Virtual DJ equipment

Marcus Lundin

April 4, 2009
Master's Thesis in Computing Science, 30 ECTS credits
Supervisor at CS-UmU: Jan-Erik Moström
Examiner: Per Lindström

Umeå University
Department of Computing Science
SE-901 87 UMEÅ
SWEDEN

Abstract

The goal of this project was to develop a digital alternative to a DJ system using vinyl records. A regular DJ system consists of two vinyl record players connected to a mixer which in turn is connected to an amplifier. This setup has some major disadvantages. Primarily the cost of the equipment and the cost and effort needed to procure the large amount of vinyl records needed. A large amount of vinyl records also is neither very portable or easy to keep track of.

The purpose of this project is to create a prototype for a DJ system using compressed audio(MP3's) on a computer hard drive instead of vinyl records. The goal was to create a system that could deliver adequate performance for home use or less demanding professional use with a lower cost and the easier handling of compressed digital audio.

Contents

List of Figures

Chapter 1

Introduction

The purpose of this project was to develop a digital alternative to a DJ system using vinyl records. In a professional setting like a club or dance hall, the disk jockey or DJ often works with a system consisting of two specially made vinyl record players and a two channel mixer tying them together. This setup allows the DJ to use several types of sound manipulations and effects(see chapter 2). Such systems are usually only used in professional settings or by people with a budget to match their interest. The purpose of this project is to create a similar system using compressed audio files on a computer instead of vinyl records. The use of compressed audio on a computer has several upsides compared to the traditional vinyl counterparts, these will be handled later in the report. This project consists of the development of a functional prototype of a DJ system. The focus of the project is the software used to play audio files. The project also consists of some hardware but these are very crude to show a proof of concept.

1.1 Motivation

The primary reason for creating a computer based DJ system is to get a cheaper and more accessible alternative to it's vinyl counterpart. Hopefully the virtual turntables would be cheaper to produce than the vinyl ones, and digital compressed music is certainly cheaper and more accessible than vinyl. Because of the limitations of digital audio the virtual turn tables will not rival the real thing for quite a while but should be a good alternative for people with a smaller budget. Similar units using CD's[4] have been around for a few years, and lately some similar units to what is described in this project[5] have also come out.

1.2 Report Outline

This thesis report describes the design, implementation and the final result of the project. To assist readers not familiar with a regular DJ system a chapter is dedicated to clarify this. There is also a chapter discussing the basics of digital audio and issues relevant to the development of the project. An overview of the thesis follows below.

- Chapter two, Classical DJ system: This chapter describes a regular DJ system. As the system is designed to emulate such a system anyone not familiar with a regular DJ equipment should read this before continuing with the rest of the report.
- Chapter three, Digital audio: This chapter gives some background about digital audio in preparation for chapter six but also for other parts of the report. It gives a short description of digital audio and associated issues relevant to this project. It requires some technical insight.
- Chapter four, Problem description: This chapter describes the goals of the project and the criteria for achieving these goals.
- Chapter five, Design: This chapter describes some of the major design decisions of the system, this chapter demands some knowledge of computer science to be fully appreciated.
- Chapter six, Digital audio resampling: This chapter describes the methods tried and explains the final algorithm for resampling digital audio. Readers not familiar with digital audio should read chapter three first.
- Chapter seven, Accomplishment: This chapter describes the work in detail and lists the most challenging problems of implementing the system.
- Chapter eight, Results: This chapter describes the system and how well it met the goals of the project. Parts of this chapter requires some programming knowledge.
- Chapter nine, Conclusions: This chapter describes the major limitations of the system as well as interesting possible future enhancements of the system. It ends with some personal conclusions about the project.
- Chapter ten, Acknowledgements: This chapter acknowledges the people who helped complete the project.

Chapter 2

Classical DJ equipment

A description of a DJ's work and equipment is given in this chapter. For readers not familiar with what a DJ does and what kind of equipment they use a description is given in this chapter. The whole project is based on emulating a proper DJ system to a large extent. Understanding of what the equipment is made up of and how it is are used is essential to understanding this report.

A DJ's basic equipment consist of two vinyl record players connected to a 2-channel mixer. The mixer is then connected to a pair of headphones, an amplifier and some speakers. The record players will be referred to as turntables in the rest of the report. Here follows a short description of the equipment used and the most important techniques used. There are many more techniques and special hardware[3] but this chapter will only describe the basics.

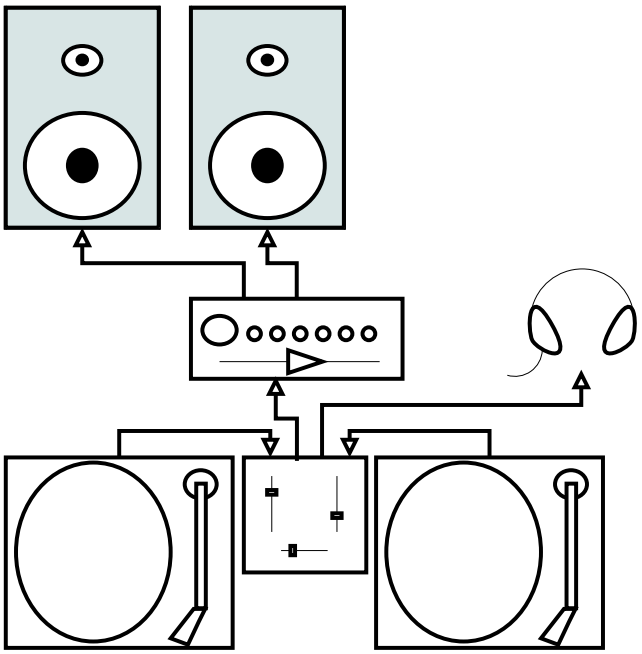


Figure 2.1: DJ system

2.1 Turntables

The turntables used are more durable and powerful than regular record players as they have to handle a lot more manhandling from their users. They also usually have a pitch control built in that allows the user to change the constant speed of the turntable.

2.2 Mixer

The mixer is a 2-channel mixer that takes in the audio signals from the two turntables. It then mixes them at set levels to a single signal that is then fed to the amplifier. The mixer controls usually consist of two level sliders used to set the sound level of each turntable, and one cross-fade slider that allows you to change the relative levels of the two channels. The cross-fade slider allows you to "slide" between full level of the left and zero level on the right one and vice versa. The mixer also has a pair of headphones attached, these can be set to output the signal from one of the two turntables. The mixer is then attached to a amplifier and some kind of speaker system.

2.3 Pitch control

The DJ can use the pitch settings on the turntable or simply rotate the record faster or slower with his/her fingers. Rotating the disk faster will result in a higher pitch and faster playback, slower will cause a lower pitch and slower playback.

2.4 Seamless mixing

Seamless mixing or cross-fading is a technique of changing tracks without a pause or jump in the beat. While one track is being played on the PA system, the DJ synchronizes the beat of another track on the second turntable. This is done by listening to the track on the PA system and listening to the track that is going to be next through the headphones, the DJ then perhaps has to pitch the track for the beats to be compatible and synchronizes the beats. When the two track are synchronized, the DJ simply lowers the level of the track being played and raises the level of the next track thereby achieving a seamless change of tracks. The level change can also be done using the cross-fade slider.

2.5 Scratching

Scratching is a technique used to produce a sound effect by quickly moving a turntable and thereby producing a high pitched noise. It is usually done by having a beat playing on one turntable using the other turntable to "scratch", using the cross-fade slider to control when the noise from the scratching should be heard over speaker system.

2.6 Adding samples

A common use of a DJ system is to add samples from the non-playing turntable to the currently playing track. If for example a track has an interesting saxophone section that the DJ wants to play over another track. To do this the DJ simply searches the sample out on the non-playing turntable and holds the turntable still. The DJ then lets it go and raises the level on that turntable to get the sample to play over the current track at an appropriate time.

Chapter 3

Digital audio

To understand the issues dealt with during this project a basic understanding of digital audio is required. This chapter will give a brief description of both compressed and uncompressed digital audio. It will also describe some of the relevant issues of working with digital audio.

3.1 Digital audio

An analog audio signal is made up of a continuous signal of almost infinite resolution[17]. To convert the analog signal to digital form the continuous curve is replaced with discrete samples at uniform distance(at least in this system). Because you replace infinite resolution with finite resolution and approximate the curves between the samples the end product is an approximation of the original sound.[17] See figure 3.1.

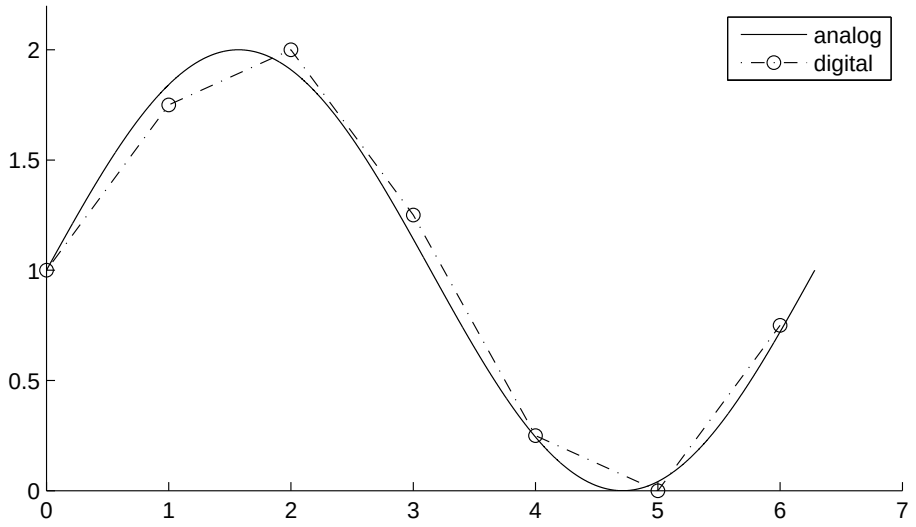


Figure 3.1: Digitized audio signal with 3 bit sample depth(8 values) 1 Hz sample-rate

3.2 Sample rate and bit resolution

The quality of the sound approximation is primarily determined by the sampling frequency and the bit resolution. Sample frequency is simply how many samples taken during one second. What a good sample rate is depends on the application, a standard sample rate for audio is 44.1 kHz. This frequency is chosen because it can represent frequencies up to 22 kHz according to Nyquist's theorem. 22 kHz is the at the upper limit of human hearing. Bit resolution is the resolution of each single sample in bits. The higher the resolution the smaller the rounding fault from the conversion between the analog and the digital value. The bit resolution used by this system is 16 bits/sample, this can represent a discrete number between 0 and 65535. For every extra sample the noise level from quantification(see 3.4) is halved[17]. For this application a higher sample rate and resolution would have been preferred, unfortunately the MP3[7] format is usually coded in 44.1 kHz and 16 bits/sample. For simple playback these sample rates and resolutions are adequate but because this application does resampling on the data and not just playback, higher values would give better results(see sections 3.4 and 3.5).

3.3 Compressed audio

This system uses compressed audio, this is simply compressed digital audio. How this compression is done is not relevant to this project. Uncompressed stereo audio sampled at 44.1 kHz and using 16 bits/sample consumes approximately 10 MB/minute, compressed audio can be as small as 1/10 of the original size[17]. Uncompressing the audio is not a trivial operation, it takes a lot of computer time. Almost all commercially available sources of digital audio, for example CD's, are encoded with a sample rate of 44.1 kHz and bit resolution of 16 bits[13]. Because of this most compressed audio is encoded with 44.1 kHz and 16 bits resolution. The most popular type of compressed audio is the MP3[7] format used in this project.

3.4 Quantification noise

When digital audio is sampled from a source with different resolution(see section 6) the samples usually have to be rounded. (see figure 3.1). This rounding operation is called quantification and adds noise[13]. This system usually resamples the audio to sample rates that require quantification and thereby adds quantification noise. Potentially the resampling process can double the quantification noise. To try to minimize the chance of audible repeated noise the rounding is done by a dither algorithm. This algorithm rounds up or down using a random number and the distance to the closest points, this adds white noise but reduces the chance of harmonised noise(called granulation noise), wich is more audiable[11, 12]. A technique called noise shaping[8, 12] to move the noise into an inaudible spectrum was considered but ultimately deemed too much work. Using a larger bit resolution could compensate for the quantification noise added, a single extra bit would compensate for the added quantification noise. Unfortunately most MP3's are only 16 bit.

3.5 Noise added by resampling

Resampling is a process of making a digital audio from another digital audio source with different sample rate(see chapter 6 for more on resampling). The resampling process usually involves quantification(see section 3.4). Resampling unfortunately adds even more noise than what is was created by the quantification process. Since digital audio data only describes discrete samples the data between the samples must be approximated[17].The errors in the approximation becomes incorporated in the resampled data. In this application the points between samples are approximated by a linear algorithm(figure 3.1). Using higher sample rates could compensate for this by reducing the size of the gaps but depending on the resampling factor audiable noise might still appear. Unfortunately most MP3's use 44.1 kHz sample rate.

Chapter 4

Problem description

The projects purpose is to create a prototype for a DJ system using compressed audio files on a hard-drive instead of vinyl records. The system is designed to emulate a regular DJ system with two turntables and a mixer as well as taking advantage of the more manageable digital audio to handle the tracks being used. The goals are primarily to create a proof of concept of the setup, but the system is designed to be a useful prototype if possible. Below follows the goals of the project.

4.1 Goals

The first goal was to create a system which could properly simulate a pair of vinyl turntables. Secondary goals was to have a as low latency as possible through the system, this is the major issue to compete with existing similar systems. The system should also preferably be usable, and not just an unusable proof of concept. Below follows the primary goals in descending order of importance.

4.1.1 Functionality

The functionality goal is defined as being able to play two tracks on two turntables as if they where played on regular DJ equipment. This goal does not include performance and usability aspects, only a proof of concept.

4.1.2 Usability

To reach the goals of usability the system has to be usable in the context it was design to be used. It should have enough performance and functions to be useful for playing music at a party or ideally to be useful in a professional setting. To do this functions to manage a playlist is needed as well as enough performance to give acceptable audio output.

4.1.3 Performance

Performance mainly refers to the latency from manipulating the control disks to when the response comes from the speaker. The latency should be as small as possible, this requires high performance from the system to process the control inputs to the resulting audio output. Another performance goal is to have the input reading and sound manipulation as accurate as possible, at least to the point where audible errors are not created.

Chapter 5

Design

The design chapter will briefly explain the external design of the system and then go through the internal design of the software. Finally a short explanation of the GUI is given. The major driving forces behind most design decisions were usability, latency and ease of implementation.

5.1 External design

The external design of the system was mostly dictated by the goal of emulating a regular DJ system using two turntables and a mixer. The system would need something to emulate traditional turntables to allow the DJ to manipulate pitch in the traditional way. These virtual turntables would have to be connected to the computer in order to manipulate the digital audio. To be able to see what tracks are playing and control what tracks would be played next a GUI was also needed. In a regular DJ system the DJ can simply read of the album cover or the record itself to find out what track will be played. This system uses an external mixer instead of mixing ditialy in the computer for several reasons. The primary reason for choosing to use an external mixer is that the hardware needed to replace the mixer would be to complicated to build. Also mixing the two audio streams in the computer would add another step of audio manipulation and would slow the system down. See figure 5.1.

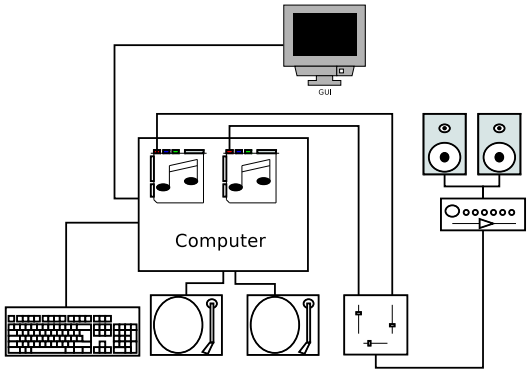


Figure 5.1: External design of the system

5.2 Turntables

The function of the virtual turntables is to allow the user of the system to control the playback of audio tracks. The virtual turntables should ideally work in the same way as a regular turntable to allow a DJ to easily migrate from a standard DJ system, but other designs were also considered. The simplest design considered was to have non-motorized disks that would only allow you to change the speed of playback by moving the disks clockwise or anticlockwise to speed up or slow down playback. This was quickly abandoned because of its limited usability.
The next design considered was a motorized disk. This design works almost like a regular turntable except that the DJ is not able to place the needle were he wants on a record. To select a track the DJ must use the keyboard controls and to search through a track the DJ must spin the disks. It is not possible to simply place the needle in the middle of a track.
To get around the issues with motorizing a disk, "virtually spinning" turntables were considered. A virtually spinning disk is stationary but has a display that is spinning. The display could for example be LED lights set around the disk lighting up in sequence to represent the spinning disk. The turntable is considered to be spinning at the same speed as the display until the DJ moves the disk, the speed is then taken from the actual disk. The disk would have to be pressure sensitive to be able to know the difference between a the disk being held still by the DJ and a "virtually spinning" disk. This design was dismissed because of the complexity of building it, and foreseen issues with the transfer from "virtually spinning" to an actually spinning disk.
The chosen design was a motorized spinning disk, both because of its usability and the simplicity of constructing one from an old record player. The turntables where simply constructed by taking and old record player, placing a mouse pad on the disk and then suspending an optical mouse above the spinning disk. The optical reader should rather have been placed below the disk but this was not realized to make the turntables easier to make. Other forms of sensors where also considered but because of the price, simplicity and availability the optical mice where chosen.

5.3 Controls

In addition to the turntables, the controls for play, pause, stop and changing tracks are done from a keyboard. To search, add and remove tracks in the playlists a keyboard interface using a console window was considered. This was unfortunately never realized because of time issues. The buttons where originally intended to be placed on the turntables but this was also never realized.

5.4 Software

The whole system is designed to run on a PC using Debian[2] Linux. It's designed to run without a graphical system (i.e. X[10]). This is to allow the system to take advantage of as much memory and CPU as possible. The choice of Linux was made because of easier access to low level libraries for critical operations.
The software is written in C++ to get good performance and good access to low level libraries. C++ was also chosen because it is an object oriented language, this aided the modular design and allowed simplification without loosing performance. Debian Linux and C++ was also the environment most familiar to the programmer. Below follows some of the bigger design decisions of the software system.

5.4.1 Audio decompression

The capabilities and complexity of the audio decompression library had some major impacts on the design of the system. The decompression is fairly slow and can not be done fast enough without buffering on the current hardware. Starting from a selected point in a file and decompressing in reverse proved very complex and could not be implemented. Both these issues where solved by using a large buffer for the uncompressed audio. As the decompression is to slow to do on demand, decompressing the whole track before playback was tested. This unfortunately created a long pause before playback started. Finally a compromise where an initial part of the track is decompressed before playback and the rest during playback was chosen. With this solution there is a possibility of buffer under-run but only in extreme cases. The system is designed to be able to play several different formats, but only the module for playing MP3's was implemented. This mainly because of the popularity of the MP3 format.

5.4.2 Cues

A cue is a point in a track which is playable at the push of a button. A system of multiple cues was considered, either from any track or only from the current track. Having cues from multiple tracks would have complicated the software considerably and was therefor dropped. It would also either need very large buffers or the capability to decompress audio in reverse (see section 5.4.1). Multiple cues for only the current track was not considered worth the effort for the limited added usability. A simple pause function was used to implement a single cue in the current track.

5.4.3 Audio buffer

A buffer was needed both for performance and to allow the system to access the audio data in reverse order. A design with a circular buffer was considered to allow seamless track changes, allow for very long tracks and the ability to use smaller buffers. A major issue with circular buffers was that they required reverse decoding of compressed audio, something that turned out to be very complicated with the library that was used(see section 5.4.1). Apart from the ability to play very long songs and use less memory the advantages were negligible. With the current cost of memory the use of a circular buffer is not necessary. A large regular linear buffer was used instead of a circular buffer and seamless track changes abandoned. Uncompressed stereo audio at 16 bit/sample and 44.1kHz sample-rate consumes approximately 10 MB/Minute[17].

5.4.4 GUI

The GUI was not a priority and was designed to only have the displays necessary for use. A fancy GUI was not the goal of this project. The decision to be able to run the system without a graphical window system demanded a graphical library that could accommodate that. The graphical library chosen was the SDL library[9]. This library also handles keyboard input. See figure 5.2.

Figure 5.2: GUI with playlist, current song and elapsed time.

Chapter 6

Digital audio resampling

A major part of the project was to find a way to play digital audio at different speeds. To play sounds faster or slower it is necessary to play more respectively less samples per second than the audio data was intended to be played. In essence: if you play twice as many samples per second the sound will be played twice as fast.
Unfortunately the common hardware can not play samples at any specified speed[17]. The alternative is to instead of playing samples at different speeds change how many samples the audio is made up of. Effectively audio that is reduced to half its original samples played at normal sample rate will play twice as fast as the original audio. This way the speed of the audio can be altered without changing the speed in which the samples are played by the sound-card. The process of creating audio with different sample rates than the original is called resampling[16, pg. 751].
Several methods of resampling the sound where experimented with until a method was found that introduced a minimum of noise into the manipulated data. The method had to be able to resample the data at close to any sample rate. Below follows some concepts of resampling and the methods explored in the project are explained. Last is a description of the algorithm finally chosen.

6.1 Up-sampling and down-sampling

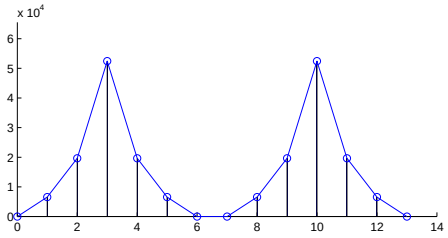
Resampling to a higher or lower sample rate is called up-sampling respectively down-sampling[14, pg. 387–389]. Up-sampling gives slower and lower pitched audio and down-sampling gives higher pitch and faster audio. Down-sampling always leads to a loss of data because samples have to be removed, up-sampling can be lossless but only if done by discrete values[16, pg. 750–751]. If up-sampling is done by non-discrete values infomation is lost.

6.2 Variable sample rate

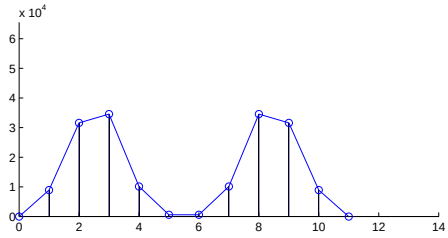
Using variable sample rates, in essence: playing the samples at a chosen speed using the sound-cards instead of resampling the data would be the ultimate solution given limited sample rates and resolutions. Just playing at different speeds would mean no manipulation of the audio signal and there by not adding additional errors the data. Because the data was made to be listened to at normal speed noise not normally audible can be heard when playing at other speeds but these were already in the original data. As mentioned earlier in the chapter most hardware unfortunately can't play with variable sample rate, because of this the method was abandoned.

6.3 Interpolation

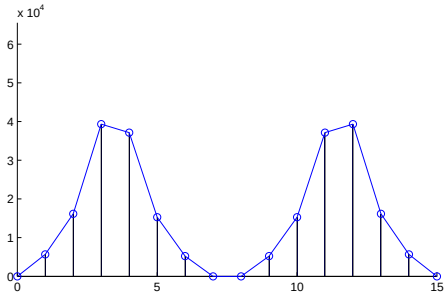
The straightforward interpolation simply works by connecting the old samples into a curve, then putting new samples on the curve at the desired density. This method is very simple to implement. This creates audible noise in the output and for this reason it can't be used. see figures 6.3. These results were discovered by experimentation and later confirmed in literature[16, pg. 750–755].



(a) original audio 14 samples



(b) interpolated down-sampled to 12 samples

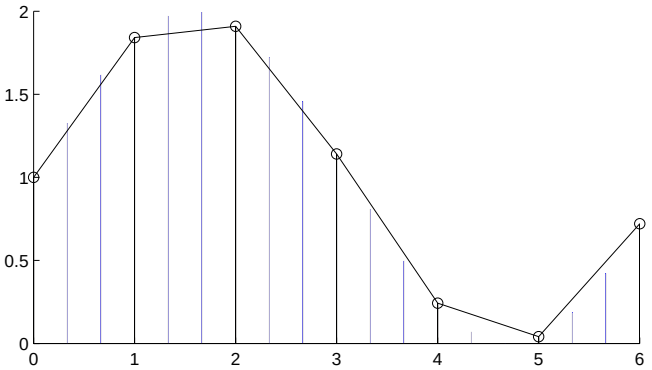


(c) interpolated up-sampled to 16 samples

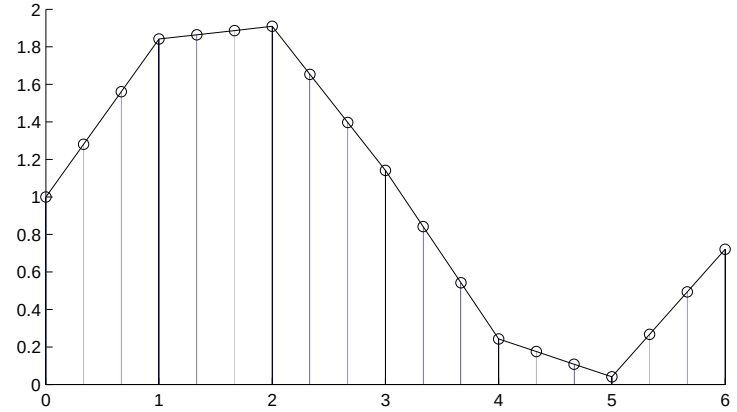
Figure 6.1: Resampling using interpolation

6.4 Discrete resampling

Discrete resampling means up or down-sampling by factors of 2,3,4 etc. Discrete down-sampling is achieved by simply dropping all samples except the one equal to the factor[14, pg. 386–389], for example: down-sampling by factor 3 mean dropping all the samples except for every third one(see figure 6.2(a)). Discrete up-sampling is done by interpolating and adding factor-1 samples between each of the original samples[14, pg. 389], for example: up-sampling by a factor 3 adds 2 interpolated samples between all original samples(see figure 6.2(b)). The methods for up and down-sampling described in this section gives the same result as using interpolation with discrete factors. The methods were found by experimenting with interpolation and calculating on paper. The methods were later confirmed in literature[14, pg. 386–389] [16, pg. 750–762]. Resampling using discrete factors resulted in audio with acceptable quality. Discrete up-sampling takes much more computer power then discrete down-sampling because all added samples must be calculated, when down-sampling the samples are simply dropped.



(a) Remaining samples in black and dropped samples in grey. Down-sampling by factor 3

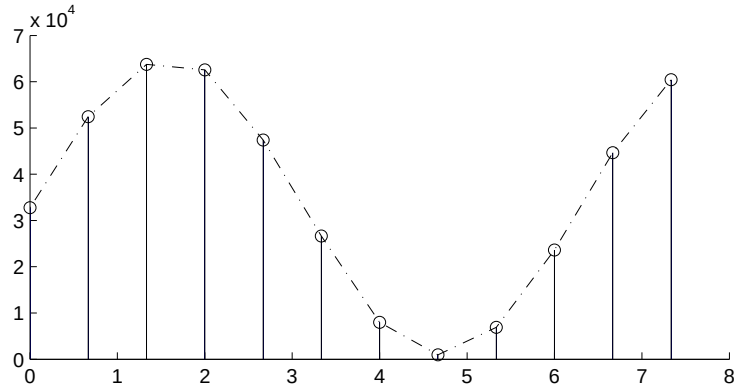


(b) Original samples in black and added samples in grey. Up-sampling by factor 3

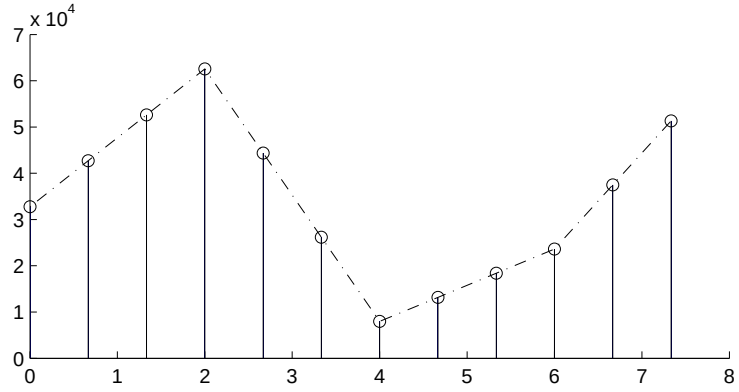
Figure 6.2: Discrete up-sampling and down-sampling

6.5 Discrete up-sampling and discrete down-sampling

To achieve close to any sample rate a method of first up-sampling then down-sampling could give a quotient, for example: up-sampling 3 and down-sampling 2 would give a final up-sampling by a factor of $3/2 = 1.5$. This could in theory give you any rational number. This method was created with calculations and later confirmed in literature[16, pg. 762–766]. Unfortunately very large numerators require a lot of calculations, using very large numerators would slow the system down to much. It is important that the up-sampling is done first as the up-sampling does not remove any information, and adds more interpolated data based on the original data. When later down-sampling, even if samples that was part of the original data are dropped, data based on the removed samples are still left and the data is not completely lost. If the down-sampling is done first data is lost and the up-sampling process can not recreate any data that has already been dropped, this adds noise(see figures 6.3(a) and 6.3(b)). Depending on the proximity to the goal number required the up-sampling factor can get very large, resulting in slow operation. An example of this would be up-sampling by a factor of $1.001 = 1001/1000$, in this case no resampling at all might be preferable to using a quotient with such a large numerator.



(a) 16 samples of a sine curve resampled to 3/4 using up-sampling first



(b) 16 samples of a sine curve resampled to 3/4 using down-sampling first

Figure 6.3: Resampling up-sampling or down-sampling first

6.6 Down-sampling by fractions

To add more resolution in the method described in section 6.5 a method of down-sampling by fractions was tried. This mean instead of only down-sampling by discrete numbers, it could be done by quotients. For example: if the wanted sample rate was 0.75 of the original an up-sampling of 3 times and a down-sampling of 4 would be needed to get $3/4 = 0.75$. If you could simply drop every fourth sample (down-sampling by 3/4), not a single up-sampling would be needed. After testing it was found that using this method often creates audible noise and was therefore not used. Some fractions could probably be used without adding too much noise but this was not pursued.

6.7 Final method

The only useful method found was the one described in section 6.5. All other tried methods added to much noise. The algorithm is basically the one described in section 6.5 with a few modifications. To increase performance, up-sampling and down-sampling is actually done simultaneously. This way a sample added from the up-sampling samples that would be dropped in the down-sampling can simply be skipped and not calculated. Because the algorithm has to resample to different speeds many times a second, the audio is processed in small chunks. To handle the sample in the beginning of the chunk of data the last sample from the previous chunk is saved, this allows the algorithm to calculate the chunks as if they were a continuous stream of samples. The used method to get the needed quotient is described in section 6.8. A very simple pseudo code description of resampling of a mono signal follows.

```
vector Input[];  % Vector to hold a chunk of the original audio
vector Output[]; % Vector to hold the resampled data
integer lastSample; % holds the last sample calculated in previous run.

begin
get qoutient for current resampling (A/B);
fill Input[];

for {x = 1 to A} do % calculate samples to be added before Input[0]
  if x != B
    continue; % the added sample would be dropped in downsampling
  else
    calculate added sample from lastSample and Input[0], write to Output[];
  end if;
end;

for {x = 1 to length of Input} do % calculate samples from Input[0] to Input[end]
  if x != B
    continue; % the added sample would be dropped in downsampling
     else
    calculate added sample from input[x-1} and Input[x], write to Output[];
  end if;
end;

lastSample = Output[last];
return Output[]
```

6.8 Conversion to rational number

To use the above mentioned method to resample the audio, a rational number with an as small as possible numerator is needed. This is next to impossible to achieve by an algorithm in close to real time. The solution used was a sorted table of rational numbers and the corresponding quotient. The when a quotient is needed the table is searched, because the table is sorted this search is relatively quick[15]. The current list used contains over 30.000 unique numbers and gives good enough resolution for this application. A larger list could be used but this list was deemed to have high enough resolution.

Chapter 7

Accomplishment

This chapter deals with the actual work and how it was accomplished. The initial sections goes through the different stages of design and construction of the prototype. The later sections deals with the different foreseen an unforeseen issues encountered during the project.

7.1 Initial idea and plans

The plans for the project was discussed for a long time with my friend and fellow computer science student Andreas Johansson (See chapter 10). We were both interested in a cheaper and more flexible DJ system to be used at home. We had considered doing a similar project on the side, but it all stayed at the idea and discussion stage.

7.2 Initial design and research

Initially the software was designed on paper, modules where laid out and the major data and control-flows where written down. Because an object oriented programming language was used a modular design came naturally. Then followed some research, primarily on the Internet for libraries that would make the layout possible. In the initial design a lot of assumptions about library and hardware functions where done, later to be researched in depth or tried in experiments.

7.3 Writing process

After the initial design and research, the process of writing the system began. The system was built using modules that later were assembled to a working system. The most critical modules were written first, to get important issues resolved early. If a module could not be made as expected it would require less rewriting later. After the most critical modules were done, the modules that would allow the critical parts to be sub assembled and tested were made. GUI and other less critical modules where made last.

7.4 Building hardware

The only hardware built were the two virtual turntables. These where simply constructed by using optical mice suspended over the spinning disks of two turntables with simple wooden structures. On the spinning disks a mouse-pad was added to allow more accurate readings for

the optical mice. To get accurate enough readings of the optical mice both more high quality mice and a better reading surfaces than initially thought was needed.

7.5 Challenging problems

Several parts of the system were much harder to get working to satisfaction than anticipated. Problems with bad documentation, complex libraries and unexpected functionality were encountered in several parts. Also problems caused by poor research on issues caused problems. Below follows the most challenging problems encountered in the course of developing the system.

7.5.1 Audio playback

The audio playback using ALSA[1] constituted a challenge because of the complex low level library used. Although it was reasonably well documented, using it was fairly complex and required a lot of testing and careful reading of the documentation. Especially since the soundcards had to be accessed and setup for low latency use.

7.5.2 Audio decompression

The Audio decompression used a library called MAD[6] to decompress MP3 files. This library was both very complex and had very poor documentation. Making the decompression module work required extensive testing, reading through source code of other programs and reading through the complex header files of the library. The final result was a usable module but several functions of the system had to be skipped to accommodate for the functions that could not be implemented. The issue that caused several functions to be dropped was the complexities of decompressing the audio starting from the middle of a track and decoding the audio in reverse.

7.5.3 Audio resampling

The audio resampling module was written from scratch and therefor had no problems with complex libraries. The major issues with writing this module consisted of poor research of what algorithms would work, several algorithms were implemented and found not to work properly. With just a small amount of research the proper algorithm would have been used from the start. See chapter 6 for more information on audio resampling.

7.5.4 Virtual turntables

The virtual turntable had two major issues, hardware and the interface of the mice to the computer. On the hardware side the issue was an underestimation of what quality of mice and the reading surface was needed to get accurate reading. After for a long time assuming the problem was the software interface, purchasing better quality laser mice and using a mouse pad for reading surface greatly reduced this problem. Another issue was that the rotating disks where not completely level and created audible pitching. The problem with the interface to the mice was also an incorrect assumption. The input included time-stamps, these were assumed to be accurate which was not the case.

Chapter 8

Results

This chapter will give a description of the final system, both the hardware and the software. The chapter will start with a description of the hardware used, then a description of the software system. After the software section there is an analysis of how well the systems reached the goals of usability, performance and functionality(see chapter 4).

8.1 Hardware

The hardware of the system consists of a computer equipped with two sound-cards, a keyboard and a monitor. The system has two virtual turntables, a mixer and a speaker system. A list of the hardware used can be found in appendix A.

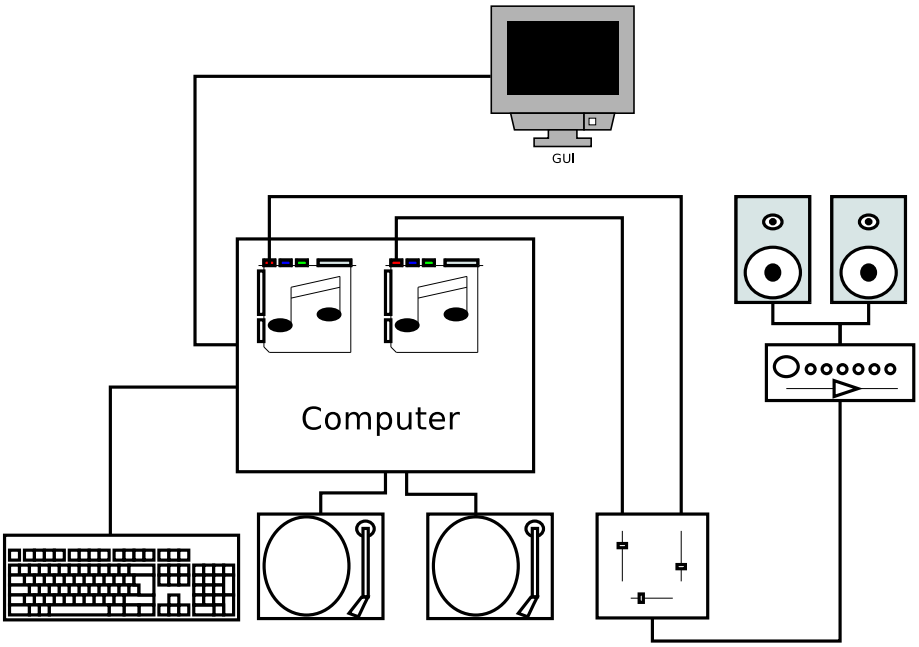


Figure 8.1: Hardware used

8.1.1 Virtual turntables

The virtual turntables are designed to work as a regular turntable with the exception that instead of producing sound the only deliver data about how fast and in which direction the disk is rotating. The data is then used by the computer to manipulate the sound playback appropriately. The prototype virtual turntables are made simply by suspending an optical mouse over the surface of the rotating disk of a regular turntable.

8.1.2 Computer

The computer and its software is basically the core of the system. It needs two sound-cards to deliver two full stereo streams of audio, the software is run on the Linux operating system. The software supplies the GUI, decompresses the MP3 files and does the audio manipulation dictated by the virtual turntables. When all the audio manipulation is done the audio is then delivered to the mixer. The whole system is controlled by the two virtual turntables and a keyboard.

8.1.3 Mixer and amplifier

The mixer and amplifier are simply off the shelf items. The mixer is a two channel mixer as used in a regular DJ system. The amplifier could be any stereo or PA system.

8.2 Software

The software is a central part of the system and almost all the work was put into that. This section will try to describe how the software system is constructed. Primarily how the audio and control is handled. Following is an UML class diagram, after that overview of how the system is used and controlled. Finally a description of how audio is handled.

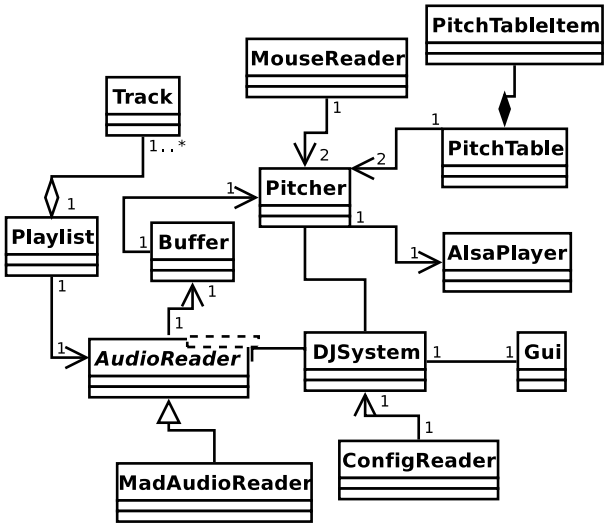


Figure 8.2: Class diagram

8.2.1 System overview

The system is controlled by the the keyboard controls and the virtual turntables. The keyboards controls lets you start and stop each respective sound-card and controls which track from the playlist that will be played. The playlists contains paths to the MP3 files used by the audio pipeline. Once playback is started by the keyboard controls the system is driven by the sound-cards(see section 8.2.2). The audio data goes through the audio pipeline and is decompressed and pitched and finally played back by the sound-cards. Below is a simplified view of the system.

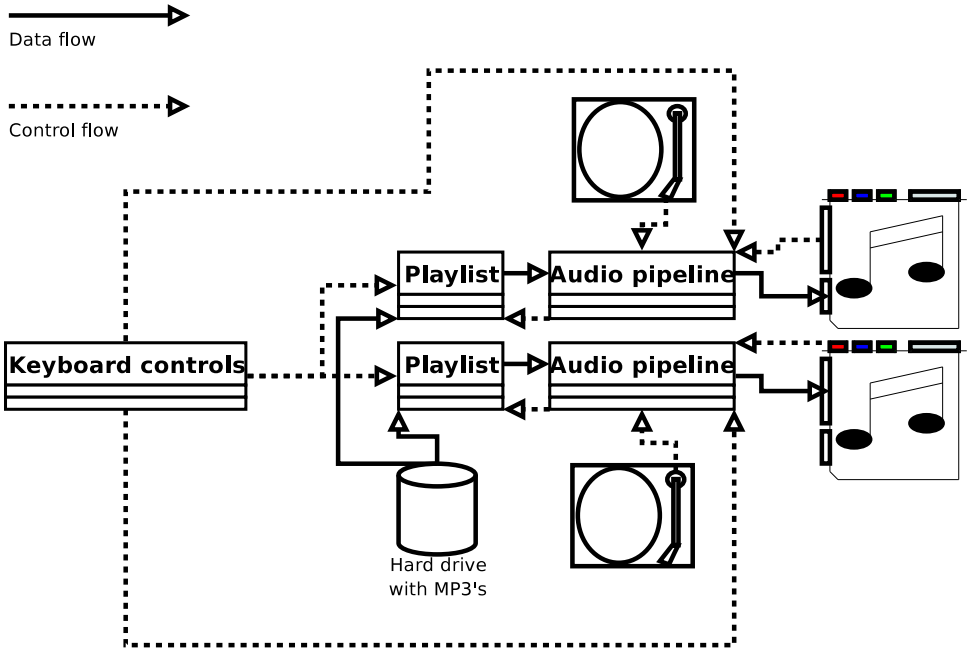


Figure 8.3: System overview

8.2.2 Audio pipeline

This view looks at the audio pipeline, it describes the path from compressed audio to decompressed and pitched audio fed to the sound-card for a single player. The compressed audio is first decompressed and put in a large audio buffer, this is done by decompressing an initial chunk before playback is started, the rest is decompressed in smaller pieces during playback. When playback is started the sound-card continuously requests audio data to output to the speaker. This data is obtained from the pitcher module by way of the sound-card control module. When the pitcher module gets a request for new data it in turn gets the current pitch from the turntable and data from the audio buffer. After obtaining pitch and audio data the pitcher module then resamples the data and sends it to the sound-card.

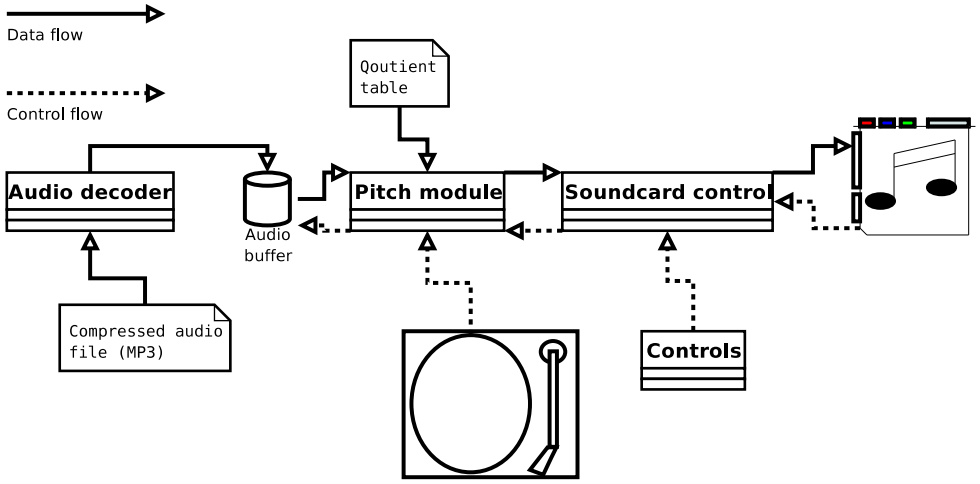


Figure 8.4: Audio pipeline overview

8.3 GUI

The GUI is very simplistic, it contains a twin display of the two players side by side. They display the current tracks, the playlists and elapsed time for each player. It also has a button display that shows if each player is playing, stopped or paused.

8.4 Controls

The controls of the system consist of the virtual turntables, a mixer and a keyboard. The virtual turntable allows you to control the speed of the playback by controlling the speed of the disk. The mixer lets you control the volume from each player with three sliders, one for each player and a cross-fade slider(see chapter 2). The keyboard has buttons to play, pause and stop the two players as well as buttons to select another track from the players respective playlists. For a complete description of the controls see appendix B.

8.5 Functionality

As the criteria for functionality consisted of creating a proof of concept that could play two tracks simultaneously using the virtual turntables, this goal was fully met.

8.6 Usability

One of the major goals of the project was to create a usable system, this section will deal with how well the system met this goal. The usability will be described as a comparison with standard DJ-equipment, then a short description of the major shortcomings and advantages with the system associated with usability. Lastly an analysis of the usefulness of the system. For a more detailed description of the shortcomings see chapter 9.

8.6.1 Comparison with a regular DJ system

The major difference between this system and a regular DJ system stems from the differences of using digital MP3's compared to using vinyl records. The system uses playlists instead of a set of records, this has both advantages and disadvantages. With a vinyl record the needle can be placed anywhere on the track, with this system playback is always started in the beginning of the track and the only way to find a point later in the track is to spin the disk. An advantage with playlists is that finding tracks is simpler than searching through a box of albums. With regards to use and control, the system is close to the usability of a regular system. The system is also similar enough in use to facilitate a migration from use of vinyl.

8.6.2 Performance

The interesting performance issues of the system with relevance to usability are latency and the accuracy of the virtual turntables. Latency refers to the time it takes from the DJ moving the virtual turntable to the resulting audio is heard from the speakers. The latency of the system is deemed to be adequate even on an older computer (See appendix A for the hardware used). Accuracy refers to how well the audio output follows the actual disk-speed on the virtual turntables. The accuracy is not good, the playback suffers from audible pitch changes. This is probably caused by the sensors used and the fact that the disks are not absolutely flat. This results in such poor audio output that the system is not useful. See section 8.7.

8.6.3 Playlists

The playlist can only be loaded on startup and can not be edited in any way during execution of the system. Having to restart the system to edit the playlist reduces its usefulness. There is also no search function in the playlists making it hard to find tracks in large playlists. But playlists are still significantly more manageable than a large box of vinyl records.

8.6.4 Virtual turntables

The current virtual turntables are designed by simply suspending a optical mouse over a record player. The stands are very crude, if they are accidently moved the system has to be restarted to function properly.

8.6.5 Usability analysis

With the exception of the performance and the missing ability to edit playlists during execution the system is reasonably useful. Unfortunately the performance part of the usability makes the system pretty much not useful for anything but playing music to people with less discerning demands on sound quality.

8.7 Performance

The audio part of the system is designed to run as close to real time as possible. This means the the time between control input and the resulting audio response should be as small as possible. Because of this performance of the system is critical. The human ear is also very sensitive to errors in audio, so the system must also be able to read the controls accurately. The system was designed with performance as a primary goal. Below follows descriptions of what steps were taken to ensure good performance, an evaluation of the actual performance and an analysis of remaining performance problems.

8.7.1 Operating environment and programming language

The operating environment chosen was to run under Linux and if possible not using a graphical window system, this had several reasons. In regard to performance the reasons where primarily the possibility to run without a performance demanding graphical window system and the simpler low level access to hardware like sound-cards and mice. The language used was C++, this had two performance benefits. Primarily C++ produces relatively fast programs compared to some other languages and secondly using C++ allows usage of many low level libraries needed.

8.7.2 Sound-cards

The sound-cards are very critical to the performance of the system, the size of the buffers in the sound-cards directly affect the latency of the system. To optimize the buffer size of the sound-cards low level access to the hardware was necessary. This was accomplished using the ALSA library. The buffer sizes must be big enough so that the sound-card never runs out of data to playback but also as small as possible to decrease latency.

8.7.3 GUI

The GUI is designed not to need a graphical window system. Window systems generally consume a lot of memory and CPU, not using one allows the DJ system to be able to use more. Also the GUI is very simple and does not demand a lot of CPU, but this probably has very little impact on performance.

8.7.4 Mice and mouse input

In this prototype the virtual turntables were made by suspending an optical mouse over a regular turntable. The performance of the mice had a significant impact on the accuracy of the system, less accurate mice gave a very poor result. The surface on the disk of the turntable was also critical, when not using a surface design for optical mice the result was unusable input. The

fact that the disks are not exactly level also added to the accuracy problems. Unfortunately high performance mice and better reading surface did not produce accurate enough input with audible inaccuracies as a result.

8.7.5 Audio decompression

The audio decompression demands a lot of computer power, because of this the decompression had to be done at appropriate times during the execution. One option would be to decompress the entire track before playing, but this caused an irritating pause before the track started. Instead the beginning part of the track is decompressed before the track is played, the rest is decompressed during the playback of the song. This solution can potentially create a buffer under-run, but only under extreme circumstances.

8.7.6 Audio resampling

The audio resampling algorithm was written from scratch to get a fast and accurate execution (see Chapter 6).

8.7.7 Performance analysis

Many steps were taken to get low latency and this was achieved, unfortunately the sound-card's buffers created a bottleneck. The buffers on the sound-cards is a latency which is unavoidable, however fast the system performs it still has to wait for the sound-card to process its buffer. As far as latency is concerned the performance is good despite the latency from the sound-cards but when it comes to accuracy the system is badly lacking. The resulting audio has audible fluctuations in pitch, these are very apparent during sustained notes. This lack of accuracy unfortunately makes the performance poor, fast response is not enough when the quality is lacking. The system was tested on the hardware described in appendix A.

Chapter 9

Conclusions

This chapter goes through the major limitations of the system and possible future solutions of these issues. After that a description of interesting future additions to evolve the system further. Finally, conclusions of what was learned during the project.

9.1 Limitations

The biggest limitations of the system are associated with the performance and the usability of the system. The primary issues to be handled to increase the future usability of the system are to improve the virtual turntables and add functionality to the playlists. Availability and audio format support are less critical.

9.1.1 Virtual turntables

The virtual turntables lacks the accuracy needed to deliver a smooth audio output. This problem is critical, audible effects due to the hardware makes the system unusable when good quality audio is required. To address the accuracy issues the turntables should be purpose built with suitable sensors. If the the method of using optical mice is kept the reading surface must be made to be absolutely flat to remove the wobble of the spinning disk, and the mounting of the sensors should be better built. The sensors should also preferably be mounted below the spinning disk allowing the DJ to use the entire disk without having to worry about bumping into the sensor.

9.1.2 Playlist functions

The system can only load playlists on startup. Functions to add or remove tracks from the playlist while the system is running would greatly improve usability of the system. A search function for finding tracks in a large playlist and functions to find a point in the middle of a track would also be desirable.

9.1.3 Availability and audio formats

The system is written to run in a Linux environment, unfortunately this makes the system unavailable to the large number of people using Windows. The system was initially designed to be run on a bootable CD with a Linux operating system, this was however never realized. Creating a Windows version could also be considered but that would demand a great deal more work. A bootable CD would allow the system to be run on any platform supporting Linux. The system was designed to be able to support other audio formats but it was never implemented, because of the popularity of MP3 this was the only format supported. Adding support for more audio formats should be relatively straight forward assuming the libraries have the needed functionality. Formats like Ogg, flac, WMA and primarily CD's would be desirable, unfortunately many audio decoding libraries including MP3 and WMA are not free.

9.2 Future additions

Of future additions considered the most interesting would be to add functionality for audio effects. Adding a "scratching" effect would make the system act much more like its vinyl counterpart and make it more interesting to use for a DJ. Additions to the GUI would also make the system more useful and make it a more interesting alternative to a regular DJ system.

9.2.1 Mixing in the computer

If the mixing could be done in the computer the audio output could be recorded digitally. This would allow the DJ to record his/her session in as good quality as possible. This would however add another step in the audio manipulation and might add latency.

9.2.2 Audio effects

The primary effect that would be of interest would be to add "scratching" sounds when the virtual turntable are moved fast. This would emulate what happens when a regular vinyl record is jerked fast causing the needle to produce a high pitched noise(see chapter 2). Other effects like echos and allowing playback at different speeds with normal pitch(called master tempo) would also make the system more attractive to a DJ.

9.2.3 GUI

Several additions to the GUI are considered. Displays for beats per minute(BPM), the current speed of the virtual turntables and time remaining of the current track would be nice additions to the GUI. Also a spectrum analyzer would be a good addition. Finally a nicer looking skin would also be nice, this could be easily implemented by editing the image file of the standard green GUI.

9.3 Conclusions

The biggest lesson learned during the completion of the project was to do more research before starting to write code. This primarily affected the audio resample module which would have taken much less time and effort if the proper research had been done before hand. Also the scale of the project was not appreciated properly, it turned out to be much more work than predicted.

Chapter 10

Acknowledgements

Contributing to this project was my supervisor Jan-Erik Moström and my friend and colleague Andreas Johnsson. I would like to thank Andreas for inspiration, ideas and technical solutions used in this project. Also I would like to thank Roger Jacobsson for help with the report. Finally I would like to thank Maria Stenlund for helping me with the report and the presentation.

References

[1] Alsa homepage. Audio playback library http://www.alsa-project.org/ Visited 04-04-09.

[2] Debian linux distrobution. http://www.debian.org.

[3] Dj portal. DJ resource page(swedish) http://www.djportal.nu/ Visited 04-04-09.

[4] Example of similar system using cds. Pioneer CDJ-1000 MK3 http://www.pioneer.eu/eur/products/44/106/462/CDJ-1000%20MK3/index.html Visited 04-04-09.

[5] Example of similar system using mp3 on computer. Vinyl Revolution Pack 20 http://www.djtechpro.com/product_detail.asp?category_id=45&sub_id=98&product_id=196 Visited 04-04-09.

[6] Mad homepage. Audio decompression library http://www.underbit.com/products/mad/ Visited 04-04-09.

[7] Mpeg homepage. MPEG layer 3(MP3) http://www.chiariglione.org/mpeg/ Visited 04-04-09.

[8] Resource on noise shaping(wikipedia). http://en.wikipedia.org/wiki/Noise_shaping Visited 04-04-09.

[9] Sdl homepage. Graphics library http://www.libsdl.org Visited 04-04-09.

[10] X window system. http://www.x.org/ Visited 04-04-09.

[11] Theunissen Baert and Vergult. *Digital audio and compact disc technology*. Second edition, 1992. pages 47–48.

[12] Bruce Bartlett and Jenny Bartlett. *Practical recording techniques*. Third edition, 2002.

[13] Russ Haines. *Digitalt ljud*. 2002.

[14] Simon Haykin and Barry Van Veen. *Signals and Systems*. Second edition, 2003.

[15] Lars-Erick Janlert and Thorbjörn Wiberg. *Datatyper och algoritmer*. 2000. pages 307–308.

[16] John G. Prokakis and Dimitris G. Manolaktis. *Digital signal processing*. Fourth edition, 2007.

[17] Lennart Zetterberg. *Ljudinspelningens A B C*. 2002.

Appendix A

Hardware Used

A.1 Computer

The computer used is a PC running Linux with the following specifications:

- Athlon 1 GHz processor
- 512 MB 100 MHz physical memory
- custom Linux 2.6.18.6 kernel
- Debian stable Linux distribution

A.2 Sound-cards and mice

Here follows a list of the sound-cards and mice used.

- sound-card 1: C-Media 8738-MC6 PCI
- sound-card 2: Trident 4DWave NX PCI
- mice: A4Tech X-750F (USB)

Appendix B

User's Guide

The user's guide will be split into two parts; setup pre-startup and a guide for using the system when it is running.

B.1 Setup

The setup of the system consists of setting a number of values in the file "config.cfg". A short description of the differnt values to set are listed below, make sure you have a computer running ALSA and has the "evdev" module installed. If you are planing to use the system without the X window system, make sure you have a working svga console.

- audioDevices:
  This tells the system what audio devices should be used in the left respectively right player. If you do not have your own setup in the ".asoundrc" they should probably be set to "hw:0,0" and "hw:1,0".
- mouseDevices:
  This tells the system what mouse devices to use to read from the left and right turntable. This should set to the corresponding event device in the "/dev/input/" directory. Depending on what Linux kernel is being run the devices may be housed in a different directory. If the devices can't be found make sure the "evdev" module is installed or compiled into the kernel.
- playlists:
  This tells the system what playlists should be used, both .m3u and .pls can be used but the .m3u file structure is preffered.
- hardware:
  Here the hardware buffer sizes are set, these should be set as low as possible, but setting them to low will cause the system to fail. The "bufferSize" variables sets the buffer size for the sound-cards, the "periodSize" variables sets the amount of audio data that should be sent out at a time. The "callbackTrigger" variables sets when the buffer is too low and needs to request new data.

- decoding:
  Here you set the amount of frames to decode in each system cycle. If the system runs to slow, try to lower this value. If you suffer from buffer underruns try and increase this value and the "initialSeed" value.
- initialSeed:
  The initialSeed variables sets how many frames of the track should be buffered before playback is started. If the tracks pause to long before playback starts try lowering the initial seed. If you experience underruns try raising the initial seed.
- pitchTable:
  The pitch table is a table used by the system. If the system is running to slow try using a smaller pitch table, if you are experiencing bad sound try using a larger table. If you like to share a table between the two players(recomended), set the right pitchtable to "none".
- mouseReads:
  These values sets how many reads each mouse device should use. If the system seems to get bad reads from the turntables, try increasing the reads. If the system is slow, try decreasing. In some cases the first read of the mice can be incorrect, if you are experiencing problems with bad reads then set the "skipFirst" values to true.
- buffers:
  These variables sets the internal buffers of the system, if the system is using to much memory you can try and lower these values. If you lower the "pitcher" values to much you can have problems with playback, especially if you use a pitcher table with large values. The "buffer" values should be big enough to accomodate the longest tracks you intend to play uncompressed, uncompressed audio uses approximatly 10 Mb/minute. The values are in bytes.

B.2 Use

This section will describe the mixer and turntables that controls the audio playback, the descriptions are limited to what the controls do and not the different technics used. After that the keyboard contols are described.

B.2.1 Turntables

The turntables work just like regular turntables, increase the speed of the disc and the audio playback speeds up and vice versa.

B.2.2 Mixer

The mixer controls the volume level of each player with two sliders. A cross-fade slider that allows you to relative levels of the two players is also avaliable, sliding this all the way to the left will make the right player quiet and vice versa.

B.2.3 Keyboard controls

The keyboard controls allows you to scroll and change tracks in the playlist, it also lets you play, pause and stop the two players. The keys are listed below.

Function	Left player	Right player
Play	z	b
Pause	x	n
Stop	c	m
Scroll up	q	t
Scroll down	a	g
Move selection up	w	y
Move selection up	s	h
Jump to selected	d	j
Jump forward	r	i
Jump reverse	e	u
Quit	Esc	Esc