Forside

## Sammendrag

## Innledning

### Bakgrunn

### Problemstilling

Installasjonsinstruksjoner?

### Innholdsfortegnelse

## Digital signalbehandling og musikkteori

### DFT - Discrete Fourier Transform og FFT – Fast Fourier Transform

Frekvensanalysen av lyden som spilles inn blir utført ved hjelp av fouriertransformasjon. Fouriertransformasjon brukes for å gjøre signaler fra tidsdomenet om til frekvensdomenet, og er definert for både kontinuerlige og diskrete signaler. Lyden som tas opp fra mikrofonen er i prinsipp kontinuerlig så lenge mikrofonen er på, men det må plukkes ut en endelig mengde verdier til analyse. DFT, diskret fouriertransformasjon, er metoden som benyttes når det finnes en endelig mengde datapunkt.

DFT er definert ved

FFT, eller Fast Fourier Transform, er en algoritme for å utføre diskrete fouriertransformasjoner mer effektivt. Den gjør akkurat det samme som DFT, men det går mye raskere. Hele operasjonen deles opp i flere små DFT'er. Litt interessant er det at en slik algoritme ble oppdaget allerede i 1805 av Carl Friedrich Gauss, men så lenge det ikke fantes datamaskiner så var det ikke interessant og den ble glemt. I 1965 ble den plukket opp igjen av J.W. Cooley og J. Turkey og igjen lansert i deres artikkel «An algorithm for the machine calculation of complex fourier series».

En vanlig DFT(diskrete fourier transform) tar O(N2) operasjoner, mens en FFT gjør den samme jobben med O(N*log*N). Dette betyr at med store datasett er tidsbesparelsen enorm.

.....mer om O(n)

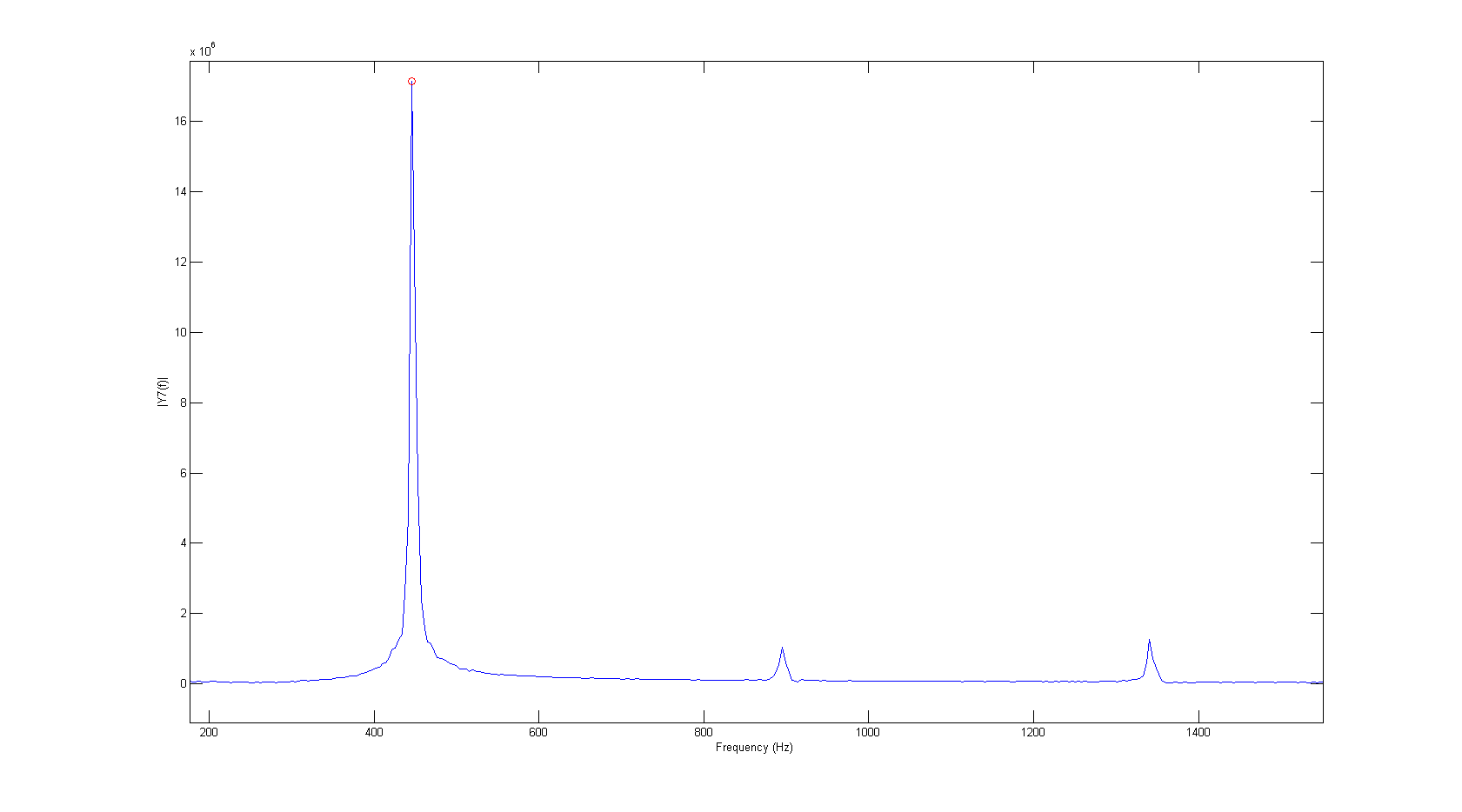
For å fullt utnytte denne algoritmen må antall punkter N være et av tallene 2n hvor n er et heltall. I frekvensanalysen appen gjør jobber den med 4096 (212) datapunkter og kan derfor fullt utnytte symmetrien i en sinus- og cosinusfunksjon.

Dataene som appen henter fra mikrofonen deles opp i «chunks», altså en endelig mendge datapunkter, og det utføres en FFT på hver chunk for å finne frekvensen.



Figur 2.1 – Typisk lydopptak, 4096 samples over 256ms.

DFT tar dataene fra tidsdomenet og gir oss en representasjon av de samme dataene i frekvensdomenet.

Figur 2.2 viser funksjonen fra figur 2.1 etter FFT, og man kan se en representasjon av amplituden til de ulike frekvensene fra det originale signalet.

Figur 2.2 – Lydopptak etter FFT

På denne figuren ser det ut som det finnes flere markerte frekvenser. Dette er et resultat av at et lydopptak har en fundamental frekvens, og flere overharmoniske frekvenser. Hvis vi har en lyd som er 440hz, har den overharmoniske frekvenser på 880hz og 1320hz.

Etter transformasjonen til frekvensdomenet er det mulig å bruke en invers transformasjon for å komme tilbake til tidsdomenet.

Består av komplekse tall; en imaginær del (sinus-bidrag til frekvenser) og en reell del (cosinus-bidrag til ulike frekvenser).

Problemer med fft, oppløsning i tidsdometet versus frekvensdomenet

### Frekvenser i Musikk

Det mennesker oppfatter som lyd, er trykkbølger i luften som oppstår fra vibrasjoner av for eksempel stemmebånd eller en gitarstreng og kalles gjerne lydbølger.

Lyden kan representeres som en sinusfunksjon. I figur 2.1 kan det være vanskelig å se at det dreier seg om bølger, men hvis vi zoomer inn kan man tydelig se bølgene.



Figur 2.3 – Zoomet inn område av figur 2.1

Bølgelengde λ er avstanden mellom to bølgetopper. Frekvens *f* er hvor mange ganger en hendelse gjentar seg i løpet av et bestemt tidsrom, i tilfellet med lyd altså hvor mange bølgetopper som passerer per tidsenhet. Bølgelengden har et inverst forhold til frekvensen, definert ved

hvor c er hastigheten til lyd i luft, 343m/s ved 20 °C,

*f* er frekvensen i hz (s-1)

λ er bølgelengden i meter

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **3** | **4** | **5** | **6** |
| **C** | 131 | 262 | 523 | 1047 |
| **C#** | 139 | 277 | 554 | 1109 |
| **D** | 147 | 294 | 587 | 1175 |
| **E♭** | 156 | 311 | 622 | 1245 |
| **E** | 165 | 330 | 659 | 1319 |
| **F** | 175 | 349 | 698 | 1397 |
| **F#** | 185 | 370 | 740 | 1480 |
| **G** | 196 | 392 | 784 | 1568 |
| **G#** | 208 | 415 | 831 | 1661 |
| **A** | 220 | 440 | 880 | 1760 |
| **B♭** | 233 | 466 | 932 | 1864 |
| **H** | 247 | 493 | 988 | 1976 |

### Noter

Ulike lenger på noter

Pauser

Sharp/flat

treble/bass

## Programmeringsverktøy

### Android

VeryNote er laget for bruk på mobiltelefoner som kjører Android operativsystem. Den siste versjonen er i skrivende stund Lollipop 5.1 (SDK 22), man må minst ha versjon 4.0 IceCreamSandwich (SDK 14) for å bruke VeryNote.

Hvis man velger å kun utvikle for den siste versjonen så risikerer man at veldig få vil kunne bruke appen, mens hvis man velger en veldig gammel versjon mister man muligheten til å bruke metoder og funksjoner som finnes i nyere API. Rundt 90% av alle enheter som kjører Android har idag versjon 4.0 eller nyere [1], vi valgte derfor å ha dette som minimumsversjon for at appen skulle kunne brukes på flest mulig enheter, samtidig som vi har et nyere API å jobbe med.

### Android Studio

Android Studio er den offisielle IDE for å skrive applikasjoner for Android ved bruk av programmeringsspråket Java. Det fungerer på mange måter som Eclipse for Java, men inneholder i tillegg flere funksjoner som er nyttige i utviklingen av en app. Det finnes en emulator man kan teste apper på uten å måtte koble til en mobiltelefon, og det er enkelt å sette opp en fungerende hello-world app for videre utvikling. Når man jobber med brukergrensesnitt kan man plassere knapper og tekstbokser rett på en forhåndsvisning, så ser man med en gang hvordan det vil se ut.

Det finnes også et verktøy som heter Memory Monitor, hvor man kan se i sanntid hvor mye minne appen bruker.

## Implementasjon av teori

### JTransform/PitchDec.java

JTransform er et Java bibliotek med åpen kildekode. Det inneholder fire forskjellige transformasjoner, Discrete Fourier Transform (DFT), Discrete Cosine Transform (DCT), Discrete Sine Transform (DST) og Discrete Hartley Transform (DHT). Vi har benyttet oss av klassen DoubleFFT\_1D og metoden complexForward(double[] a).

Når brukeren trykker på Record-knappen begynner mikrofonen og ta opp lyd i 16bit PCM format. PCM, eller Pulse-code modulation, er en digital representasjon av et analogt signal. For å unngå at bakgrunnsstøy blir analysert må amplituden på lyden være over 4400. Når lyden blir gjort om til 16bit PCM, representerer tallene amplituden på lyden, og vi fant ved hjelp av testing at 4400 var et tall som eliminerte bakgrunnstøy uten at en må rope unødvendig høyt inn i mikrofonen for at resultatene skal bli analysert og vises på skjermen.

Opptaket fra mikrofonen blir lagret i en array, som det så blir gjort en FFT på. Vi leter etter den fundamentale frekvensen mellom 131hz og 1976hz. Dette dekker 4 oktaver fra C til H. VeryNote er i hovedsak laget for å analysere opptak av sang, derfor er frekvenser utenfor dette spekteret mindre interessante og ikke inlkudert.

Deretter brukes resultatene fra FFT for å finne frekvensen med høyest amplitude. Dette skjer i filen PitchDec. Som sagt tidligere, selv om lyden i prinsipp er et kontinuerlig signal, må den deles opp i biter som kan analyseres. Opptaket blir gjort med en «chunksize» på 4096, og raten er 16000. Det betyr at hver «chunk» går over (1000\*4096)/16000 = 256ms. Hver «chunk» blir så analysert vha FFT. Se figur 2.2. Hvis vi zoomer inn ser man tydelig at den fundamentale frekvensen i dette tilfellet er mellom 440 og 450hz.



Figur 2.4 Zoomet inn område av figur 2.2

Frekvensen med høyest amplitude blir lagt til en «cluster». Siden vinduet analysene blir utført på er så små, vil det normalt være flere vindu med samme frekvensresultat etter hverandre. Metoden *isNear* blir kalt for å sjekke om frekvensen er mindre enn 5% forskjellig fra forrige frekvens. Hvis den er mindre enn 5% forskjellig blir den tolket som en fortsettelse av den samme noten og lagt til samme «cluster». Hvis den er mer enn 5% forskjellig blir den tolket som en ny note og en ny «cluster» blir opprettet.

Et normalt FFT resultat vil vise fundamental frekvens, i tillegg til frekvensens overharmoniske frekvenser. Se figur 2.2

For å unngå at overharmoniske frekvenser blir oppfattet som egne toner, slåes «clusterene» som inneholder frekvenser som er overharmoniske sammen vha metoden *isHarmonic.* Denne metoden sjekker harmonisk faktor, og om denne er over 5% forskjellig. Hvis den er mindre enn 5% blir det oppfattet som en overharmoni av den forrige frekvensen og lagt til samme «cluster». Hvis forskjellen er større enn 5% er den ikke overharmonisk og fortsetter og være sin egen «cluster».

Etter at like frekvenser og overharmoniske frekvenser er luket bort finnes den frekvensen med høyest amplitude og denne sendes videre i programmet.

### NoteSearch

Notene blir lagret som objekter med informasjon om blant annet hertz-verdi og om de er sharp/flat. Det blir også laget et HashMap *notes* i filen NoteSearch.java med frekvensverdier som brukes for å finne hvilken note som er nærmest gitt frekvens. Etter at frekvensen er funnet i PitchDec.java er det ikke sikkert denne er helt lik en eksakt notefrekvens, da kalles metoden *findNearestNote* som sammenligner differansene mellom funnet frekvens og de som finnes lagret i *notes* for å finne den noten med minste differanse mellom funnet frekvens og lagret eksakt frekvens. Denne blir så returnert for så å skrives ut på skjermen.

### Dynamisk endring av notelengde

## Oppbygging av kode

### Kodestruktur

Handler/tråder

### (viktige ting i koden, f.eks config.contex <3 )

### Sentrale metoder

#### dimens.xml

Dimens.xml inneholder alle størrelser og posisjoner for alle bilder som skrives ut på skjermen. Størrelsene er skrevet i prosent slik at skjermen ser lik ut uansett skjerm størrelse og oppløsning. Når en metode, f.eks *setX()*, trenger en størrelse må den vite hvor mange piksler av skjermen den størrelsen er. Siden alle skjermer har forskjellig antall piksler i forhold til både størrelse og oppløsning, kan ikke disse tallene hardkodes. Metoden *setX()* kaller derfor *FitToScreen()* med hvor stor prosent av skjermen den trenger. Først blir metoden «getPercent» fra MainActivity.java kalt med navnet til prosent størrelsen. «getPercent» går inn i dimens.xml og henter ut det «item »-et med det rette navnet. «getPercent» henter deretter ut floatverdien, som er en prosent verdi, og legger den i en float som returneres. Metoden sender denne float verdien til en av to svært like metoder. Disse metodene heter «returnViewHeight» og «returnViewWidth», hvor «returnVeiwHeight» returnerer piksler i forhold til prosenten av høyden og «returnViewWidth» returnerer piksler i forhold til prosenten av vidden. «setX» får da tilbake en størrelse i piksler.

#### onFocusChanged

Denne metoden ser om vinduet har fokus. Hvis vinduet mister fokus blir metoden som stopper opptaket kalt, slik at selv om man åpner menyen uten å avslutte innspillingen blir den likevel stoppet. Når vinduet får fokus igjen blir størrelsen satt på «linLayout», «lowestLayer» og bakgrunnsbildet. Dette ligger her motsetning til «onCreate» pga at når «onCreate» blir kalt finnes ikke størrelsene enda.

### XML

XML, eller Extensible Markup Language, er markeringsspråket som brukes for å sette opp brukergrensesnittet i Android apper. XML er en kombinasjon av tekst og markeringer med informasjon om teksten, som for eksempel størrelse og plassering på skjermen. Det fungerer på mange måter som HTML, HyperText Markup Language, hvor flere elementer kan nestes inn i hverandre. Store deler av brukergrensesnittet i appen er forhåndsdefinert i ulike XML-filer, mens noen deler som for eksempel knappene for å redigere noter, blir opprettet direkte i koden.

#### Layouts

Det finnes tre ulike layout-oppsett man kan bruke for å organisere et skjermbilde, de kan også nestes for å oppnå det ønskede resultatet.

Brukergrensesnittet i VeryNote er i hovedsak konstruert ved hjelp av Relative layouts. Innholdet i layouten plasseres relativt i forhold til hverandre, hvert element har sin plassering definert i forhold til andre elementer i layouten. Utenom Relative Layout blir det også brukt Linear Layout, hvor elementene blir plassert etter hverandre i en spesifisert retning, og Frame Layout som gjør det mulig å vise flere RelativeLayouts over hverandre.

#### Scrollview

Scrollview et et felt som kan rulles fram og tilbake over skjermen. Feltet midt på skjermen som viser notene er i starten et statisk felt, men etterhvert som notene blir skrevet ut vil feltet bli rullbart.

#### ImgViews

Brukes for å vise bilder til brukeren. Notene som animeres på skjermen etterhvert som brukeren synger er alle ulike «ImgView».

#### TextView

Brukes for å vise tekst til brukeren. En variasjon av dette som kalles EditText lar brukeren endre teksten, som for eksempel i feltet hvor brukeren kan lagre et egendefinert navn for filen som lagres.

#### Buttons

XML element for å opprette knapper i brukergrensesnittet. Det finnes flere ulike knapper, i VeryNote finner vi standard knapper som brukes for de ulike redigeringsknappene. Knappen som styrer metronomen er en «Switch», og en «SeekBar» brukes for å la brukeren velge en hastighet for metronomen.

### UI

#### Layers

I activity \_main.xml ligger hovedoppsettet til brukergrensesnittet. Det består av et «Frame Layout» som holder de tre «Relative Layout» -ene som blir referert til i koden. Når «Relative Layout» blir lagt til et «Frame Layout» på denne måten blir de stablet oppå hverandre, dette gjør at noen elementer kan ligge oppå andre elementer. Det «Relative Layout» -et som ligger nederst i activity\_main.xml filen er det «Relative Layout» som ligger øverst i brukergrensesnittet.

#### Scrollview animating

Funksjonen «moveLinLay» animerer det «Relative Layout» -et «linLayout» som ligger i «scrollView», ved å flytte det og å øke størrelsen på det hvert 70ms. «moveLinLay» er en «Runnable» som blir kalt med den første noten etter at brukeren har trykket på «record» -knappen. Det vil si at «linLayout» ikke beveger seg før den har fått inn en note. Når brukeren trykker på «record» -knappen neste gang vil «linLayout» bli animert til slutten selv om brukeren så på begynnelsen av innspillingen.

#### Note plassering

Etter at noten har gått gjennom PitchDec.java blir den sendt til *showPitchDetectionResult()* som ligger i MainActivity.java. *findNearestNote()* søker gjennom HashMap-et *notes* i NoteSearch.java og returnerer det noteobjektet som ligger nærmest den noten som ble returnert av PitchDec.java

Dette noteobjektet blir deretter sendt til *NotesOnScreen()*. Her blir «ImageView» -et *currentNote* og det «Relative Layout» -et «imgLayout» lagt klar til den eventuelle noten. Noten blir deretter sendt til *noteLength()* hvor det finnes ut om noten er lang nok til å bli vist på skjermen. Her sjekkes det også hvilken vei noten skal tegnes, om noten skal markeres og om noten trenger ekstra linjer. Hvis noten er lenger en 1/16 dels note blir noten, markeringen og de ekstra linjene skrevet til hvert sine «ImageView» som alle legges til «imgLayout».

Videre i *noteOnScreen*() for «currentNote» tildelt en id slik at det skal kjennes igjen ved redigering. Noten får også tildelt hvor på skjermen den skal skrives ved å søke gjennom dimens.xml filen etter navnet på noten. En lytter blir lagt til noten for redigeringen senere og «imgLayout» blir lagt til «linLayout».

Hvis *findNearestNote()* returnerer den samme noten flere ganger blir «noteLength» kalt direkte.

#### Redigering av noter

På hvert bilde som viser en note eller pause ligger en lytter av «MyTouchListener». Når brukeren trykker på en note, etter at innspillingen er ferdig, blir noten og alt tilhørende blått. Dette skjer ved hjelp av et fargefilter som legges til alle «barnene» til forelderen til noten, dvs alle notens «søsken» og den selv. Når en note blir valgt, blir også en del knapper laget og lagt ut på skjermen. Disse knappene går inn under tre kategorier.

##### Delete Note

Etter at brukeren har trykket på slett note knappen, begynner den med å identifisere noten og dens plass i «allNotesForXML», under navnet «allNotes» i denne filen, og endrer lengden til noten til «» i stedet for den lengden den har. Grunnen for dette er at ExportXML bare inkluderer de objektene som har en lengde, slik at når en note ikke har lengde blir den ikke lagt til i XML-filen. Etter det blir alle bildene som var markert med blått slettet fra brukergrensesnittet og knappene blir også gjemt igjen.

##### ♭ /#/n

Når brukeren trykker på en av disse tre knappene, sjekkes det først om noten allerede har dette fortegn og det skal fjernes eller om dette fortegnet skal legges til. Hvis noten allerede har dette fortegnet blir bildet som representerer det fjernet fra brukergrensesnittet. Noten blir funnet i arrayen og noten blir satt til å ikke ha fortegn. Hvis derimot noten ikke har det valgte fortegnet, må dette legges til. Først blir de andre mulige fortegnene sine bilder funnet og fjernet. Noten blir også funnet her og blir satt til å ha det korrekte fortegnet. Etter dette blir et blide med det nye fortegnet laget og lagt til brukergrensesnittet.

##### Down/Up

Når brukeren flytter noten en stamnote opp eller ned, må navnet på noten oppdateres i «allNotesForXML» -arrayet. På samme måte som de andre knappene blir noten funnet frem til og ved hjelp av «setNoteName» -metoden, som setter det nye navet til noten som den nye id-en til bildet av noten, får noten sitt nye navn. Hvis noten er markert med «#» eller «♭» fjernes disse, siden det ikke finnes noen garanti for at den noten eksisterer i denne dur/mol- en. Knappene for disse markeringene bli også oppdatert, slik at brukeren kun kan velge alternativer som finnes. Når noten flyttes kalles også metoden «fixLines», som sletter alle notelinjene og tegner de inn på nytt, med det som nå er det rette antallet. Den nye plasseringen til noten på skjermen blir funnet ved å søke etter den nåværende plasseringen i «yValueSearch», og hente ut enten den foregående (opp) eller den neste (ned) verdien i arrayet. Hvis noten er opp ned blir den forskjøvet tilsvarende

I redigeringen av noten var ideen å bruke «setName» og «setSharp», men pga at alle like noter har samme id, måtte det løses på en annen måte.

#### Meny

Menyen består av seks knapper, hvor Record og A4 alltid vises i actionbar såfremt det finnes plass på skjermen, mens de fire gjenståene knappene alltid vises som en pop-up meny. Dette styres av tag’en ifRoom i XML-filen menu\_main.xml. Funksjonaliteten til menyen styres av en switch/case som sjekker hvilken knapp brukeren trykker på, og kaller videre den tilhørende metoden(e) for valgt knapp.

## Appens virkemåte

### Hovedskjerm

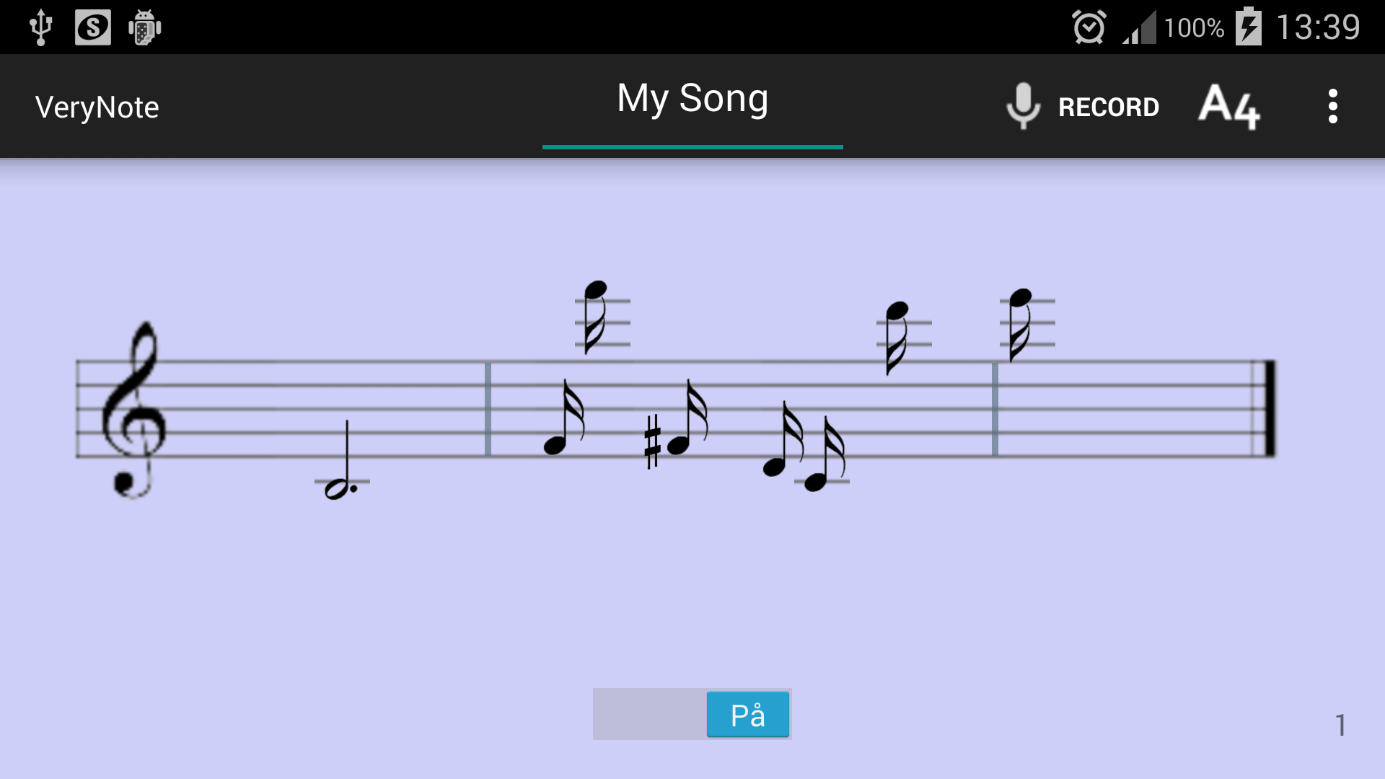
Hovedskjermen består av en «Action-Bar», navnet på appen, filnavnet og flere knapper og funksjoner. Nederst er en knapp for å slå av og på metronomen, og midt på skjermen er et notesystem med G-nøkkel.



Figur 6.1 – Skjermbilde av appens hovedskjerm ved oppstart.

#### Record

Når brukeren trykker på «record»-knappen når mikrofon sybolet står ved siden av den, vil appen begynne å analysere lyden den får inn og mikrofon sybolet vil endre seg til ett stopp symbol. Det er nå brukeren begynner å synge og appen vil vise hvilke noter brukeren synger ved å skrive noten ut i notesystemet på skjermen. Når brukeren trykker på «stop»-knappen igjen, vil appen slutte å analysere lyden som kommer inn og symbolet vil endre seg fra et stopp symbol tilbake til record og mikrofon sybolet. Brukeren kan scrolle gjennom notene og hvis brukeren trykker på «record»-knappen igjen kan han fortsette på den samme inspillingen. De nye notene vil bli lagt til å slutten av den forrige.



Figur 6.1.1 – Skjermbilde av appens hovedskjerm etter innspilling, med metronomen på.

#### A4

Til venstre for record-knappen er det en knapp med et A4 symbol. Denne knappen spiller av en frekvens på 440hz som tilsvarer en A. Denne knappen skal hjelpe brukeren å «stemme seg selv» før de begynner å spille inn en sang. Firetallet representerer oktaven.

#### Tittelfelt

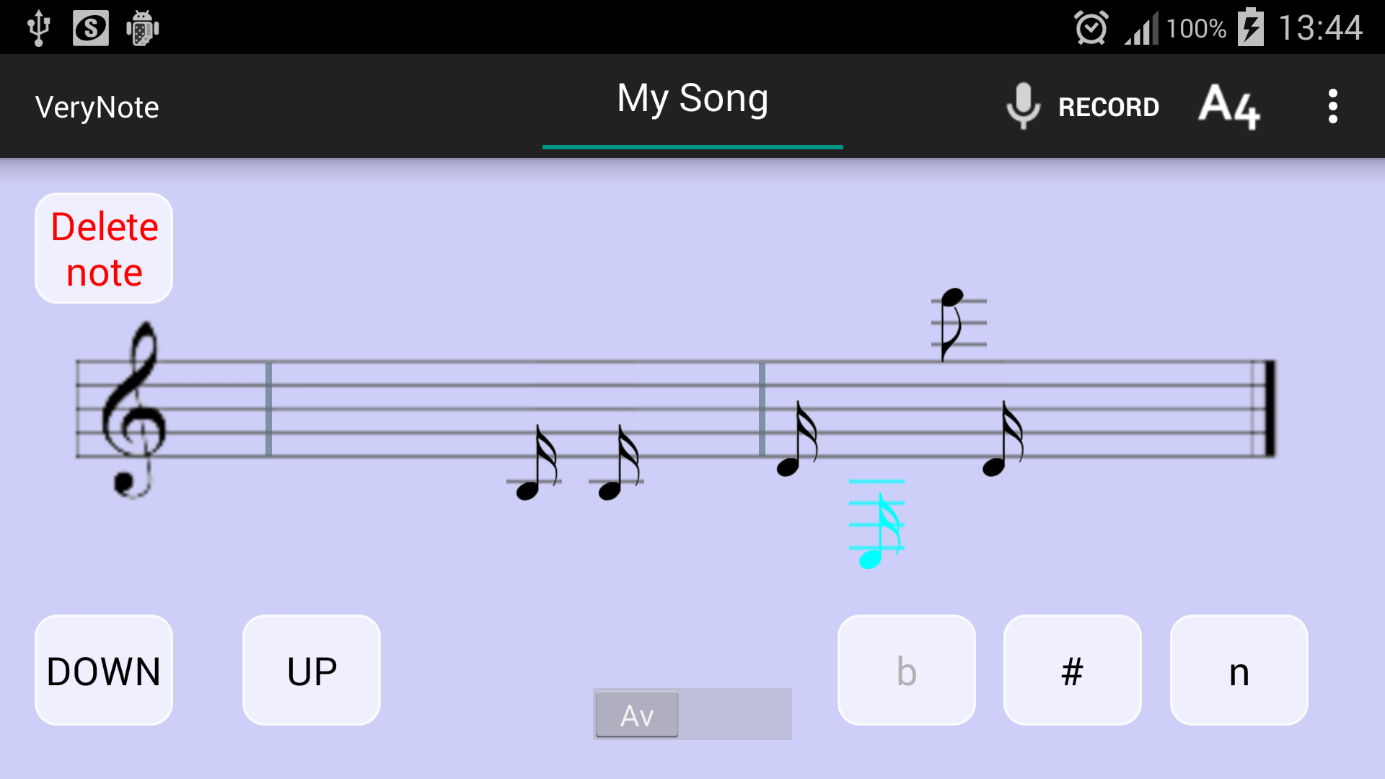
Når du åpner appen står det «Untitled» øverst. Når brukeren trykker på ordet, kan de selv legge til et egendefinert navn på filen de jobber med. Dette navnet blir tittelen på filen om brukeren velger å lagre eller dele filen.

#### Metronom

Nederst på skjermen står en av/på knapp. Denne knappen styrer metronomfunksjonen. Hastigheten er som standard 80BPM, men brukeren kan selv endre dette i settings-menyen. Når brukeren slår på knappen begynner telefonen å vibrere i gitt intervall, og det vises en liten teller nederst til høyre på skjermen.

### Redigering

Etter at brukeren har sunget inn notene sine, kan det være at de vil endre noe. Det er enkelt å legge til/ta bort fortegnene ♭/#/n, eller flytte notene opp og ned. Alle noter og pauser kan også fjernes.



Figur 6.2 – Skjermbilde etter at brukeren har valgt en note for redigering.

#### Valg av noter

Brukeren trykker på den noten de vil endre, noten vil endre farge og telefonen vil vibrere kort for å indikere at en note har blitt valgt. Når brukeren er ferdig å rediegere kan de trykke på den samme noten igjen for å velge den bort, eller trykke på en annen note for å velge den istedet.

#### Delete Note

Hvis brukeren har sunget helt feil, eller ombestemt seg kan de trykke på «delete note»-knappen. Denne knappen sletter noten

#### Down/Up

Disse knappene flytter den valgte noten opp eller ned en stamtone.

#### ♭ /#/n

Den valgte noten kan brukeren velge hvordan noten skal markeres. Hvis noten allerede har et fortegn kan denne fjernes ved å trykke på knappen som tilsier denne markeringen. F.eks hvis noten er markert med en «♭ » kan brukeren fjerne den ved å trykke på «♭ »-knappen. Den kan selvsagt legges til igjen ved å trykke på knappen en gang til. Hvis brukeren heller vil endre markeringen for noten fra «♭» til «#» kan brukeren trykke på «#»-knappen. Hvis noten ikke finnes i C-skalaen vil knappen være deaktivert, slik at man ikke kan markere en A med verken «#» eller «♭»

### Meny

Metoden for å åpne menyen varierer etter hvor ny androidtelefonen som appen kjører på er. Eldre telefoner har en «meny» knapp nederst til venstre, nyere telefoner har gjerne ikke denne knappen, da åpnes menyen ved å trykke på de tre prikkene ved siden av A4 knappen.



Figur 6.3 – Skjermbilde av appen med menyen åpen.

#### New Masterpiece

Når brukeren vil starte på nytt, kan de trykke på «new masterpiece»-knappen i menyen. Appen vil bli resatt, alle notene og tidligere innstillinger vil bli slettet. Tittelen vil også bli satt tilbake til «Untitled»

#### Save/Share MusicXML

MusicXML [2] har blitt et populært filformat for å dele musikk. Det er en standisert måte å skrive musikk i digitalt format slik at det enkelt kan deles mellom brukere av forskjellige applikasjoner.

Når brukeren trykker på «Save MusicXML»-knappen vil en xml-fil med all informationen fra appen bli lagret under det navnet brukeren selv har skrevet inn. Hvis brukeren ikke har valgt ett navn vil filen bli lagret som «Untitled». Hvis det allerede finnes en fil under samme navn, vil filen ikke bli lagret og bruker en få en medlig om at en fil ved dete navnet allerede eksisterer. Det blir da opp til brukeren om de vil gi denne filen ett nytt navn, eller finne den gamle filen og slette den.

«Share MusicXML» knappen åpner en ny meny som lar brukeren velge hvordan de vil dele xml-filen. Den kan for eksempel sendes til email, eller over bluetooth.



Figur 6.3.2 – Skjermbilde avr delingsmenyen.

#### Settings

Når brukeren trykker på «Settings»-knappen åpnes et pop-up vindu med noen valg. Her kan brukeren velge om de vil synge inn i F eller G- nøkkel og hvilken hastighet de vil ha på meteronomet. Hastigheten på meteronomet påvirker også hvor ofte man ser taktstreker når man spiller inn noter. Når brukeren velger å endre hvilken nøkkel de synger i, vil alle notene bli slettet.



Figur 6.3.3 – Skjermbilde av Settings-menyen.

## Konklusjon og videre arbeid

### Konklusjon

### Videre arbeid

Dette har vært ett svært interesangt og lærerikt prosjekt. Hadde vi hatt mer tid ville vi selvsagt ha lagt til mer funkjonalitet. Et godt eksempel på noe vi dessverre ikke hadde nok tid til å implementere er midi playback. Siden android ikke støtter javax.sound måtte vi ha skrevet vår egen ekvivalent av dette eller funnet en måte å bruke dette tiltross for androids limitasjoner, og det ville tatt mye mer tid en det vi hadde til rådighet. Vi ville også lagt til flere valg av nøkler, slik at man også kunne ha spilt inn på en C-nøkkel, i tilleg til valgene som er implementert men G-nøkkel og F-nøkkel. Appen ville gitt mulighet til å velge og å vise taktarten.

Det ville være mulig å synge inn i forskjellige dur-er og mol-er, og hatt note finne ut om de skulle være markert som «sharp» eller «flat». Hvilken dur/mol ville også selvsagt vært markert i margen sammen med nøkkelen og taktarten. Appen ville også hatt større utvalg av noter, både kortere og mer spesielle tilfeller. Den korteste noten tilgjengelig nå er 16-dels note, men de finnes 256-dels noter. Den ville hatt bindebuer og legatobuer for å symbolisere at noter skal holdes over taktstreken eller utføres som en bevegelse(ett pust), og muligheten til å legge inn dobble takt streker for å sybolisere en ny frase. Den ville også hatt sammenbundede og triol noter og muligheten til å endre og forlenge puser. Vi ville også ha lagt inn mulighet til å skrive metronomisk angivelse.

Det ville også vært naturlig å legge til en mulighet for å legge til en note, ikke bare endre på de man allerede har på skjermen. For eksempel om brukeren trykker og holder på skjermen, kunne det dukket opp en meny med valg for hvilken oktav noten skulle være, sharp/flat osv.

Vi vurderte også og utvikle et dataprogram som fungerte sammen med appen, slik at man lett kunne få filene sine til dataen og redigere dem videre der. Dette programmet kunne også hatt støtte for å spille tilbake en MIDI-versjon av det brukeren synger siden da kunne man utnyttet javax.sound biblioteket.

Knapper for å endre lengde

Mot slutten av utviklingen av appen ble vi oppmerksom på at om den lå og tok opp lyd en stund begynte animasjonen og stotre litt. Dette var mer merkbart på den eldste test-telefonen enn på de nyere telefonene. Ved hjelp av MemoryMontior i Android Studio så vi at den eldste telefonen slapp opp for minne, mens de nyere kunne fortsette å allokere nok minne etterhvert som det var nødvendig. Hadde vi hatt mer tid hadde vi vært nødt til å ta en nærmere kikk på dette. Det er ikke et stort problem slik den står nå, selv på den gamle telefonen kan man synge i over et minutt før man merker at animasjonen ikke helt holder følge. (Notene blir fremdeles lagret som de skal, det er bare animasjonen som stopper opp.) På de nyere telefonene kunne man synge i X minutt før man merket noe feil med animasjonen.

...mer?

Spør tom:

skrive i detalj om androidting? Imgview etc - JA

-kommentere kode i rapport?

Zippe ned kode, skrive kort om vedlegg(kildekode), hvordan delt opp

Skrive for skjerm/print

Sannsynligvis skjerm

Installasjonsinstruksjoner

Apk, forklaring hvis tid, el link

Forkortelser

Ikke liste, ikke figurliste, skriv sammen med første forkortelser

Forside – for hånd+scanne eller på pc

Fylle i på pc, ikke på signatur

Norsk oversetting

Skriv det som er naturlig, ikke fornorske

Legge ved kode for matlab?

Nevn at figurer er fra matlab, legge ved kode som referanse

Todo: bytte notelinjer med hjelpelinjer, markering med fortegn

Metoder med annen tekstype

## Vedlegg

Kildekode

(Kompilert app)

## Referanser

Diverse oppslag:

stackoverflow.com

xml : http://developer.android.com/guide/topics/ui/declaring-layout.html

<http://www.bidouille.org/misc/androidcharts>

<http://www.musicxml.com/>

http://music.columbia.edu/cmc/musicandcomputers/

jtransform: https://github.com/wendykierp/JTransforms

pitchdec: https://code.google.com/p/androidtuner/source/browse/src/com/example/AndroidTuner/PitchDetector.java?r=134743f47bfb0a0842f3dc73a75fed52923c8917

THE BOOK

Lærebok i fysikk : Svingninger og bølger, Arnt Inge Vistnes 2013 <http://folk.uio.no/arntvi/SvingBolg.html>

The Science of Musical Sound, John R. Pierce, 1983

Tabell over frekvenser side 21

http://www.fontspace.com/robert-allgeyer/musisync

Appicon : <http://www.clipartlord.com/free-giant-panda-clip-art/>