Smartphone Application Development

Porting existing PC game for multiple platforms

Mattias Öst
Abstract

Smartphone Application Development

Mattias Öst

An already created PC game exists and the goal of this project is to find a solution to port the game to the most popular smartphone platforms using mostly one generic code. The game to be ported is a musical game where the player uses an acoustic guitar to interact with the game using with the help of real time audio analysis. This thesis goes through the development process to reach the objective by investigating proper tools and issues around the performance and human-computer interaction from the perspective of the smartphone. The result of the project is an application prepared for the platforms Android and Ios.
## Contents

1.0 Introduction .............................................................................................................. 2  
  1.1 Problem description ................................................................................................. 2  
  1.2 Limitations ................................................................................................................ 3  
2.0 Background ............................................................................................................... 4  
  2.1 Chord King ................................................................................................................ 4  
  2.2 Smartphones .............................................................................................................. 4  
  2.3 Libraries .................................................................................................................... 5  
    2.3.1 Cocos2dx ............................................................................................................. 5  
    2.3.2 LibGDX .............................................................................................................. 5  
  2.4 Music theory ............................................................................................................ 6  
    2.4.1 The guitar ........................................................................................................... 6  
3.0 Method ...................................................................................................................... 8  
  3.1 Chosen limitations .................................................................................................... 8  
  3.2 Tools ........................................................................................................................ 9  
    3.2.1 Eclipse ............................................................................................................... 9  
    3.2.2 Android SDK .................................................................................................... 9  
    3.2.3 LibGDX ............................................................................................................ 10  
    3.2.4 XML ................................................................................................................ 11  
  3.3 Implementation ....................................................................................................... 12  
    3.3.1 Application design .............................................................................................. 12  
    3.3.2 Interface design ................................................................................................. 13  
    3.3.3 Graphics ........................................................................................................... 15  
    3.3.4 Audio analysis ................................................................................................. 16  
4.0 Result ....................................................................................................................... 18  
  4.1 The application ........................................................................................................ 18  
5.0 Discussion ............................................................................................................... 21  
  5.1 Future improvements ............................................................................................... 21  
6.0 Conclusion ............................................................................................................... 22  
References ..................................................................................................................... 23
1.0 Introduction

Learning a new musical instrument can be a tough process, a lot of people who are struggling with the instrument eventually give up before they are comfortable playing it. It may be boring, hard or both. One solution to this problem is to combine the learning process with gaming entertainment. A prototype PC game already exists with the preferred functionality required to play with an acoustic guitar. The goal is to port the current version to the most popular handheld devices. The application which will be created in this project aims for both new- and experienced guitar players.

1.1 Problem description

The objective of the project is to port the current PC version of Chord King [1] to the smartphones, preferably with the operating systems Google Android and Apple Ios.

To accomplish this a few problems have to be solved. The signal processing will be adjusted to match smartphone properties. The GUI¹ will be redesigned since the users will mainly use the touch screen to interact with the menu system instead of the regular mouse and keyboard, which were used in the original version of the game. The application needs to have a good design that fits different screen sizes and adjusts properly independently of the screen size, while maintaining good space allocation with consideration of HCI² for smaller screens. In addition to the problems just mentioned the project needs to investigate if the smartphones has enough computing power to perform intense real time audio analysis during the game play. The analysis is also in need of improvement since the original game had some issues under certain circumstances where it could not differentiate notes which shared similar frequencies [1].

For example the problem between two different triad chords which both had three frequencies where one or more of those were close enough to the other chord frequencies. In this situation the program had some issues to decide which one of the chords the player actually played on the guitar. As a result of the problem description above, the goal of the project is to evaluate if there exist tools to be able to keep the development mainly to one generic code and port it while solving the problems mentioned and achieve support for multiple platforms. Another important aspect is to examine if the solution can maintain performance on all platforms and if the hardware of the smartphones have enough computing power for the real time audio analysis.

¹ GUI, Graphical User Interface
² HCI, Human Computer Interaction
1.2 Limitations

The primary goal is to develop a working application for both Android and iOS. However, if no suitable solution can be achieved within the timeframe of the project, the development will mainly be focused on creating a working application for the Android platform. The graphics in the game will not be prioritized but should be at least of the same quality as the PC game. The graphics part of the development process will be limited to adjust for the new screen sizes that the game will be played on. Although if there is some time left at the end of the project, some graphic upgrades may be considered. The audio analysis of the guitar notes will be limited to the ones in the original version which are the regular chords in major and minor version, A, B, C, D, E, F, G.
2.0 Background

2.1 Chord King

Chord King is the name of the prototype PC game which is going to be rewritten for handheld devices such as smartphones and tablets. It was developed by physics students in 2011 for their bachelor project with the goal of solving the problem on how to make a program to recognize different acoustic guitar notes [1]. The vision for the game was to teach the players to play the guitar using simple assignments in game and when they had enough experience they could advance to real songs. However the game has not been used by others than the developers since the original development. The source code is written in Python which is a scripting language with great portability for different operating systems. The language itself is known for making programming fast in terms of functionality with limited lines of code while maintaining its readability [2]. However it will not be used for this project since it does not have good support for the targeted platforms Android and Ios.

Chord King is not a regular game, it combines learning with entrainment and is a good tool for guitar players. The game shares some similarities with games like Guitar Hero [3] and Frets on Fire [4]. In all of those games the player has hit as many notes as possible to achieve the goal and set new high scores. What differentiates Chord King among the others is the use of an acoustic guitar instead of a plastic game guitar with buttons or keyboard. To be able to use an acoustic guitar audio analysis has to be made in real time while rendering the game graphics and verify if the sound recorded by the application is in fact one of the supported notes in the game and if it was played at the right time.

2.2 Smartphones

A smartphone is a phone with more computing capabilities than a regular phone and it includes much of the same functionalities as a desktop PC. However the size is significantly smaller and the user mainly interacts with a touch screen or a small keyboard, even though the small keyboard exists the touch screen seems to be the winning concept by looking at the current market. The smartphones with touch screens have increased in popularity over the last decade. Sales have been increased rapidly since Apple launched their first iPhone in 2007 to the public [5]. The year after the release of the first iPhone version competitors started making phones similar to the iPhone with touch screens and the global sales for all smartphones in 2008 were measured to 140 million units\(^3\) [6]. Today, in the year of 2013 analysts estimate that the global sales of smartphones will reach the 1 billion mark [7].

\(^3\) Unit, as in one smartphone
seen the smartphone market is growing rapidly. So if the applications created today in general have good support for the popular operating systems, the exposure to potential users is great. The leading operating systems in this space are Android developed by Google and Ios which is developed by Apple [8].

2.3 Libraries

The goal of the project is to develop a solution for both Android and Ios since they are the market leaders which means that the application will reach out to most users if it supports both of those operating systems. Therefore a cross platform library with the primarily focus on Android and Ios is the ultimate choice if the application is going to be released commercially. There are a lot of libraries available in this space and they all have different properties. The libraries that has been considered for this project is Cocos2dx [9] and LibGDX [10] which are primarily made for game programming. Both of the libraries have big communities with active forums and decent tutorials to get started which is great for new developers in this area.

2.3.1 Cocos2dx

Cocos2dx is an extension from the older library Cocos2d which was only for the Ios platform. Cocos2dx gives the developer the opportunity to choose among different languages for example C++, Lua or Javascript to develop the application. However it lacks somewhat in the documentation compared to LibGDX. Although it contains a lot of features for game development, graphics, physics, gui classes and several other tools [11].

2.3.2 LibGDX

LibGDX is another library with multiple support for different platforms, it has decent documentation and is written in Java for most of the part. Java is not the best language in terms of performance but LibGDX do not suffer too much since all critical parts where performance is at stake the code is written in C/C++. LibGDX has Java as the language for development and has similar features like Cocos2dx except some interesting parts in the audio section which will be of good usage for the project. It includes optimized methods for audio analysis such as DFT⁴ [12] and FFT⁵ [13]. As mentioned earlier this is some of those critical parts where LibGDX uses

---

⁴ DFT, Discrete Fourier Transform  
⁵ FFT, Fast Fourier Transform
C/C++ instead of Java [14]. Another good feature is the ease to setup a working development environment in Windows 7 using LibGDX compared to Cocos2dx.

### 2.4 Music theory

Music is made by notes which is made by various music instruments. When the instrument is played the sound produced is made up from vibrations which gets distributed through air or water etc with different pressure. Those vibrations are measured per second in the unit Hz\(^6\) and are called the frequency of the note. The human processes those vibrations through the ear drums and sound can be heard [15]. A typical healthy person can hear frequencies in the range between 20-20000Hz, this varies more with older persons where a decline in the scale is considered normal [16].

In the western music system the notes are divided into the chromatic scale which means that the notes are divided into intervals where each one is called an octave. The notes in each octave is named after the first seven letters in the alphabet, C, C#/Db, D, D#/Eb, E, F, F#/Gb, A, A#/Bb and B, the # is called a *sharp* and the corresponding b is called a *flat*. As shown some tones have different names but they share the same sound. Every octave has semi tones which are the smaller interval between one note and the following note. To make the frequencies officially standard the ISO\(^7\) has set the A4 to 440hz, A4 is short for the fourth octave in tone A [17]. For example the A3 is 220hz and A5 is 880hz, as seen in the example the frequencies halves or doubles. Notes does also have overtones, in the case of A4 the overtones are 880hz, 1320hz, 1760hz and continues upwards. This shows that every tone has overtones which are multiples of the original frequency, this will be essential to remember since the analysis will refer to this theory. Lastly a guitar example, when the guitar is played correctly the sound we tend to hear is the original frequency with a mix of overtones which makes each guitar sound unique [18].

#### 2.4.1 The guitar

A regular guitar has six strings named E low, B, G, D, A, E and when the guitar is tuned in the most common way the strings produce the tones E4, B3, G3, D3, A2, E2. To play different notes, a fretboard\(^8\) is used. The frets are the metal strips along the neck of the guitar and are used to adjust the sound of the note played. The finger

---

\(^6\) Hz short for Hertz
\(^7\) ISO, International Organization for Standardization
\(^8\) Fretboard, the neck of the guitar with the frets
should be used on the specific string with pressure on the area between the strips. The frets are placed along the board so each of the fret position represents a semi tone. To play more custom notes on the guitar the same technique is used but on several strings, called chords. There are various types of chords and one of the most common ones are the triad chords, they contain three different tones and are the type of chords the application aims to support.
3.0 Method

The development for this project used an agile approach. There are a lot of different agile methods available which are used in various projects. They are popular since it makes the team well prepared for rapid changes during the development stage and the product is always in a state where it is working. The chosen method for this project is parts from Scrum [19] since the full Scrum approach cannot be reached with only one developer and a short term project like this. Scrum approaches the problems through an iterative and incremental process to control the risk and predictability of the project. A Scrum project uses a backlog which is divided into stories, preferably described from the perspective of the end user. The stories are in turn divided into moderate tasks that will be prioritized and planned for the next sprint. This enables the whole iterative development process and the progress is shown in a burn down chart which makes it easy to see if the project is on track or falling behind. There are several roles within a Scrum project a product owner, scrum master and development team [19]. For this project the scrum master and product owner was assigned to the original developers of the PC version and there was only one developer assigned to the development team, myself.

The development process was divided into 3 sprints.

- Sprint 1 was aimed to setup the skeleton code for the overall design of the application, load all assets and make a decent package structure.

- Sprint 2 was focused to implement a fully working interface by doing a market research of popular applications and creating and choosing the design for all the screens in the menu system while following HCI guidelines.

- Sprint 3 was to implement the game play including the audio analysis.

3.1 Chosen limitations

The project was focused on to find a suitable solution for both Android and Ios but since there was no Ios device available during the development phase all the testing was only made on the Android platform. Therefore the application is only well prepared for Android at the current stage. Additionally the goal of the project was to make a prototype with working audio analysis so graphic improvements were not considered a high priority and only the basic notes were implemented.
3.2 Tools

The tools and development environment for the project was a Windows 7 PC using the Eclipse IDE [20] for coding. Other tools and libraries was the Android SDK\textsuperscript{9} [21], ADT\textsuperscript{10} [22], LibGDX and XML [23].

3.2.1 Eclipse

The Eclipse IDE is a develop environment primarily for Java with support to integrate the ADT. This integration adds different configurations related to Android programming in Eclipse and also support for both emulators and phones plugged in using USB cable.

3.2.2 Android SDK

Android SDK is used to create applications for Android devices and is also required when developing applications with LibGDX. To understand how Android applications work, the application lifetime needs to be investigated which includes the cycle for each activity in the application. An activity is usually a class which presents parts of the application to the user and implements in general several or all of methods onCreate(), onStart(), onResume(), onPause(), onStop(), onDestroy() to be able to handle all of the events which can appear on a smartphone.

- onCreate() is called on startup, used to initialize the application.
- onStart() is called when the application becomes visible to the user.
- onRestart() is called after the application has been stopped and going to start again.
- onResume() is called when the user launching the application again and is done with the event which caused the pause in the first place. For example a phone call or another application which got more priority.
- onPause() is called when there is some kind of interruption for the application, for example a phone call or the user presses the home button on the phone.

\textsuperscript{9} SDK, Software Development Kit

\textsuperscript{10} ADT, Android Development Tools
- `onStop()` is called right after the `onPause()` method and when the application is no longer visible to the user.

- `onDestroy()` is called when the user exits the application and all the memory used by will be released.

![Activity Cycle Diagram](image)

Figure 1: The activity cycle [24].

The **Figure 1** represents a diagram of activity cycle and how the activity moves between the states, when and where each of the methods is called [24].

### 3.2.3 LibGDX

LibGDX was the library of choice for this project. It was chosen over Cocos2dx because it has a bigger community and nice features and easy to use with Windows 7 and therefore more suited for the project. LibGDX supports all the different aspects of game development, graphics, audio, input, file io and a application framework with a design that reminds of the regular Android SDK activity cycle. The
application framework is used to handle the main loop and the life cycle with the supported methods create(), resize(), render(), pause(), resume() and dispose() which have the same purpose as in the original Android activity cycle described above except the render() and resize() which is called for rendering graphics in the application respectively resizing.

3.2.4 XML

XML, short for Extensible Markup Language is used to store data while maintaining the object structure. XML uses tags to separate the elements from each other and makes it easy for both computers and humans to interpret the data [23].

XML was used to save all the data which had some sort of value to the application or the user experience such as songs, high scores etc.
3.3 Implementation

The implementation is described in four topics, application design, interface design, graphics and audio analysis.

3.3.1 Application design

To be able to develop for the platform of choice a good solution is offered by LibGDX to deploy the application to multiple platforms. To achieve this the several projects have to be created with one starter class in each project and one greater home project where all the generic code is created for usage over all platforms. The following code shows how the starter class is designed for the Android version.

```java
package com.ChordKing;
import android.os.Bundle;
import com.badlogic.gdx.backends.android.AndroidApplication;
import com.badlogic.gdx.backends.android.AndroidApplicationConfiguration;
import com.chordking.ChordKing;

public class MainActivity extends AndroidApplication {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        AndroidApplicationConfiguration cfg = new AndroidApplicationConfiguration();
        cfg.useWakelock = true;
        cfg.useGL20 = true;
        initialize(new ChordKing(), cfg);
    }
}
```

It imports all the Android assets and also the starter class from generic home project. Which in this case is `com.chordking.ChordKing`, used to be able to create a new instance of the class in the home project. The `cfg` of type `AndroidApplicationConfiguration` is the basic configuration for the application. It is used in all the projects but with different type depending on the target platform for the class. In this case the Android version contains only two configurations.

- `cfg.useWakelock`, when it is set to `true` the application forces the screen to be on. This is crucial for the application since during the game play there is no touch interaction from the user to the screen. Instead of regular touch events the user uses the guitar to interact with the game.

- `cfg.useGL20`, sets OpenGL 2.0 either on or off, for this application it is always on.
The home project which contains the class *ChordKing* is the core of the whole LibGDX application. This is the class which gets instantiated in all the starter classes for each platform. It needs to implement the *ApplicationListener* interface or extend the *Game* class which itself implements the ApplicationListener. The class which implements the ApplicationListener interface is called once the application is created, resumed, rendered, paused or destroyed. The LibGDX lifecycle is essential for mobile applications and works similar as the Android lifecycle. The difference between extending the Game or just implement the ApplicationListener interface is some additional features like the *Screen* support which gets added if Game is extended. Screen is an LibGDX interface which is used to implement different screens in the game, this project uses several screens for the menu system and game play. The menu system screens is pre loaded in the application startup class which extends Game. It acts like a screen manager which switches between the screens and does not re create new screens for the menu system each time the user navigates through the application. Every screen is pre loaded except for the one which handles the game play. The problem here is to find some ratio between memory versus performance and help the garbage collector. The application also loads all available assets needed for the game at launch such as songs, textures, fonts etc. Since operating on files is very expensive this is only done at startup for performance reasons later on and a to get a smooth user experience.

### 3.3.2 Interface design

The interface of the original version was made for PC and not suited for smartphones. However if the interface was just ported as it was issues like too small buttons and lack of space would be some of the problems. The design process started by checking the guidelines to create interfaces set by Norman and Nielsen [25]. They help the designer to make good choices during the design stage. The design principles are divided into three main topics *learnability*, *effectiveness* and *accommodation*. The topics are summarized below.

- **Learnability**, to make sure everything is visible so the users can easily see what functionality the program offers. Use symbols and language that the intended users should be familiar with. Make everything clear to the user by organizing objects and their properties in the interface so it can be related with their functionality and how it should be used.

- **Effectiveness**, provide support for users to navigate through the interface and make it easy to understand when and where the user is in control with the help of good feedback.

- **Accommodation**, make the interface flexible, stylish and attractive. The overall design should always be aimed to be polite to the user.
A market research of currently popular gaming applications was made to see how their interfaces were built and to get the feel of what the regular user was used to. This was a very important part of the design process to be able to achieve one of the important design principles. By reviewing the top lists of gaming applications in the market patterns could be found among the popular applications. Rough sketches were drawn for the project. The main menu and song/level menu is shown below.

![Chord King](image)

Figure 2: The main menu sketch.

As seen in Figure 2 the main menu contains big buttons compared to the screen size and are a common thing among the popular applications in the market. The symbols for the buttons are standard symbols (front arrow, back arrow, question mark) and are widely used even outside of applications. However the placement of the symbols are more common to smartphone applications. Those are the elements for the main menu which follows the only design principle which can be achieved with static grayscale sketches, *learnability*. The user should be familiar with the main menu right after the first start.
Figure 3: The song/level menu sketch.

The big boxes in middle of Figure 3 are supposed to be different levels. A typical smartphone is limited to the screen size and a lot of level choosers use the finger to drag on the screen, called swipe or finger swiping to choose among the levels. It was an idea which originated from the market research. The sketch uses the same style for the same back button as the main menu and also contains some check boxes. This screen is not as simple as the main menu but still follows the same design principle as the main menu and should be fairly easy to understand for the user. The other design principles effectiveness and accommodation were carefully thought of during the development stage of the design process and achieved programmatically and graphically.

3.3.3 Graphics

LibGDX offers several ways to draw graphics on the screen, this project uses the ShapeRenderer and SpriteBatch. The ShapeRenderer is a class which is used to draw different shapes with basic properties such as filled, lined, position, size and color. An alternative way to render graphics is to use the SpriteBatch class which uses images loaded into memory. Using the SpriteBatch is better for performance reasons than the ShapeRenderer and can be even more improved by instead of binding each texture to the SpriteBatch a so called TextureAtlas can be used. It is a picture file containing multiple images to be drawn. This allows the SpriteBatch to only bind one texture to itself but can still draw different graphical objects by drawing specific regions of the file [26]. This feature is not used for this project, the graphics used during the game play is simply drawn by a ShapeRenderer except for the background
and a few other small objects which were drawn using the SpriteBatch and was bound separately. It is not recommended to use the ShapeRenderer for more than debugging purposes since it is significantly slower than drawing graphics using a SpriteBatch. However, since the project did not prioritize the graphic part the ShapeRenderer was fine. Although it is one of the things that will be added for the future work since it will upgrade the application in terms of performance and graphics. All the rendering responsibility during the gameplay is dealt to one single class the GameRenderer, it grabs all objects to be rendered from the GameWorld which updates the game state every frame and contains all the information needed. This style makes it easy to implement more objects and features for future development.

### 3.3.4 Audio analysis

To be able to do the real time analysis, the audio recording has to be made during the gameplay. It is recorded in a separate execution thread which record the sound as a collection of amplitude values. Each value in the collection is called a sample. By just recording and collecting samples the data is represented in the time domain. To be able to do the analysis the representation has to be in the frequency domain which is achieved by performing the Discrete Fourier Transform algorithm or the equivalent more improved version called Fast Fourier Transform which is used in this project. The new representation contains the magnitude of the frequencies [13]. The LibGDX extension library for audio analysis offers several implementations of the Fourier Transform and after some testing and comparison between the implementations significant frame rate drops could be seen. The project ended up with the fastest from the testing process, KissFFT\(^{11}\) [27] which is a library that is implemented using the programming language C.

The length of the sound which is recorded is crucial for the accuracy of the analysis. The problem by recording in real time during the game play is the more samples the more delay for the rest of the game to update the game state. For this project the recording occurs every 256ms which is the result from 2048/8000 where the dividend is the buffer size and the divisor is the sample rate. As a result the accuracy is roughly calculated to 1/0.256 or 3.9hz which leads to an representation of the spectrum where there is 3.9hz between each point.

To resolve the problem with overtones as mentioned in the music theory section 2.4, the Harmonic Product Spectrum algorithm is applied. The algorithm works by knowing that the spectrum of a note is represented as a series of peaks where each peak is a new overtone, hence the name Harmonic. There are two main parts of the

\(^{11}\) KissFFT, Keep it simple stupid Fast Fourier Transform
method, down sampling and multiplication. The down sampling is made by copying the array and keep every other element and every third element etc. As a result the copies of the spectrum gets compressed with the left over elements filled by zeroes to keep all arrays of the same size. By multiplying all the spectrums with each other the remaining array corresponds to the peaks of the fundamental frequencies\textsuperscript{12} and limits the noise of the recording [28].

At the end of the analysis the peaks gets measured by their magnitude and a strong magnitude is a sign of strength in the note. To decide which of the notes that has been played the application takes average of the magnitude of all frequencies for the current note. In addition the application also takes the maximum value from the harmonic product spectrum in consideration to differentiate notes with similar frequencies. The application currently only supports some the notes with a maximum of three frequencies, which are the triad chords.

---

\textsuperscript{12} Fundamental frequency, the original frequency
4.0 Result

The resulting application from the project is a fully working prototype which can calculate and differentiate the supported notes with better accuracy than the PC version. The code is prepared for cross platform and should support both Android and iOS but as mentioned earlier the testing of the application was limited to the Android platform since no iOS device was available. The game has been played and tested on Samsung - Galaxy S and Galaxy S3. This shows that even old devices can do well with the audio analysis. This was one of the possible issues mentioned in the problem description given the heavy computations which had to be made for the analysis. During the testing phase the significance of an fairly quite environment showed up, noise and talking loud while playing did interfere with the application. When application runs as intended under good circumstances the smartphone microphones could recognize the supported notes up to 4-5 meters which is a pretty good since the intended usage is closer to the smartphone. However the result may not be limited to that particular length since it was the size of the room during the testing.

4.1 The application

The application is summarized with the different screens as topic for presentation of functionality and design.

Figure 4: The main menu.

The Figure 4 shows the main menu with simple temporary graphics. The overall layout matches the sketches drawn from the interface design with the one big title and button placement where one bigger play button is in the middle and smaller buttons on the sides for exit and help screen.
As seen in Figure 5 the result of the level menu is close to the sketches drawn in the design process. The user can simply choose the speed to adjust the difficulty of the song by touching the checkboxes. To choose level the user has to use finger swiping in the height of the transparent grey boxes where the levels are supposed to flow in the direction of the swipe. Each level box contains information about the current song. This is a popular style to choose levels among a lot applications in the current application market and this style takes care of the limited screen size. When the user has chosen the song to play by swiping to correct level a simple press on the small play button inside the grey box will transfer the user to the game play screen.

The last one represents the game play of the application, notes moves from right to left and the green box is the area where the user should have played the correct
note. If the application could not match the sound with the note then it is set to failed and the note color is switched to red or the corresponding green if the application in fact did match the sound with the correct note. Up in the left corner is the score with a multiplier which gets multiplied as soon as the user hits one or more notes in a row. As seen in Figure 6 the note graphics stretches over the virtual guitar strings to be played on for the current note. Although the note name is not implemented yet since the graphics drawn using the ShapeRenderer discussed in the Graphics section. For this prototype the only thing that differentiates the notes are their unique color. However new graphics is of high priority for future improvements since it will be an upgrade for both graphical- and performance reasons by using regular images.
5.0 Discussion

The project goal from the problem description was to port the PC version of Chord King as a smartphone application preferably for both iOS and Android and improve the analysis. To be able to port the game the library LibGDX was used described in the background section. As mentioned the application is currently only tested on Android devices with good result and there should not be any problems to make it up and running for iOS without too much work in the starter class. My impression of the libraries LibGDX and Cocos2dx is that LibGDX was originally made for a lot of platforms except iOS which has been implemented lately and the opposite for Cocos2dx which leans more to the iOS platform. So as a developer with the problem of choosing the right library to develop for multiple platforms some time and preparation should be spent to decide which of the platforms that should be the main target to deploy for is a good start.

The market for smartphones is increasing rapidly as discussed in the background section. The application market for the smartphone is a fairly new market that did not exist couple of years ago with a lot of opportunities. As a developer the best idea is to release the application to all popular platforms which is faster achieved with cross platform libraries for example the ones presented in this thesis, though they were more aimed for game development.

This project led me to design and create an application for the smartphones which I was new to and it was a good learning experience. The development process used parts from the agile approach Scrum discussed in the method section, it was not the first time I experienced the Scrum approach but it is a good habit to get used to since it is widely used in projects today. The game Chord King is now compatible with smartphones running iOS and Android and the sound analysis of works great and even performs well at distances as far as 4-5 meters using the regular smartphone microphones. However one fact to keep in mind is that the testing was only made on phones from one manufacturer. To optimize and be more accurate in the terms on how the analysis works in general, more phones from different manufactures should be included in the testing to see if the microphones differs much in comparison to each other.

5.1 Future improvements

Improvements for the application is to add real songs to the game, improve the graphics and as just mentioned in the discussion the testing should be made on more devices to get an overall view on how the game works in regards of the microphones. Another feature which would improve the social aspect of the application would be to implement communities such as Facebook and Twitter to the game and keep a global high score list.
6.0 Conclusion

In general the application works as intended and needs some improvement but the goal described in the problem description was achieved. The application is prepared for cross platform usage. The problem related to hardware got answered after the first testing phase of the audio analysis and the smartphones today is more than enough to perform real time analysis with an algorithm which embeds the Fast Fourier Transform. The analysis also successfully got more improved than the original and can now differentiate the problem notes with a good success rate where the notes shared similar frequencies. The GUI got redesigned to fit smartphone screens with the help of the guidelines and the market research described in the interface design section. So to summarize the game Chord King now works and performs well on smartphones and with just a guitar and a smartphone the user can start playing and experience a new portable way to learn and improve the guitar skills.
References


