the application more real, it would be interesting to design applications which enables users to simulate these techniques by shaking the mobile phone (nowadays almost all mobile phones have an accelerometer) or by using a touch screen, which have not been done on Nokia mobile phones so far.



# References

- [1] Hansen, K. F. (2002). The Basics of Scratching. *Journal of New Music Research*, 31(4), 357-365
- [2] K. F. Hansen and R. Bresin. The Skiproof virtual turntable for high-level control of scratching. Submitted, 2009
- [3] <http://en.wikipedia.org/wiki/Phonograph>, 26 June 2009
- [4] [http://en.wikipedia.org/wiki/Martin\\_Block](http://en.wikipedia.org/wiki/Martin_Block), 18 June 2009
- [5] [http://en.wikipedia.org/wiki/Disc\\_jockey](http://en.wikipedia.org/wiki/Disc_jockey), 26 June 2009
- [6] Hansen, K. F. (2001). Playing the turntable: An introduction to scratching. *TMH-QPSR*, 42(1), 69-79.
- [7] Hansen, K. F., & Bresin, R. (2006). Mapping strategies in DJ scratching. In *NIME '06: Proceedings of the 6th international conference on New interfaces for musical expression*, Paris, France, 188-191
- [8] <http://en.wikipedia.org/wiki/Equalization>, 5 May 2009
- [9] <http://www.csc.kth.se/~kjetil/software.html>
- [10] <http://en.wikipedia.org/wiki/Scratching>, 26 June 2009
- [11] [http://en.wikipedia.org/wiki/Chirp\\_\(scratch\)](http://en.wikipedia.org/wiki/Chirp_(scratch)), 24 April 2008



# Appendices

## Appendix A. Audio mobile phone player script in Python

```
##### FIRST THE NECESSARY MODULES ARE IMPORTED #####
```

```
import appuifw
import e32
import key_codes
import audio
from audio import*
import graphics
from graphics import Image
import wave
import struct
import sys, urllib, urllib2, os
```

```
global music
music = None
```

```
##### INDICATOR OF THE VOLUME #####
```

```
def circles(n):
    y = 250
    x = 120
    for i in range(n): #n is the number of green circles
        img.point((x,y), (0,255,0), width=7)
        x += 10
    for i in range(10-n): #the number of black circles
        img.point((x,y), (0,0,0), width=7)
        x += 10
    handle_redraw()
```

```
#### WELCOMING SCREEN IMAGE ####
```

```
img = Image.open('e:\python\DJTools\INTERFACE1FNOKIA.jpg')
```

```
def handle_redraw(rect):
```

```
    canvas.blit(img)
```

```
appuifw.app.screen='full'
```

```
canvas = appuifw.Canvas(redraw_callback=handle_redraw)
```

```
appuifw.app.body = canvas
```

```
def quit():
```

```
    app_lock.signal()
```

```
#### MAIN INTERFACE ####
```

```
def menu2():
```

```
    global img
```

```
    img = Image.open('e:\python\DJTools\INTERFACE2FNOKIA.jpg') #Replace the  
    welcome image with the main image
```

```
    handle_redraw(()) #Redraw the canvas so the second image is shown
```

```
L = audio.Sound.open("e:\python\DJTools\baby.wav")
```

```
#### EXIT MENU --> FROM MAIN MENU ####
```

```
Listexit = [u"Yes", u"No"]
```

```
def menuexit():
```

```
    index = appuifw.popup_menu(Listexit, u'Really quit?')
```

```
    if index == 0:
```

```
        quit() # Quit from the application
```

```
    if index == 1:
```

```
        menu() # Go back to the menu
```

```
#### EXIT MENU --> FROM MENU2 ####
```

```
def menuexit2():
```

```
    index = appuifw.popup_menu(Listexit, u'Really quit?')
    if index == 0:
        quit()
    if index == 1:
        menuoptions()
```

```
#### OPTIONS MENU ####
```

```
Listopt = [u"1- Select File", u"2- Stop", u"3- Back", u"4- Exit"]
```

```
def menuoptions():
```

```
    global music

    index = appuifw.popup_menu(Listopt, u'Menu Options')

    if index == 0:
        if music is not None:
            circles(music.current_volume())
            a = music.current_volume()
        else:
            a = 0
        open_audiofile(a)

    if index == 1:
        if music is not None:
            circles(music.current_volume())
            stopsong()
        menuoptions()

    if index == 2:
        if music is not None:
            circles(music.current_volume())
        menu2()

    if index == 3:
        if music is not None:
            circles(music.current_volume())
            stopsong()
        menuexit2()
```

```
##### TURN THE VOLUME UP #####
```

```
def VolUp():
```

```
    global music

    music.set_volume(music.current_volume() + 1)
    n = music.current_volume()
    circles(n)
```

```
##### TURN THE VOLUME DOWN #####
```

```
def VolDown():
```

```
    global music

    music.set_volume(music.current_volume() - 1)
    n = music.current_volume()
    circles(n)
```

```
##### FORWARD THE SONG #####
```

```
# Forwards the position of the audio file 1 second #
```

```
def forward():
```

```
    global music

    c = music.current_position()
    music.set_position(c+1000000)
```

```
##### REWIND THE SONG #####
```

```
# Rewinds the position of the audio file 1 second #
```

```
def rewind():
```

```
    global music

    c = music.current_position()
    music.set_position(c-1000000)
```

```
#### SCRATCHING TECHNIQUES ####
```

```
# Add different scratching techniques #
```

```
def effect1():
```

```
    global music, L
```

```
    a = music.current_volume()
```

```
    music.set_volume(a-2)
```

```
    L = audio.Sound.open("e:\python\DJTools\baby.wav") # Opens the scratching  
    technique
```

```
    L.set_volume(a)
```

```
    L.play() # Plays the scratching technique
```

```
    e32.ao_sleep(2)
```

```
    music.set_volume(a)
```

```
    L.stop()
```

```
def effect2():
```

```
    global music, L
```

```
    a = music.current_volume()
```

```
    music.set_volume(a-2)
```

```
    L = audio.Sound.open("e:\python\DJTools\luzi.wav") # Opens the scratching  
    technique
```

```
    L.set_volume(a)
```

```
    L.play() # Plays the scratching technique
```

```
    e32.ao_sleep(0.6)
```

```
    music.set_volume(a)
```

```
    L.stop()
```

```
def effect3():
```

```
    global music, L
```

```
    a = music.current_volume()
```

```
    music.set_volume(a-2)
```

```
    L = audio.Sound.open("e:\python\DJTools\chirps.wav") # Opens the scratching  
    technique
```

```
    L.set_volume(a)
```

```
    L.play() # Plays the scratching technique
```

```
    e32.ao_sleep(2)
```

```
    music.set_volume(a)
```

```
    L.stop()
```

```
def effect4():
```

```
    global music, L
```

```
    a = music.current_volume()
```

```
    music.set_volume(a-2)
```

```
    L = audio.Sound.open("e:\python\DJTools\combo_23.wav") # Opens the  
    scratching technique
```

```
    L.set_volume(a)
```

```
    L.play() # Plays the scratching technique
```

```
    e32.ao_sleep(0.6)
```

```
    music.set_volume(a)
```

```
    L.stop()
```

```
def effect5():
```

```
    global music, L
```

```
    a = music.current_volume()
```

```
    music.set_volume(a-2)
```

```
    L = audio.Sound.open("e:\python\DJTools\crab_flare.wav") # Opens the scratching  
    technique
```

```
    L.set_volume(a)
```

```
    L.play() # Plays the scratching technique
```

```
    e32.ao_sleep(2)
```

```
    music.set_volume(a)
```

```
    L.stop()
```

```
def effect6():
```

```
    global music, L
```

```
    a = music.current_volume()
```

```
    music.set_volume(a-2)
```

```
    L = audio.Sound.open("e:\python\DJTools\sc2.wav") # Opens the scratching  
    technique
```

```
    L.set_volume(a)
```

```
    L.play() # Plays the scratching technique
```

```
    e32.ao_sleep(2)
```

```
    music.set_volume(a)
```

```
    L.stop()
```

```
#### CHANGE OF THE SAMPLE RATES ####  
# Increase the speed reproduction of the audio file #
```

```
def increase_speed():  
  
    global music, soundfile  
  
    b = music.current_volume()  
    x = music.current_position()  
    newsamp = 60000  
    f = open(soundfile, 'r+')  
    w = wave.open(soundfile, "r")  
    a = w.getframerate()  
    c = struct.pack('l', newsamp)  
    f.seek(24, 0)  
    f.write(c)  
    f.flush()  
    music = audio.Sound.open(soundfile)  
    music.play(audio.KMdaRepeatForever)  
    if a == 44100:  
        music.set_position(x/1.36)  
        music.set_volume(b)  
    else:  
        music.set_position(x/2.74)  
        music.set_volume(b)  
    f.close()
```

```
# Decrease the speed reproduction of the audio file #
```

```
def decrease_speed():  
  
    global music, soundfile, audiofile  
  
    b = music.current_volume()  
    x = music.current_position()  
    newsamp = 22000  
    f = open(soundfile, 'r+')  
    w = wave.open(soundfile, "r")  
    a = w.getframerate()  
    c = struct.pack('l', newsamp)  
    f.seek(24, 0)  
    f.write(c)  
    f.flush()  
    music = audio.Sound.open(soundfile)  
    music.play(audio.KMdaRepeatForever)  
    if a == 44100:
```

```

        music.set_position(x*2)
        music.set_volume(b)
    else:
        music.set_position(x*2.74)
        music.set_volume(b)
    f.close()

```

**# Normalize the speed reproduction of the audio file #**

```

def normal_speed():

    global music, soundfile, audiofile

    b = music.current_volume()
    x = music.current_position()
    newsamp = 44100
    f = open(soundfile, 'r+')
    w = wave.open(soundfile, "r")
    a = w.getframerate()
    c = struct.pack('l', newsamp)
    f.seek(24, 0)
    f.write(c)
    f.flush()
    music = audio.Sound.open(soundfile)
    music.play(audio.KMdaRepeatForever)
    if a == 22000:
        music.set_position(x/2)
        music.set_volume(b)
    else:
        music.set_position(x*1.37)
        music.set_volume(b)
    f.close()

```

**#### SELECT AUDIO FILE MENU ####**

```

def menu():

    global soundfile
    appuifw.app.menu = [(("Choose the sound you want to hear!", open_audiofile))]

```

```
#### STOP AUDIO FILE ####
```

```
def stopsong():
```

```
    global music

    a = music.current_volume()
    circles(a)
    music.stop()
    normal_speed()
    music.stop()
    menu2()
```

```
def select_audiofile():
```

```
    global audiofile
    path = "e:\python\DJTools\Music\"
    index = None
    filelist = os.listdir(path)
    ll = []
    for file in filelist:
        ll.append(unicode(file))
    index = appuifw.selection_list(choices=ll , search_field=1)
    if index is not None:
        audiofile = path + ll[index]
        return audiofile
    else:
        return None
```

```
def open_audiofile(currentvolume):
```

```
    global soundfile, music
    soundfile = select_audiofile()

    if (soundfile is not None):

        menu2() # upgrade the background picture
        music = audio.Sound.open(soundfile)
        if currentvolume == 0:
            currentvolume = music.current_volume()

        circles(currentvolume)

        music.set_volume(currentvolume)
        music.play(audio.KMdaRepeatForever)
        normal_speed()
```

```
##### MUTE FUNCTION #####
```

```
# If the volume is more than 0 sets the volume to 0 #
```

```
# If the volume is 0 sets the volume to the current volume #
```

```
def mute():
```

```
    global previous, music
```

```
    current = music.current_volume()
```

```
    if music.current_volume() == 0:
```

```
        circles(previous)
```

```
        music.set_volume(previous)
```

```
    else:
```

```
        circles(0)
```

```
        previous = current
```

```
        music.set_volume(0)
```

```
##### SWITCH FUNCTION #####
```

```
# The sound is silenced during 0.1 seconds #
```

```
def switch():
```

```
    global music, L
```

```
    current = music.current_volume()
```

```
    L.set_volume(0)
```

```
    music.set_volume(0)
```

```
    e32.ao_sleep(0.1)
```

```
    L.set_volume(current)
```

```
    music.set_volume(current)
```

```
##### ASSIGN OF THE DIFFERENT FUNCTIONS TO THE KEYS #####
```

```
canvas.bind(key_codes.EKeyHash, mute)
```

```
canvas.bind(key_codes.EKeyStar, switch)
```

```
canvas.bind(key_codes.EKeyRightArrow, VolUp)
```

```
canvas.bind(key_codes.EKeyLeftArrow, VolDown)
```

```
canvas.bind(key_codes.EKeyUpArrow, forward)
```

```
canvas.bind(key_codes.EKeyDownArrow, rewind)
```

```
canvas.bind(key_codes.EKeyYes, menuoptions)
```

```
canvas.bind(key_codes.EScancodeSelect, menuoptions)
```

```
canvas.bind(key_codes.EKey1, effect1)
```

```
canvas.bind(key_codes.EKey2, effect2)
```

```
canvas.bind(key_codes.EKey3, effect3)
```

```
canvas.bind(key_codes.EKey4, effect4)
```

```
canvas.bind(key_codes.EKey5, effect5)
canvas.bind(key_codes.EKey6, effect6)
canvas.bind(key_codes.EKey7, decrease_speed)
canvas.bind(key_codes.EKey8, normal_speed)
canvas.bind(key_codes.EKey9, increase_speed)
```

```
appuifw.app.exit_key_handle = quit
app_lock = e32.Ao_lock()
app_lock.wait()
```

## Appendix B. Keys' function

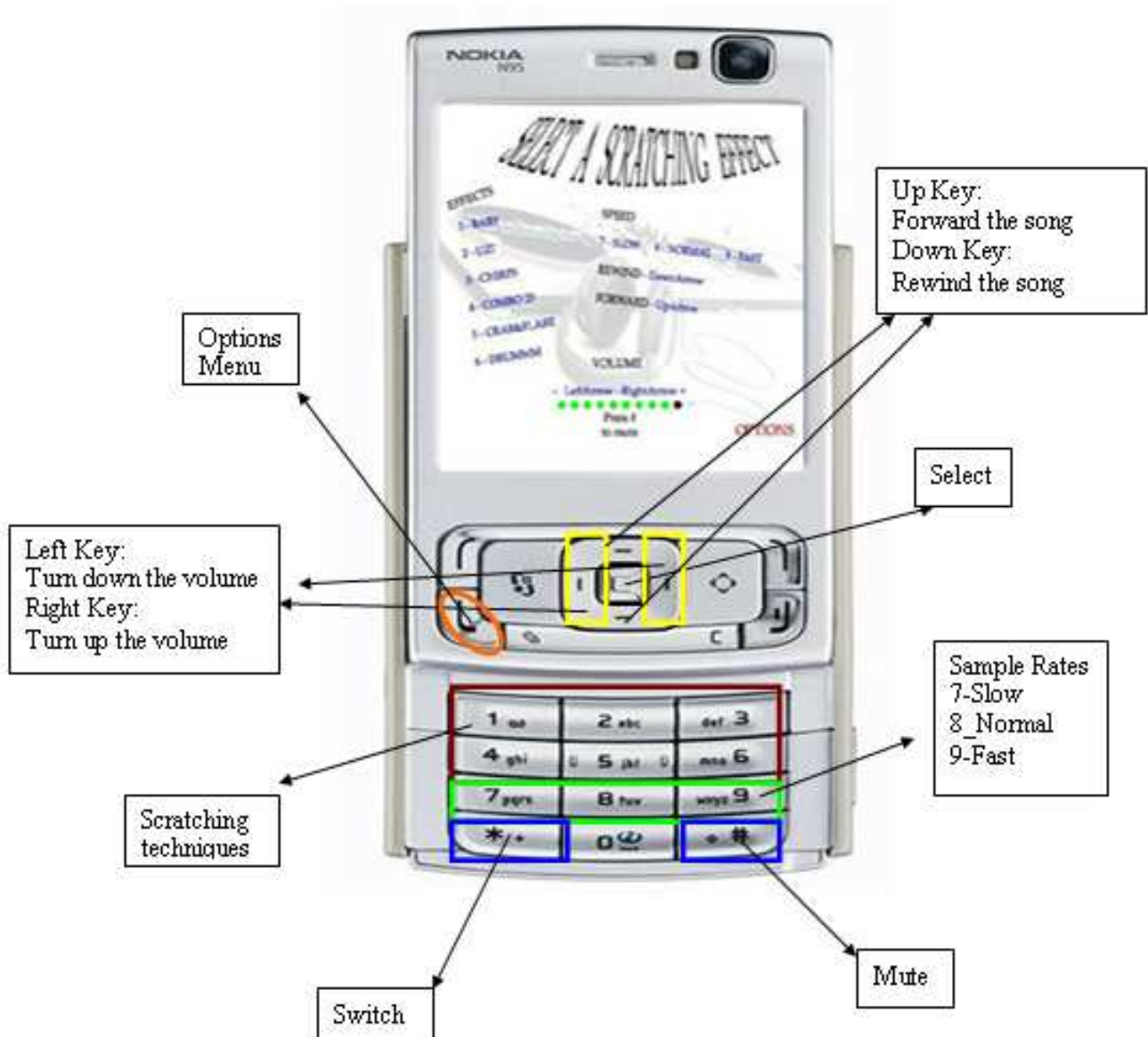


Figure 27. Keys' functions