

EVOLUTIONARY MUSIC: CONTRAPUNTAL GENERATED SONG

BY

TEOH SIN KIAT

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONS)

Faculty of Information and Communication Technology

(Perak Campus)

MAY 2013

UNIVERSITY TUNKU ABDUL RAHMAN

REPORT STATUS DECLARATION FORM

Title: _____

Academic Session: _____

I _____
(CAPITAL LETTER)

declare that I allow this Final Year Project Report to be kept in University Tunku Abdul Rahman Library subject to the regulations as follows:

1. The dissertation is a property of the Library
2. The library is allowed to make copies of this dissertation for academic purposes.

Verified by,

(Author's signature)

(Supervisor's signature)

Address:

Supervisor's Name

Date: _____

Date: _____

UNIVERSITI TUNKU ABDUL RAHMAN

FACULTY OF _____

Date: _____

SUBMISSION OF FINAL YEAR PROJECT /DISSERTATION/THESIS

It is hereby certified that _____ (ID No: _____) has completed this final year project/ dissertation/ thesis* entitled “_____” under the supervision of _____ (Supervisor) from the Department of _____, Faculty of _____, and _____ (Co-Supervisor)* from the Department of _____, Faculty of _____.

I understand that University will upload softcopy of my final year project / dissertation/ thesis* in pdf format into UTAR Institutional Repository, which may be made accessible to UTAR community and public.

Yours truly,

(Student Name)

*Delete whichever not applicable

DECLARATION OF ORIGINALITY

I declare that this report entitled **“EVOLUTIONARY MUSIC: CONTRAPUNTAL GENERATED SONG”** is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature : _____

Name : _____

Date : _____

ABSTRACTS

This project will develop an application which Evolutionary Music Generator with Contrapuntal feature. This system can help Music Producer to produce the evolutionary music melody based on the producer need. In this project, the music producer can generate new evolutionary music melody by choosing what music pattern they need, for example rhythm of the music, range of the pitch, or how many layers of contrapuntal melody are needed. The melody generator module is applied with artificial intelligence (AI) techniques. Evolutionary search algorithm was selected to apply on the module. Evolutionary Search algorithm is a well-known technique that used to solve most optimization needed problem. The common underlying idea behind: setup initial population, perform mutation or improvement by the particular measurement. This project will develop by using incremental and iterative development methodology. Reason is there were many additional functions could be added when it is applicable.

TABLE OF CONTENTS

TITLE	i
DECLARATION OF ORIGINALITY	iv
ABSTRACT	v
TABLE OF CONTENTS	vi
LIST OF FIGURES	ix
LIST TABLES	xi
LIST ABBREVIATIONS	xii
CHAPTER 1 INTRODUCTION	1
1.1 Overview	1
1.2 Problem Statement and Innovations	2
1.3 Objectives	3
1.4 Project Scopes	4
1.5 Contributions	5
CHAPTER 2 LITERATURE REVIEW	6
2.1 Studies on existing similar system	6
2.1.1 Melisma Stochastic Melody Generator	6
2.1.2 Sean Patrick Hannifin’s Melody Generator	7
2.1.3 Comparison between both programs	8
2.2 Studies on music theory	9
2.2.1 Rhythm and Tempo	10

2.2.2 Pitch and Octave	12
2.2.3 Counterpoint	15
2.3 Studies on Artificial Intelligence (AI)	16
2.3.1 Tools for Artificial Intelligence – Search and optimization	16
2.3.2 Tool chosen for this project	19
2.4 Studies on Suitable Programming Language	20
2.4.1 JFugue – Java API for Music programming	20
2.5 Studies on Software Engineering Methodology	21
2.5.1 Structural Methodology	22
2.5.2 R.A.D. Methodology	24
2.5.3 Agile Methodology	26
2.5.4 Outcomes of Studies on Software Engineering methodology	28
CHAPTER 3 METHODOLOGY	29
3.1 Chosen Methodology	29
3.2 Application on chosen Methodology	29
3.2.1 Stage 1- Planning	30
3.2.2 Stage 2 – Analysis	31
3.2.3 Stage 3 – Design	31
3.2.4 Stage 4 –Implementation	32
3.2.5 Stage 5 – Testing	33
3.2.6 Stage 6 – Evaluation	33
3.2.7 Stage 7 – Deployment	33
3.3 Technology Involved	34
3.3.1 Hardware Requirement	34

3.3.2 Software Requirement	35
3.4 Project Plan	38
3.5 Project Analysis	41
3.5.1 Use Case Diagram	41
3.5.2 Activity Diagram	42
3.5.3 Sequence Diagram	46
CHAPTER 4 INTERFACE DESIGN & SYSTEM FEATURE	50
4.1 Interface Design	50
4.2 System Feature	52
CHAPTER 5 IMPLEMENTATION & DEPLOYMENT	55
5.1 System Implementation	55
5.1.1 How Harmony Search algorithm works	56
5.1.2 How the algorithm been implemented into the module	57
5.2 System Installation	59
CHAPTER 6 SYSTEM TESTING	60
6.1 Unit Testing	60
6.2 Functional Testing	67
6.3 Integration Testing	71

CHAPTER 7 FUTURE ENHANCEMENT & LIMITATIONS	74
7.1 System limitations	74
7.2 Future enhancement	75
CHAPTER 8 CONCLUSIONS	76
REFERFENCES & BIBLIOPGRAPHY	77
APPENDIX	80

LIST OF FIGURES

Figure Number	Title	Page
2.1	Screenshot of Melisma Stochastic Melody Generator Interface	6
2.2	The 2 Sean Patrick Hannifin's Melody Generator Interface	7
2.2.0	A quote from Charles Darwin (1871)	8
2.3.1	The notation that representing a Breve	9
2.3.2	The notation that representing a SemiBreve	9
2.3.3	The notation that representing a Minim	9
2.3.4	The notation that representing a Crochet	10
2.3.5	The notation that representing a Quavers	10
2.3.6	The notation that representing a Semiquavers	10
2.3.7	The relationship between these rhythmic values	10
2.3.8	order of the pitch	12
2.3.9	The Distance between A pitch to the same pitch on another octaves	12
2.5.1	Phases in waterfall model	21
2.5.2	Stages of Prototype Development	23
2.5.3	Incremental and Iterative Developments	25
3.4	Gantt chart for Project Development	36
3.5.1	Use Case Diagram	37
3.5.2	Activity Diagram for Case Generator	38
3.5.3	Activity Diagram for Case ExportMelody	39

3.5.4	Activity Diagram for Case PlayBack	40
3.5.5	Activity Diagram for Case PostEdit	41
3.5.6	Sequence Diagram for Case Generator	42
3.5.7	Sequence Diagram for Case ExportMelody	43
3.5.8	Sequence Diagram for Case Playback	44
3.5.9	Sequence Diagram for Case PostEdit	45
4.1	General Layout of System	51
4.2.1	User Interface Design of the core module Compose Music	52
4.2.2	Menu bar design of the system	53
4.2.3	Main menu design of the system	53
4.2.4	Interface Design of the Export Music Module	54

LIST OF TABLES

Table Number	Title	Page
2.5.1	Advantage and Disadvantage of Waterfall Model	22
2.5.2	Advantage and Disadvantage of Prototyping	24
2.5.3	Advantage and Disadvantage of Incremental and Iterative	26
3.3.1.1	Hardware requirements for user	33
3.3.1.2	Hardware requirements for developer	34
3.3.2.1	Software requirements for user	34
3.3.2.2	Software requirements for developer	35
5.1.1	Further Explanations of HS algorithm	57
6.1.1	Unit Testing 1: Compose Music	61
6.1.2	Unit Testing 2 : Playback the music	62
6.1.3	Unit Testing 3 : Edit the music	63
6.1.4	Unit Testing 4 : Export the music	65
6.1.5	Unit Testing 5 : Open 3 rd -party Application	66
6.2.1	Functional Testing : Compose Music	67
6.2.2	Functional Testing : Edit a music	68
6.2.3	Functional Testing 3: Playback Music	69
6.2.4	Functional Testing 4: Export Music	70
6.2.5	Functional Testing 5: Open 3 rd -party Application	71
6.3.1	Integration testing	71

LIST OF ABBREVIATIONS

<i>AI</i>	Artificial Intelligence
<i>CD</i>	Compact Disc
<i>GA</i>	Genetic Algorithm
<i>GUI</i>	Graphical User Interface
<i>HM</i>	Harmony Memory
<i>Hmcr</i>	Harmony memory considering rate
<i>Hms</i>	Harmony memory size
<i>NI</i>	New Improvisations
<i>Par</i>	Pitch adjusting rate
<i>SDLC</i>	Software Development Life Cycle
<i>UTAR</i>	University Tunku Abdul Rahman

CHAPTER 1: INTRODUCTION

1.1 OVERVIEW

Music Industry consists of the company firm or some people that running the business by producing and selling music. The core members that making this industry survive are the music composer, who compose and perform the music. Second are the producer and the relevant professionals will involve whole music composition and creation, and then packaging and sell the recorded music. For last time hi-speed broadband internet has not emerged yet, normally the recorded music will store in physical media form (e.g. phonograph cylinder, Gramophone, compact cassette). Nowadays, everything is goes to digitize. Not only the end product, music composing also be done in a digitized environment. Many software applications are made to aid musician write their music sheet in digital form, after that the software would able to read the musical notation wrote by composer then perform rendering of the new music. The composer was no longer necessary to perform that new piece of music manually or hire a band; everything can be done by the software. Such function was made a great changing in the proportion between composer and audience. It looks like everybody also can easily write their own piece of music as their favorites, although they may not know how to play the musical instruments, physically.

Composer is a person who writes the music. Nowadays many composer they write their music by assisted with computer or another hardware. Note that a composer, who writes music, can also be person who writes the lyrics too, that we called lyricist. In fact there are quite a sizeable number of composer they does the jobs of producer, and sound engineer as well.

Since composer wrote music for more than one instruments, they are required to have a deep understanding of the instruments they are creating music for. That is why a

composer is also a musician; sometime they are able to playing several instruments to professional standards. So composers are always musicians, but being a musician does not make you a composer.

1.2 PROBLEM STATEMENT & INNOVATION

Creativity bottleneck met by music industry

Due to the convenience of nowadays technology, playing a music are no longer require complicate step such as get the CD/Tape then put it into a CD/Tape player in order to playback the music. Now everything can do in single machine, for example an MP3. We can store the music into digital storage then playback them with the software/embedded music player. This thing was same goes to the **music composition**, everything including the music script can store in software copy. We can render the music by using the particular music composition software. This kind of technology has changing the trend of music composition. And it will lead the music industry growing faster and faster. **The main problem statement was**, people (including both composer and audience) are often get used to write/compose particular melodic turn and kept repeating it especially the pop music. Gradually, the **music industry will face the creativity bottleneck** for looking new piece/pattern of melody. During the struggling of the music evolution, music industry could suffer on market shrinking because people will get bored to the music. They may feel the new music on market was just look very old-fashioned. And such case may bring a great impact of economics.

Timeline normally are short and tight to plan a new music sequence.

Let us further discuss the problem, in perspective of a composer when they are producing a new album. In normal trend, a new music album will have approximately 10 songs. Each of them must stand in different kind and different style. For such project, the composers are needed to write up approximately 10 kinds of different style music, or more than that. And as we know normally every project has a tight time constraint, to make the requirement fulfilled (conduct 10 or more than that kinds of music) the composer will normally distributed the work to another “peers” or do it themselves under pressure. Distributing the job to a group of people is a good way to cope the problem but it require more financial budget because it need to hire more composer. This solution could easily applied by the large, well budgeted company. How about the private, low-budgeted studio and the amateur? If let them working alone under pressure, the quality of the new music will unable to guarantee. People normally will not come out too much creativity idea when working under pressure unless that has some hints or some “keys” to open up their mind.

1.3 OBJECTIVES

The main objective of this project is about to help/assist the music composer/music producer to inventing the new piece/pattern of music melody. **This is to solve the first problem statement, which Creativity bottleneck met by music industry.** In normal way, music composer need to consider that whether the new melody written has been existed before or similar with others and copyrighted. And due to fast growing of music industry, this problem had become more and more serious. The overcome such problem, the proposed music melody **generator could instantly generate the new, original piece of music melody randomly, to let the composer have**

some idea to write up something new. If user found that the piece of music melody was similar with particular existing melody they can just ask the proposed music melody generator to compose another new piece of music.

In additional way, the user could easily generate the various type of music style them wish to have. That means the proposed system will **let the user to determine the input parameter.** For example, let the user to choose whether where the key note should start (key scale) with or how the rhythm should behave on the desire new music melody piece. **And this is to solve the second problem statement.**

1.4 PROJECT SCOPE

This project develops an application that able to compose new melody which is so called contrapuntal music. This project involves an artificial intelligent technique to generate melody for new proposed music piece. The generator will first randomly generate the first string of music melody and then improve the degree of usability by using the artificial intelligent technique, step by step.

For contrapuntally organized music, there must be consist of multiple layer of music melody string, combining all together plays simultaneously. In order to have better listening experience, there must be some rules in between the music note and its counterpoint. These rules have to consider in the melody generator module, the melody evolving algorithm.

To have higher probability to generate an evolutionary music that totally fresher than before, there must have the various option that could affect the music characteristic, something like pitch range, key point, rhythm type, tempo and etc. These parameters will able to modified by user, by providing them some graphical user interface (GUI), for

example a combo box which able to let user choose the whether the minor scale or major scale will be used, or both also taking in consideration.

After the algorithm generated the melody, there could able to instantly preview them. They can either instantly playback the music or save it down then playback it on later time.

If the result music is good enough to accepting it, users are able record it down into playable format for future sharing purpose. There is various file type to save, like midi format for direct play, or note file to list of note, the pitch of every note.

1.5 CONTRIBUTION

In general, this project is going to benefit music studio and composer. As we know, composer does music composition continuously, with this system, composer and producer can save a lot of times. With this system, composer and producer no need to use any cassette or compact disc to handover the music demo. It can reduce the human negligence for example avoid lost of the hardware media. In traditional, composers need to plays the new melody first then only note it down. In this activity, many paper may use up because the composer keeps alter the notes to make it better. After that he needs to device to record it down and save it in the storage media. All of these processes are apparently time consuming. This is the reason of implementing computerized and paperless system to replace current music composition process. The propose system providing more user friendly and eco-friendly environment. The system able to generate melody and provide music amendment feature to the composer, thus the composer can save up a lot of time and hardware resource to produce a new music demo. It is also very helpful regarding to the producer as the producer could receive a new music demo and produce the complete song in shorter period.

Chapter 1: Introduction

The most interesting part in this system is artificial intelligent technique which applied to support music composition. Composers or producers just need to key in the parameters, and then this system will do its job to produced out a fresh, unique music melodies.

CHAPTER 2: LITERATURE REVIEW

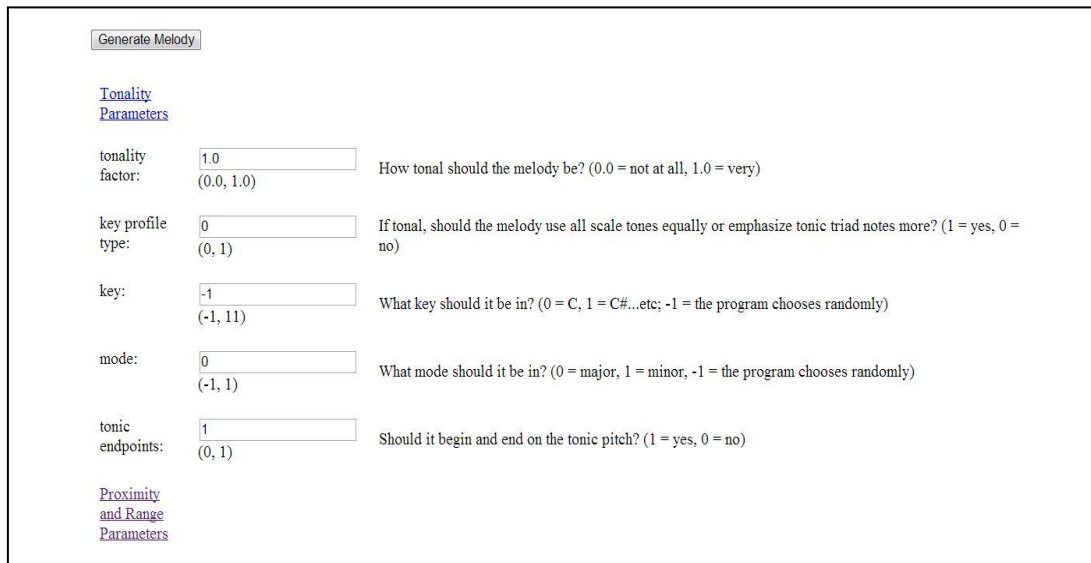
2.1 STUDIES ON EXISTING SIMILAR SYSTEM

2.1.1 Melisma Stochastic Melody Generator

The Melisma Stochastic Melody Generator is a web-based music composing tools done by David Temperley and Daniel Dominic Kaplan Sleator. David Temperley is an Associate Professor of Music Theory from Eastman School of Music, while Daniel Dominic Kaplan Sleator is a Professor of Computer Science from Carnegie Mellon University, School of Computer Science. The core algorithm was done by David Temperley while the web interface was done by Daniel Dominic Kaplan Sleator. The program was available from <http://www.link.cs.cmu.edu/melody-generator/>.

The Melisma Stochastic Melody Generator is a computer program that generates original melodies using some stochastic method. From this program, user can generate a melody using provided parameter options, with their own choice. The result melody can be returned to user in various form of file format, which including midi file format, to play directly on user machine, or with the note file, which record down the list of the notes, showing the *ontime* (in millisecond), *offtime* (in millisecond), and pitch of each note (e.g. 60 indicates the pitch was on middle C).

To play the melody in midi file, there are two approaches to do it. First one is, directly open the midi file link plays directly on your web browser, some third party player plug-in (e.g. Apple QuickTime Player) are needed to make your browser able to plays it. The second way is download the midi file to your local computer, use your local machine media player to plays the melody.



The screenshot displays the 'Generate Melody' interface. At the top left is a button labeled 'Generate Melody'. Below it, the section is titled 'Tonality Parameters'. The parameters are as follows:

Parameter	Value	Range	Description
tonality factor:	1.0	(0.0, 1.0)	How tonal should the melody be? (0.0 = not at all, 1.0 = very)
key profile type:	0	(0, 1)	If tonal, should the melody use all scale tones equally or emphasize tonic triad notes more? (1 = yes, 0 = no)
key:	-1	(-1, 11)	What key should it be in? (0 = C, 1 = C#...etc; -1 = the program chooses randomly)
mode:	0	(-1, 1)	What mode should it be in? (0 = major, 1 = minor, -1 = the program chooses randomly)
tonic endpoints:	1	(0, 1)	Should it begin and end on the tonic pitch? (1 = yes, 0 = no)

At the bottom of the interface, there is a section titled 'Proximity and Range Parameters'.

Figures 2.1: Screenshot of Melisma Stochastic Melody Generator Interface

2.1.2 Sean Patrick Hannifin's Melody Generator

The Sean Patrick Hannifin's online Melody Generator is a free online tool allows composer and songwriters to generate melodies.

This online melody generator was not only allowing user to compose a melody, there was also allow user to register their own account on this website, make it available to store and replay their previously generated melody. The simple account registration process was only requiring a username, password, and their email address. E-Mail Confirmation is required to verifying the user authenticity.

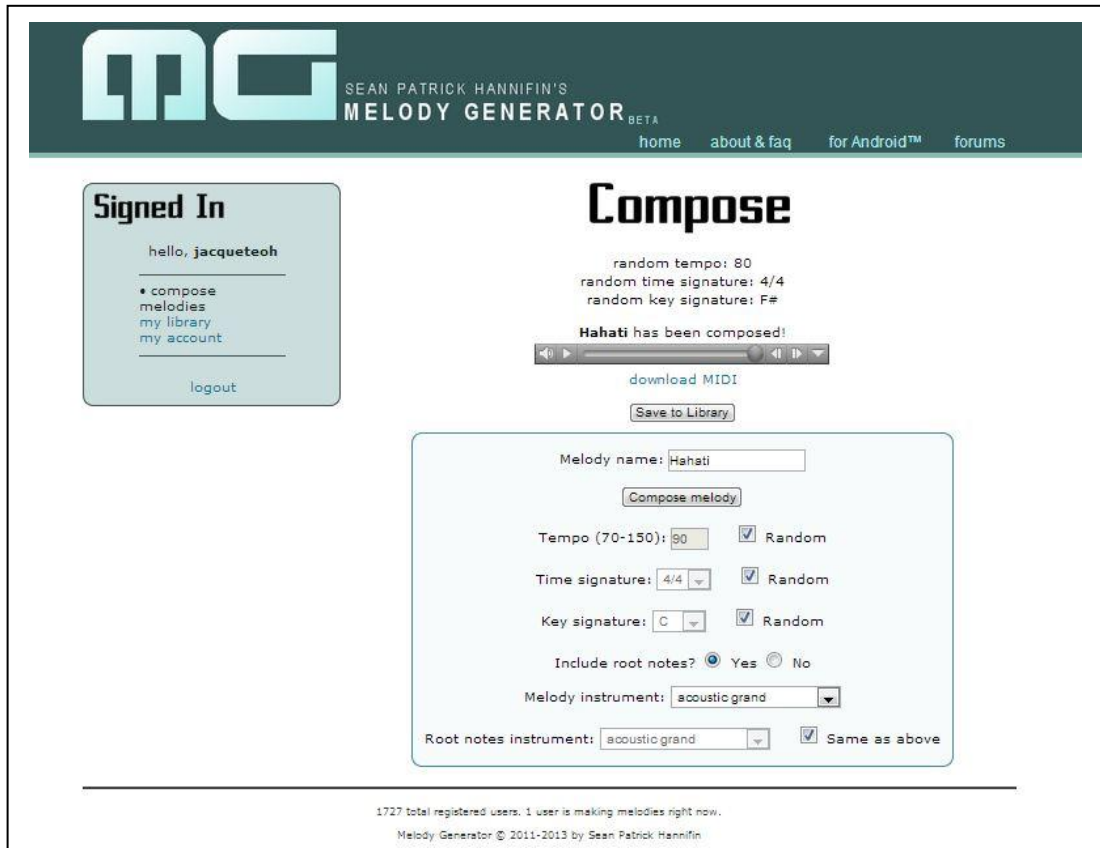


FIGURE 2.2: The 2 Sean Patrick Hannifin’s Melody Generator Interface

2.1.3 COMPARISON BETWEEN BOTH PROGRAMS

Comparing with the Melisma Stochastic Melody Generator, this Sean Patrick Hannifin’s Melody Generator was holding a main forte that was; the melody generated by Sean Patrick Hannifin’s Melody Generator can be a polyphony music melody while the melody generated by Melisma Stochastic Melody Generator is a monophonic kind of melody. However, the Melisma Stochastic Melody Generator was providing more specific parameter option compare with the Sean Patrick Hannifin’s Melody Generator.

2.2 STUDIES ON MUSICS THEORY

According to Eduardo R. Miranda & John Al Biles (2007) Evolutionary Musicology is a subfield of bio-musicology. It justifies the psychological mechanisms of music perception and production in evolutionary theory. Including vocal communication in non-human animal species, theories of the evolution of human music and cross cultural human universals in musical ability and processing.

The history of evolutionary musicology can be traced back to **Charles Darwin**, who wrote in his book, named **Descent of Man**:

"When we treat of sexual selection we shall see that primeval man, or rather some early progenitor of man, probably first used his voice in producing true musical cadences, that is in singing, as do some of the gibbon-apes at the present day; and we may conclude from a widely-spread analogy, that this power would have been especially exerted during the courtship of the sexes,--would have expressed various emotions, such as love, jealousy, triumph,--and would have served as a challenge to rivals. It is, therefore, probable that the imitation of musical cries by articulate sounds may have given rise to words expressive of various complex emotions."

Figure 2.3.0 A quote from **Charles Darwin (1871)**

And the importance of music that describing mysterious of music could also been traced back from him, **Charles Darwin(1871)**,

“Music must be ranked amongst the most mysterious with which is endowed”.

This statement has described about music **holds a great** mysterious part and **great potential** to influence humanity among the world.

2.2.1 Rhythms and Tempo

Rhythm is one of the most important components on modern music notation, and it is a fundamental part of the music theory. Rhythmic lengths are commonly represented by differing them in the head or tail of a note.

The **Breve** is the longest rhythmic value that used by today.



Figure 2.3.1: The notation that representing a Breve.

The second longest is the half of breve, called **SemiBreve**, sometime called *whole note*:



Figure 2.3.2: The notation that representing a SemiBreve

The half of Semibreve we called **Minim**, or Half Notes :

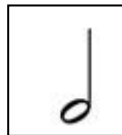


Figure 2.3.3: The notation that representing a Minim.

The further split of the rhythmic values respectively :

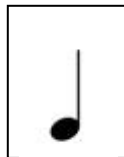


Figure 2.3.4: The notation that representing a half of Minim, **Crotchets**.



Figure 2.3.5: The notation that representing a half of Crotchets, **Quavers**.



Figure 2.3.6: The notation that representing a half of Quavers, **SemiQuavers**.

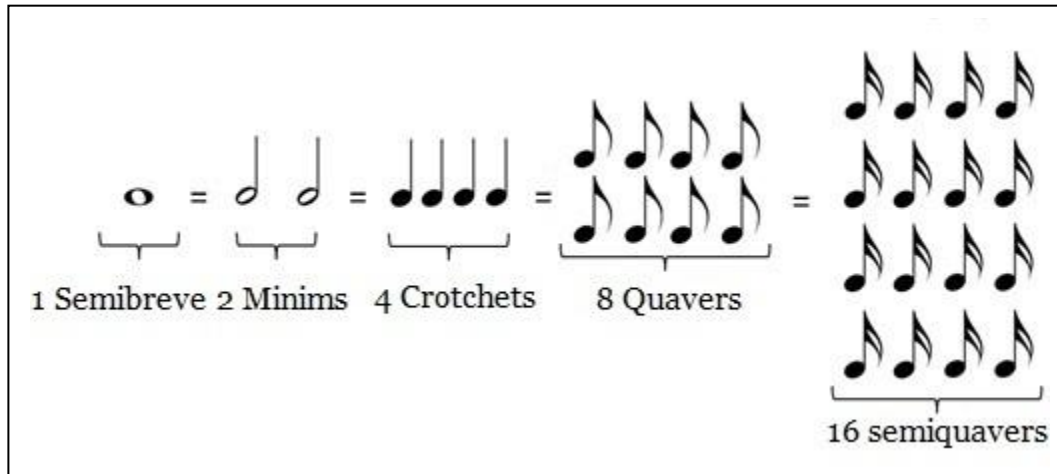


Figure 2.3.7: The relationship between these rhythmic values. (Taking from

<http://www.musictheoryhelp.co.uk/fundamentals/1/basicrhythm1.jpg>)

Tempo is a speed of a time signature that shows the number of beats in each bar/measure. Usually we determine the tempo by using “beats per minute”(or so-called BPM). For example:

$$\text{♩} = 80$$

Indicates there will 80 crotchets beat per minute. Besides that, there has another way to representing the tempo value:

- **Moderato** indicates a tempo that approximately 100 beat per minute.
- **Allegro** indicates a tempo approximately 120 beats per minute.

2.2.2 Pitch and Octaves

Pitch is the way frequencies are assigned to a relative musical scale. Common the beginner of piano player their first pitch they learned is about the Middle C. Middle C is the C key which stay in middle of piano key. The Middle A is a pitch to which Orchestras tune and is commonly called as ‘Concert Pitch A’. The Concert pitch A, also called middle A, was internationally set at 440 Hertz(Hz).

The frequencies of the pitch of the keys on a piano can be worked out by this formula:

$$F(n) = 440[2^{(n-49/12)}],$$

where **n** is the number of the key, from 1 until 88.

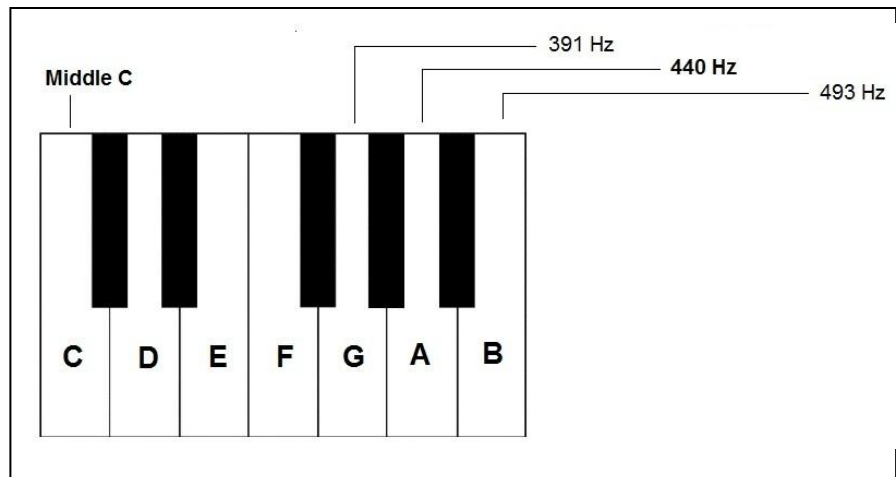


Figure 2.3.8: The order of the pitch, from middle C to middle B

An **octave** is the name that a distance between one notes to the next with the same name, there are total of **12 semitones** in an octave.

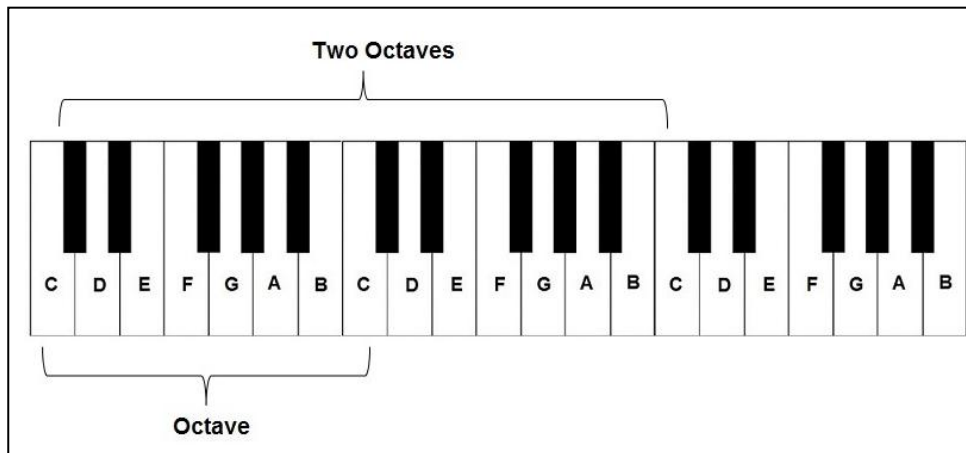


Figure 2.3.9: The Distance between A pitch to the same pitch on another octaves.

The relation between a pitch with different octave that is:

Lets,

A4 = the pitch A below an octave with middle A

A5 = the middle A, or concert pitch A

A6 = the pitch A above an octave with middle A

Frequency (A4) = $(1/2) * \text{Frequency (A5)} = (1/2) * 440\text{Hz} = 220\text{Hz}$

Frequency (A6) = $(2) * \text{Frequency (A5)} = 2 * 44\text{Hz} = 880\text{Hz}$

That's mean a pitch is **two times of the frequency** for the same pitch on **lower octave**, and **half of the frequency** of the same pitch on its higher octave.

2.2.3 Counterpoint

Note that Contrapuntal has the same meaning as Counterpoint. Counterpoint is the relationship between 2 or more independent voices, a musical texture and method of composition. This method was commonly use to construct the harmony melodies.

There are **5 species of counterpoint**, which are:

1. Note towards note.
2. Two notes toward one note.
3. Four notes toward one note.
4. Notes rhythmically offset against one another (suspensions).
5. All four of above together in “Florid” Counterpoint.

From <http://www.musictheoryhelp.co.uk>, some term has been defined:

Parallel Motion is when the parts move in the same direction, usually in the same intervals.

Oblique Motion is when one part moves and the other remains on the same note.

Contrary Motion is when both parts move in opposite directions.

2.3 STUDIES ON ARTIFICIAL INTELLIGENCE (AI)

Artificial Intelligence (AI) is a branch of computer science concerned with making computer behave like humans. The term was coined in 1956 by John McCarthy at the Massachusetts Institute of Technology.

Artificial Intelligence includes:

- **Games playing:** instructing computers to play games such as chess and checkers.
- **Expert systems:** instructing computers to make decisions in real-life situations (for example, some expert systems help doctors diagnose diseases based on symptoms).
- **Neural networks:** Systems that simulate intelligence by attempting to reproduce the types of physical connections that occur in animal brains.

- **Robotics:** instructing computers to see and hear and react to other sensory stimuli and so on.

AI can be specified as the capability of computer to do something as humans, recognize as intelligent behavior. These include:

- Searching: get a good “item” with providing some limited constraint and guideline for a plenty of population data.
- Surmounting constraints: searching a path that will match with a constricted area, for example solve a complex maze.
- Recognizing patterns: matching items that have likely characteristics, or evaluating on entity when its characteristics are not stated clearly.
- Making logical inferences: making conclusions depending with realized reasoning methods such as deduction and induction.

2.3.1 Tools for Artificial Intelligence – Search and optimization

Many problems in AI can be solved in theory by intelligently searching through many possible solutions. Evolutionary computation is the one of those. Evolutionary computation uses a form of optimization search. For example, they may start with a population of organisms, allow them to mutate and recombine, choose the most best-fit generation to survive, the result should not necessary be absolutely accurate, but it will the nearest from there (ultimate goal). Form of evolutionary computation includes:

- Swarm intelligence algorithms - ant colony or particle swarm optimization
- Evolutionary algorithms - genetic algorithms, direction-based search, harmonic search algorithm

From (Engelbrecht, Andries P 2005), **Ant colony optimization algorithm** is a method that optimizing a problem by models the action of an ant colony. These methods are designed to solve the problems that are needed to finding the goals. The artificial ‘ants’ work as simulation agents locate optimal solution by moving through a parameter space representing all possible solutions. The real ants lay down pheromones to directing each other to resources while exploring their environments, while the artificial ants does the similar jobs that they record their current positions and the quality of their solutions. So that in later simulation iterations more ants locate better solutions.

Genetic Algorithm is a method that borrows from scientific discovery about the evolutionary nature of biological genes. GA makes use of **fitness functions**, which are relationships among criteria, to grade candidates. GA also uses evolutionary methods such as **crossover** and **mutation** on chromosomes, or a chain of information to find the best examples from a very large field of possibilities. The general step of GA is:

1. Generate the initial population, the possible solutions to a problem
2. Calculate the corresponding fitness value to all possible solutions
3. Choose the current best fit solution, perform crossover and mutation
4. Recalculate the fitness value of the offspring.
5. Check whether the offspring has fulfilled the global optimum, if not, repeat from step 2.
6. The process should be terminate and re-initialize new population if the repeat step has perform too much time.

Harmony Search Algorithm is a method that uses a novel stochastic derivative which utilizes the experiences of musicians in Jazz improvisation and can be applicable to discrete variables. Instead of inclination information of an objective function, the stochastic derivative of Harmony Search determines a probability to be selected for each value of a decision variable. (Zong Woo Geem 2011)

The general steps for Harmonic Search Algorithms:

1. Generate random vector (X^1, X^2, \dots, X^{hms}) as many as **hms(harmony memory size)**, then store them in harmony memory (HM).

$$\text{Harmony Memory, HM} = \left[\begin{array}{ccc|c} x_1^1 & \dots & x_n^1 & f(x^1) \\ \vdots & \ddots & \vdots & \vdots \\ x_1^{hms} & \dots & x_n^{hms} & f(x^{hms}) \end{array} \right]$$

2. Generate a new vector X' . For each component X'_i

- With probability **hmcr**(harmony memory considering rate; $0 \leq hmcr \leq 1$), pick the stored value from HM:

$$x'_i \leftarrow x_i^{int(u(0,1)*hms)+1}$$
- With probability $1 - hmcr$, pick a random value within the allowed range.

3. Perform additional work if the value in step 2 came from HM.

- With probability **par** (pitch adjusting rate; $0 \leq par \leq 1$), change x'_i by a small amount $x'_i \leftarrow x'_i + \delta$ or $x'_i \leftarrow x'_i - \delta$ for discrete variable; or $x'_i \leftarrow x'_i + fw \cdot u(-1, 1)$ for continuous variable.

- With probability $1 - par$, do nothing.
4. If x' is better than the worst vector x^{Worst} in HM, replace x^{Worst} with x' .
 5. Repeat from step 2 to step 4 until termination criterion is satisfied.

The parameters of the harmonic search algorithm are:

- hms = the size of harmony memory. It generally varies from 1 to 100. (typical value = 30).
- $hmcr$ = the rate of choosing a value from the harmony memory. It generally varies from 0.7 to 0.99. (typical value = 0.9).
- Par = the rate of choosing a neighboring value. It generally varies from 0.1 to 0.5. (typical value = 0.3)
- δ = the amount between two neighboring values in discrete candidate set.
- fw (fret width, formerly bandwidth) = the amount of maximum change in pitch adjustment. This can be $(0.01 * \text{allowed range})$ to $(0.001 * \text{allowed range})$.

2.3.2 Outcome of studies on AI technique

As the drawback of Genetic Algorithm, to perform the optimization searching through genetic mutation, fitness function must be clearly defined and measurable. However, the fitness for a quality of music is hard to be determining because every audience stands on different music taste, there was no common taste between every audience. Meanwhile, since the Harmony Search Algorithm was an idea that came from a musician in jazz improvisations. Then it must able to using back for composing new music melody. So the Harmony Search Algorithm will be the method that choosing to develop the core module in this project.

2.4 STUDIES ON SUITABLE PROGRAMMING LANGUAGE

2.4.1 JFugue – Java API for Music programming

JFugue is an open-source Java API for programming music. JFugue makes the music programming become simple; here is an example from its official website:

```
Player player = new Player();  
player.play("C D E F G A B");
```

Beside that, JFugue have many more features:

- Music String that let user specify notes, chords, instruments, tracks.
- Music can be played at runtime, or saved to and opened from MIDI files
- Music can be sent to and received from external devices: keyboards, mixers, etc.
- A "Pattern" of music can be transformed and manipulated in interesting ways
- Support for microtonal music, intuitive rhythm tracks, anticipating musical events
- Other music parsers and renderers can be easily integrated into the JFugue architecture

JFugue is also having good applications in which music is generated at runtime, such as:

- Algorithmic, generative, aleatoric, or **evolutionary music**
- Music editors. Beat boxes, drum machines
- Jazz improvisers, mimicking classical composers, **AI in music**
- Procedural synthesis, virtual instruments, interactive software plaything

- Dynamic mood setting, adaptive music, music that depends on game state, games that require musical skill.

In advance, JFugue is the great tools for music programming because it is simple enough for controlling the music melody string, instrument selection and more than that. Make it easy to implement AI techniques within it. Thus Jfugue will be my choice as implementation tool.

2.5 STUDIES ON SOFTWARE ENGINEERING METHODOLOGY

System Development Life Cycle (SDLC) is a conceptual model used to manage the projects by involving all the necessary stages. The history of the term “Systems Development Life Cycle” is very vague but it naturally came into being since the 1960s when developers started to create programs specific to a certain need.

There is a great temptation to just resign to the fact of creating software based on the problem. Simple computing software in website is a simple problem that could be done by programmer. It is just a matter of using code to efficiently implement the software in a website. However, there are steps should be done before create the software. Initially, we must know what kind of computing software and the components that should be added. Next we need to plan the actual codes that will be used and test it properly before implementation. Without any of these steps, problem easily occurred when the program is implemented.

There are many types of methodology, such as Waterfall, Rapid Application, Interactive Development, Prototyping, Agile Method, and so on.

2.5.1 Structural Methodology - Waterfall Development

The waterfall model was invented to aid software engineer during the system development. The model will lead the software engineers building their system step-by-step. The model got its name from the pattern of its processes which resembles a waterfall. By applying this model, the project will be divided to many phases in which the phase have to be done before we go to the next phase. These phases which consist in the waterfall model are shown in the figure below.

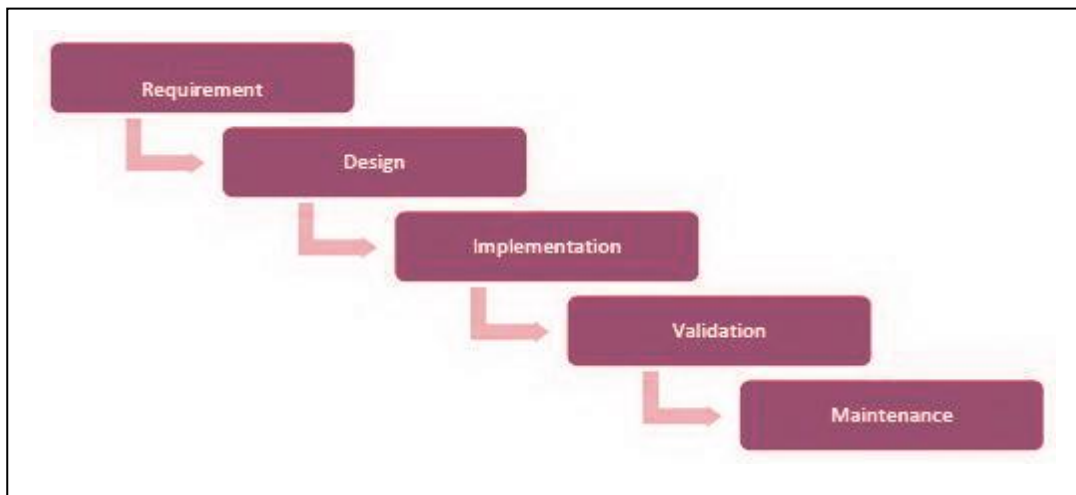


Figure 2.5.1 Phases in waterfall model

There are some advantages and disadvantages in using waterfall model in software development. S.Dilhan (2011)

Advantages	Disadvantages
The model is famous amongst the developers and therefore it is easy to use and understand facilitates communication	The time taken for every stage is very long which might cause delay in the project
Detailed descriptions of every steps allow new member in development team to understand the project and carry on with the task	A stage must be completed before going to the next stage
Detailed requirements are identified at the earlier stage before proceeding to next stage to prevent the change of requirement	Backtracking in the stage might be difficult and it will require a lot of cost
	User involvement in the project is very low as user will not provide feedback immediately

Table 2.5.1 Advantage and Disadvantage of Waterfall Model

The waterfall model has some advantages; it consists of detailed requirement which can be identified earlier before we go to the next phase to avoid the amendment of requirement. The waterfall model is common models which are used by the software engineer due to its simplicity and it is easy to understand. The detailed description in every phase helps the new recruit in the development team to understand the system better so that they could kick start their task as soon as possible. However, there are some

disadvantages in using waterfall model. Amendment of the requirement in the later stage is not possible and it will incur a very high cost. The time taken to go from a stage to another stage is very long which might cause delay in the project. Other than that, the user involvement is very low where feedback from the user is not available.

2.5.2 Rapid Application Methodology - Prototype development

The evolutionary development is also known as prototyping. Prototype is an initial version of a system which involve user involvement and evaluation of the system. The prototype only focuses on the functional requirement of the system rather than non-functional requirements. The main purpose of prototype is to access the feasibility and verify requirements. (M. Rachmadi 2010)

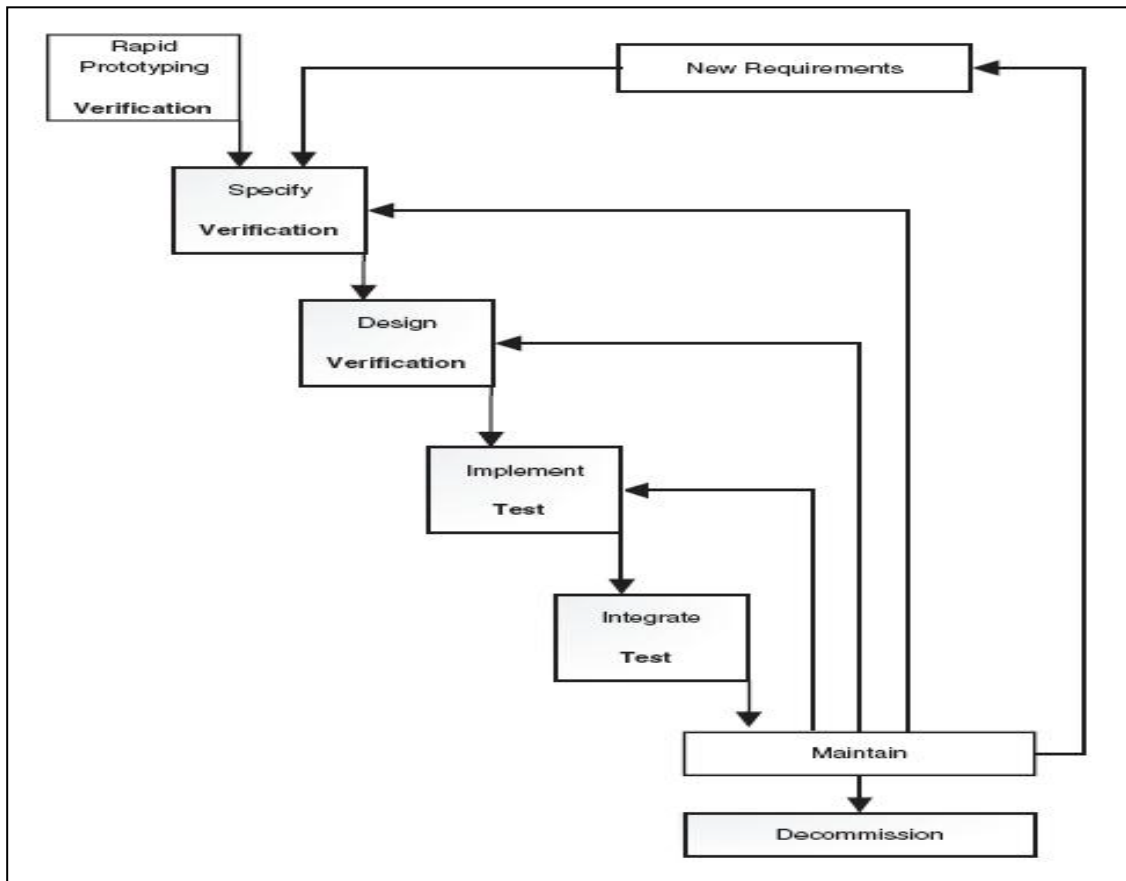


Figure 2.5.2 Stage of Prototype Development

There are some advantages and disadvantages of using prototype in developing software.

Advantages	Disadvantages
Reduce cost by eliminating the documentation	The prototype is unable to reused due to the bad quality of the code developed
User involvement to provide feedback which increase the quality of the system produced	Insufficient analysis as documentation is eliminated
Reduce development time as an initial version of system is being built in a very fast face	Source code for the system might be unorganized which makes the maintenances difficult
	User expect the final version of the system to be same as the prototype.

Table 2.5.2 Advantages and Disadvantages of Prototyping

The prototyping in the evolutionary development has a advantages such as reducing the development time due to the building of system is a very fast pace to show to the user. Prototyping focus on the programming rather than the documentation because the developers wanted to show the system to the user to get the feedback and not the documentation which the user finds them difficult to understand.

There are few disadvantages in using prototyping: there is insufficient analysis due to the elimination of documentation. The user also will assume that the final version of the system would be the same as the prototype. The maintenances of the system can be very difficult as the source code of the system is not well-organized. The prototype is

unable to be reuse as the final system because the quality of the prototype would be normally low.

2.5.3 Agile Methodology - Incremental and Iterative Development

Incremental and Iterative development is a type of software development model which allow the software engineer to build the system and later the system is being improved after testing work have been done. The basic idea of the incremental and iterative development is to build a system through repetitive cycles known as iteration and in a shorter period. At each iteration process, the design modifications are made and new functions are added.



Figure 2.5.3 Incremental and Iterative Developments

There are advantages and disadvantages in using the incremental and iterative model in developing a system. The details are summarized into the table below.

Advantages	Disadvantages
User can use the first version of the system to gain experience and make changes to the requirements in next iteration	System development might run overtime and budget
User can use the first increment of system immediately if the requirements are satisfied without the need to wait for the full system	Client need to be actively involves in the development
The highest-priority services receive the most testing which reduce the chance of software failures in the system.	

Table 2.5.3: Advantages and Disadvantages of Incremental and Iterative Development.

The advantages of using the incremental and iterative development are that the user does not need to wait for the complete system to be release before using it. As soon as the first version of the system is complete, the user can straightaway use it to perform the desired task. The user is able to use the system first and perform modification in the requirement in the next iteration. The highest priority function received the most testing and this would reduce the chance of software failure in the system. However the disadvantages of using this development would be the continuous iteration of the software might lead to over time and budget. Besides, the user involvement is very high as the user has to be with the developer throughout the process of developing the system as well as during the testing.

2.5.4 Outcomes of Studies on Software Engineering methodology

After some studies on the three methodologies, incremental and iterative development model has chosen for the proposed contrapuntal melodies generator. This approach allows the developer to build the initial system once the basic requirements has been identified and enable the user to use the system as soon as possible. Besides, the user can also provide feedback on the initial version of the system and if there are any bug or modification in requirement, changes to the system can be made. This chosen methodology will be discussed in details in Chapter 3.

CHAPTER 3: METHODOLOGY

The methodology chosen must be appropriate and suitable for the development of the system as it will be step-by-step guide that the developer must follow in order to deliver the system successfully (J. Kim. 2011). In this chapter, a methodology has been chosen to apply in the development of proposed contrapuntal melodies generator.

This application will develop by using Agile Method. First we had analysis the problem met by music industry, we formulate them then propose our idea that using computer to act as composer role, does the music composition task.

Next we had analysis the possible usable Artificial Intelligent techniques. We had chosen an algorithm which implements most likely a music composer behavior. This algorithm was named Harmony Search algorithm. We use the Harmony Search Algorithm for the core module of this project.

After that we scout the most suitable programming language to materialize the project. Finally we choose Java Platform Music Programming, JFugue. System planning and design, including UML diagram, explaining how the generator works, and User Interface Design will be draw out for later coding references. To strengthen the knowledge, few music theory books have been studied before the project has been started.

3.1 CHOSEN METHODOLOGY

After some studies on the suitable software engineering methodology in the Chapter 2, the incremental and Iterative Development is chosen. The reason for choosing this methodology is because of the nature of the system that is going to developed. There some basic functions of the systems can be built first so that the user is able use it. The

user will be able to provide feedback to the developer and improvements can be made in the next iteration (Iterative Application Development 2012). Since the requirements of the user are always changing, it would be wise to perform incremental development on the system so that more functions can be added or changed later.

3.2 APPLICATION ON CHOSEN METHODOLOGY

There are 7 stages in the Incremental and Iterative Development model which are Planning, Analysis, Design, Implementation, Testing, Evaluation and Deployment (J. Weintraub 2006). In this section every single phase in this methodology will be discussed in detail.

3.2.1 Stage 1 – PLANNING

In this stage, planning has been done. After some discussion with the project supervisor, a project name “Evolutionary Music: Contrapuntal music melody generator” is produced. Once the name has been confirmed, studies on few existing Music melody generators have been done. We know that the main purpose of existing random melody generators is about giving some new idea to the composer, and also evolve/improving the musical taste on current trends. The current situation of the musical industry is about, there are too many similar music melodies that have been produced. Users/Audience have no choice but have to accept whatever comes out from the current music company. There is a need to do something to overcome such a problem/situation, such as evolve the current music taste, giving some option to the user such as providing them a chance to become their own music composer, or anything else as long as it makes it turn over the current bad situation. As the proposed project, we are currently doing something that is mentioned in the previous sentences. The proposed music generator will not only give the current music composer a chance to attempt some idea to compose something new, it also gives a chance

to let the audience become a composer, create their own music melodies. This should give a great help on the evolvement of current music.

After some researches on the existing music industry, the problem statement has been created. Based on the problem statement, we came out with the objective of this new system which is to randomly generate a new, original music melody based on the parameter input by user. As the project supervisor suggests, the music melody generator should be able to generate the melodies with contrapuntally organized, which the current existing music melodies generator did not have such function yet.

Deliverables: Problem statement, objective and project scopes

3.2.2 Stage 2 – ANALYSIS

In this stage, analyses have been made and one of them is the requirement analysis. After assessing the existing melodies generator, there are some basic functions that should have, for example parameter input module, melody generator module.

Next, literature reviews in terms of user interface, features, music theories, artificial intelligence technique, suitable programming language, and methodology are made. This is done to ensure that the project that is delivered matches the user's requirements and expectation. By learning on the user interfaces used by the existing melodies generator, we can create a system that suits the requirement of the user. The user interface design must be simple yet attractive. Minimum usage of words is used and most important thing is that the arrangement of the media element must be appropriate. The feature of the system that had been studied is being considered and will be included into the proposed system if the feature is usable and helpful for the user in performing tasks. Once we have the information, a system requirement document which consists of the requirement of user must be created.

Deliverables: Requirement analysis, methodology, UML diagram

3.2.3 Stage 3 – DESIGN

The design stage is where the project development begins. In this stage, decision on the user interface must be made. The appearance of the system must be appropriate and the functions that the system have must be stated. The interface design of the system included the layout of the system, the position of button, the input area estimation, and the position of proper navigation picture is being displayed. Besides, the color used on the system must be appropriate. The consistency in the position of the media elements: color, font size and layout must be maintained all the time.

A test plan must be designed so that the testing can be done once the system has been developed. The test plan contained the procedures that are required to perform testing. The test data must also be included in this test plan so that the user knows what should be input into the system and check whether the output is as expected. This test plan can be used after the initial version or the first version of the system has been developed.

Deliverable: user interface design, test plan, system specification

3.2.4. Stage 4 – IMPLEMENTATION

Implementation stage is a stage where the first version of system shall begin. The initial version of the system is developed based on the system requirement created in the earlier stage. As for the user interface design, the document that drafted in design stage

will be used as a guideline. Next, the feature that must be included in the system has been identified earlier. The programming language used must be as stated in the design stage. Changes in the programming language used are prohibited unless it is necessary. The development of the system must match the methodology as planned earlier so that the development process will go accordingly.

Deliverables: initial version of system

3.2.5 Stage 5 - TESTING

Once the system has been developed, testing shall be performed as planned. The user must input the test data as written in the test plan and check whether the output is as expected. If the output is not as expected, it will be recorded so that the evaluation on the system can be done.

Deliverable: System testing

3.2.6 Stage 6 - EVALUATION

At this stage, an evaluation of the system will begin. This evaluation will check whether the system functions as intended. From the testing at the previous stage, we can check whether the system meets the user requirement. We can decide whether the system will go into the next iteration or need to perform incremental on the existing functions. There are 2 things to consider at this stage, if the user wants to have more functions then the system will go into the next iteration where the development will go back to the planning stage. Another thing is that, the user wanted to enhance the existing function which will lead the development of the system to go back to the planning stage as well

Deliverables: test result, evaluation and feedback

3.2.7 Stage 7 – DEPLOYMENT

In this stage, the system has meet the requirement from the user and functions work as intended which support user in performing tasks. Therefore, the system is ready for the deployment and ready to be used by the user. The whole processes of development must be documented so that the system can be easily maintained in the future. At the same time, it would also help the user to understand how the system can be operated.

Deliverable: Final version of system

3.3 TECHNOLOGIES INVOLVED

3.3.1 Hardware Requirement

The hardware required to run the Contrapuntal Music Melody Generator are as follow:

Description	Minimum requirements
Processor	1.6GHz or faster processor
Memory	512MB or higher
Hard Disk	5GB of available hard disk space
Video Card	DirectX 9 or higher
Resolution	1024 x 768 or higher
Audio playback	Standard Soundcard and Multimedia Speakers

Table 3.3.1.1 Hardware requirements for user

The hardware required to develop the Contrapuntal Music Melody Generator are as follow:

Description	Minimum requirements
Processor	2.0GHz or faster processor
Memory	1GB or higher
Hard Disk	10GB of available hard disk space
Video Card	DirectX 9 or higher
Resolution	1024 x 768 or higher
Audio playback	Standard Soundcard and Multimedia Speakers

Table 3.3.1.2 Hardware requirements for developer

3.3.2 Software requirement

The software required to run the Contrapuntal Music Melody Generator are as follow:

Description	Minimum requirements
Operating system	Windows XP Service Pack 3 or newer
Media player	Standard Media player
Third party Environment setup	Latest version of Java Runtime Environment (JRE)

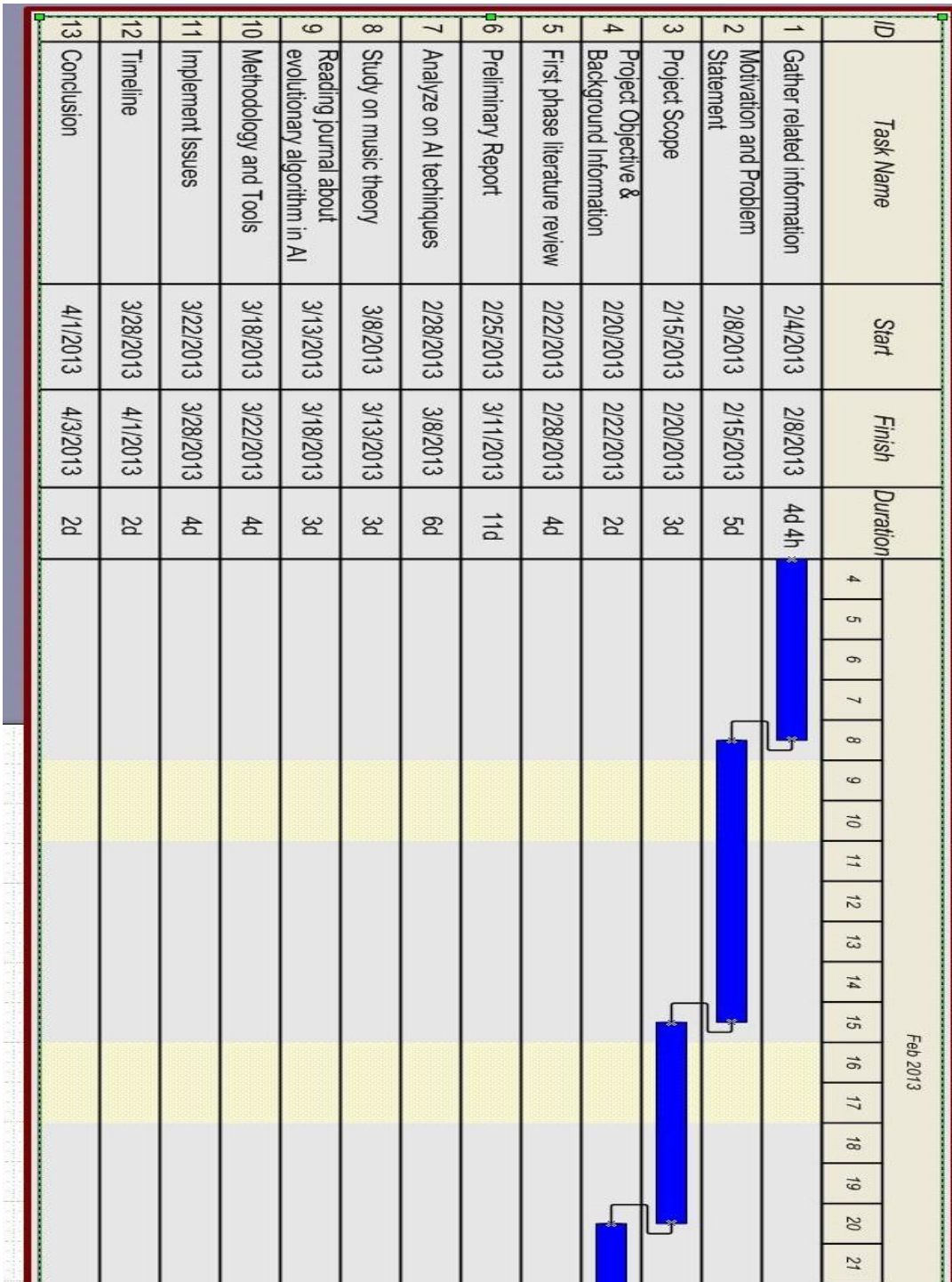
Table 3.3.2.1 Software requirements for user

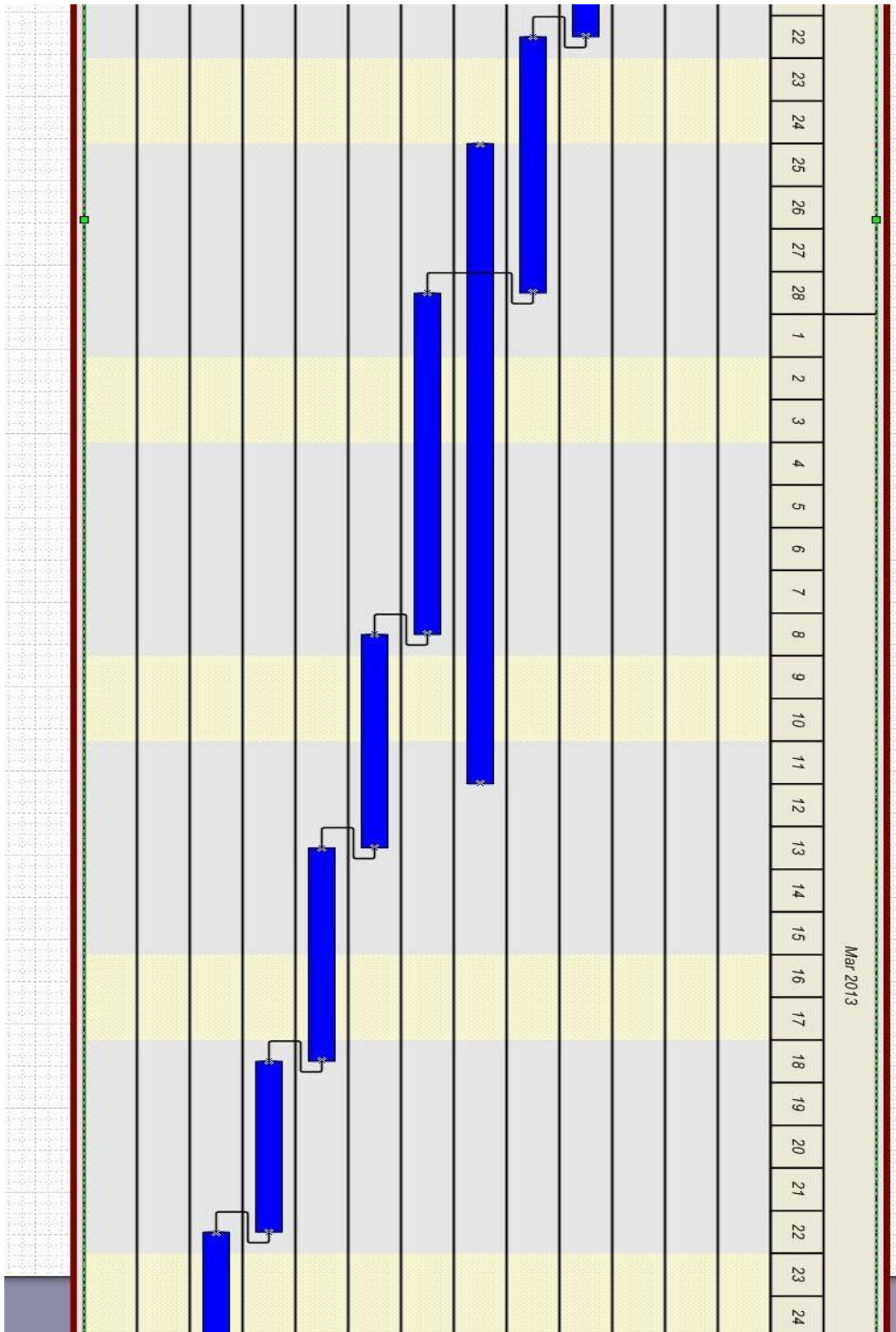
The software required to develop the Contrapuntal Music Melody Generator are as follow:

Description	Minimum requirements
Operating system	Windows XP Service Pack 3 or newer
Media player	Standard Media player
Third party Environment setup	Latest version of Java Runtime Environment (JRE)
Development Kit	Latest version of Java Development Kit (JDK)
Interface design	Netbeans IDE 7 or higher version

Table 3.3.2.2 Software requirements for developer

3.4 PROJECT PLAN





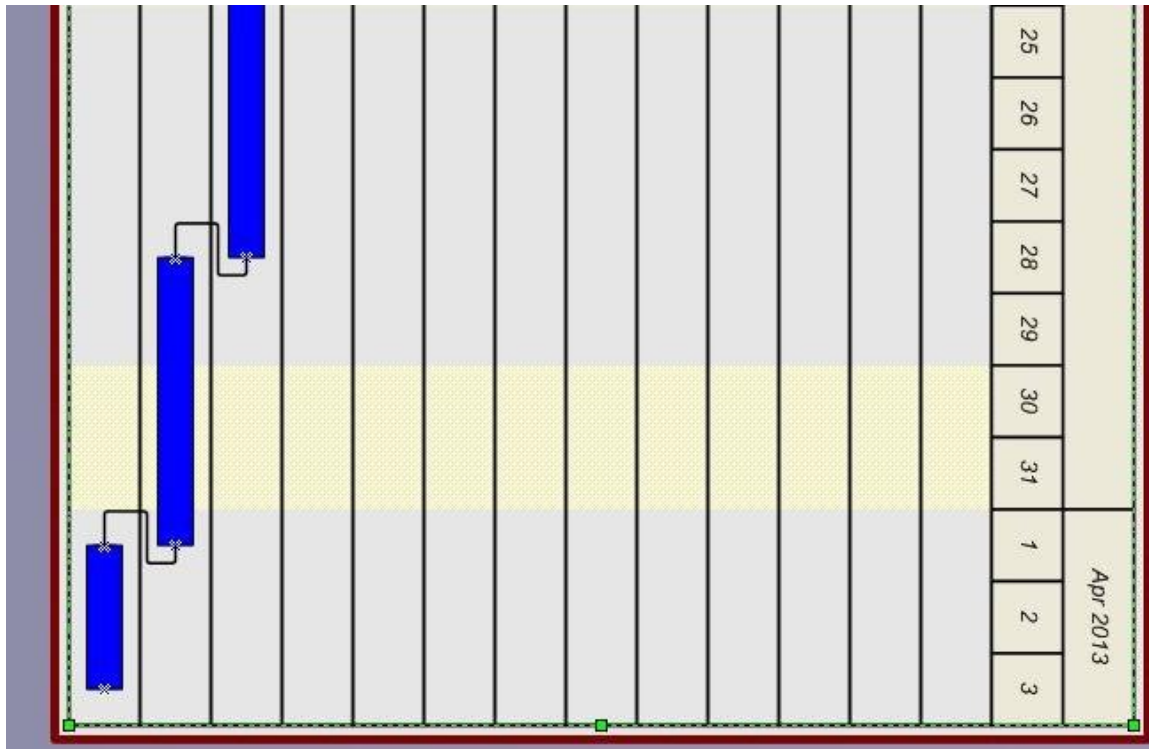


Figure 3.4 Gantt chart for Project Development

3.5 PROJECT ANALYSIS

3.5.1 UML - Use Case Diagram

A use case diagram has been prepared when in analysis stage. It consists of 4 main modules as planned.

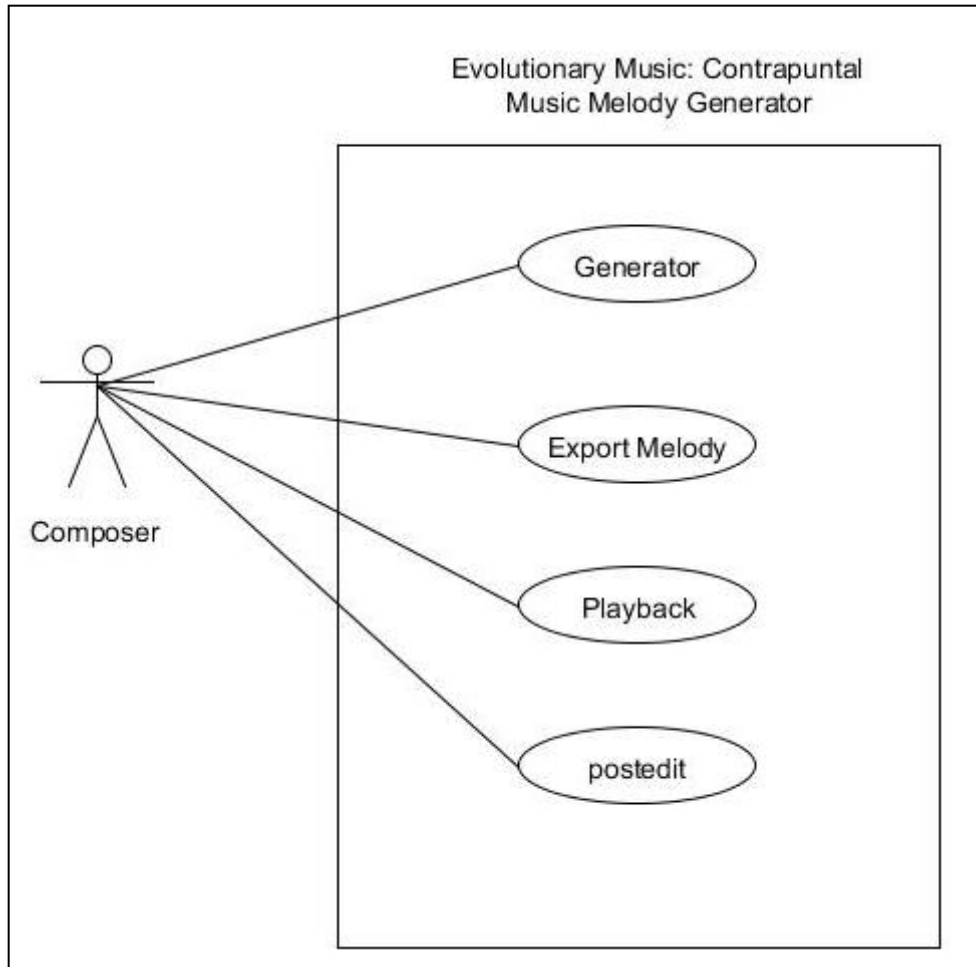


Figure 3.5.1: Use Case Diagram

3.5.2 UML – Activity Diagram

For each use case, the corresponding activity diagram has been constructed to elaborate the activity involved inside that case.

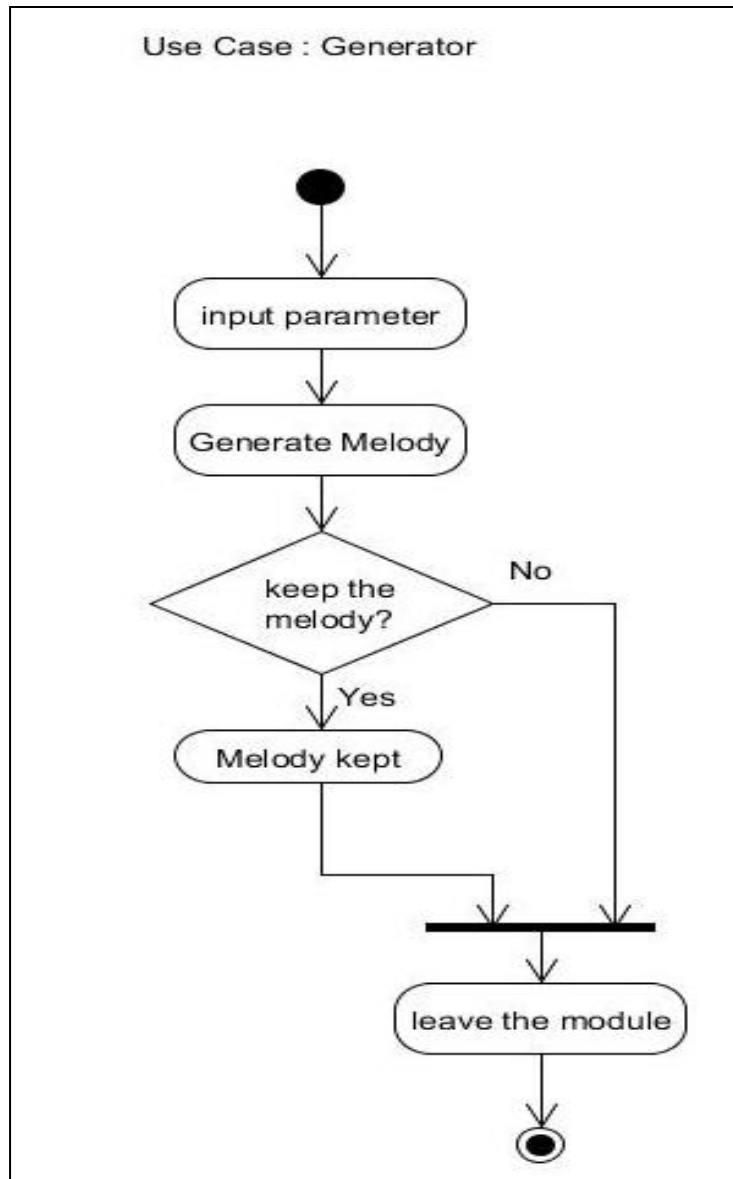


Figure 3.5.2: Activity Diagram for Case Generator

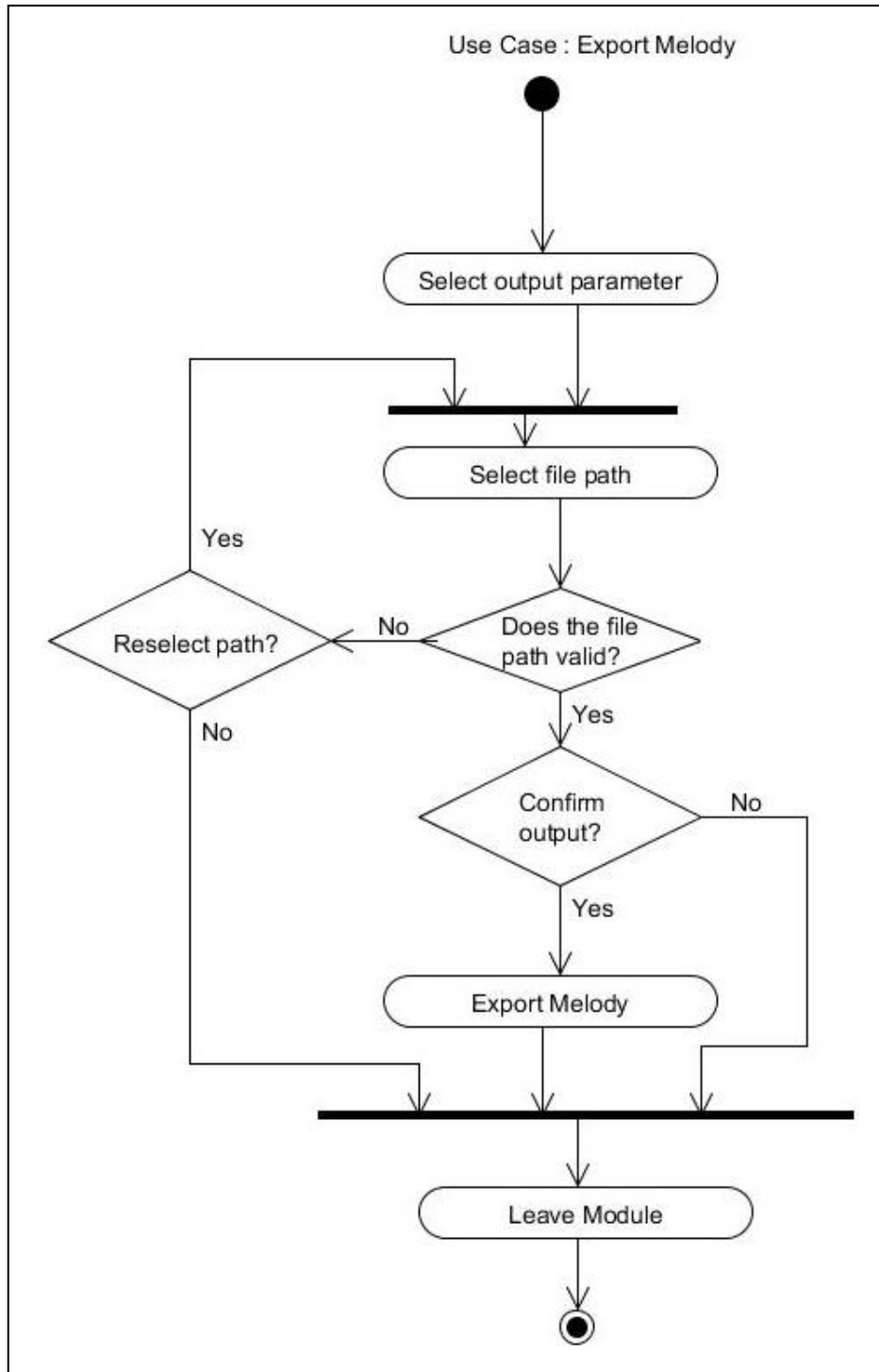


Figure 3.5.3: Activity Diagram for Case ExportMelody

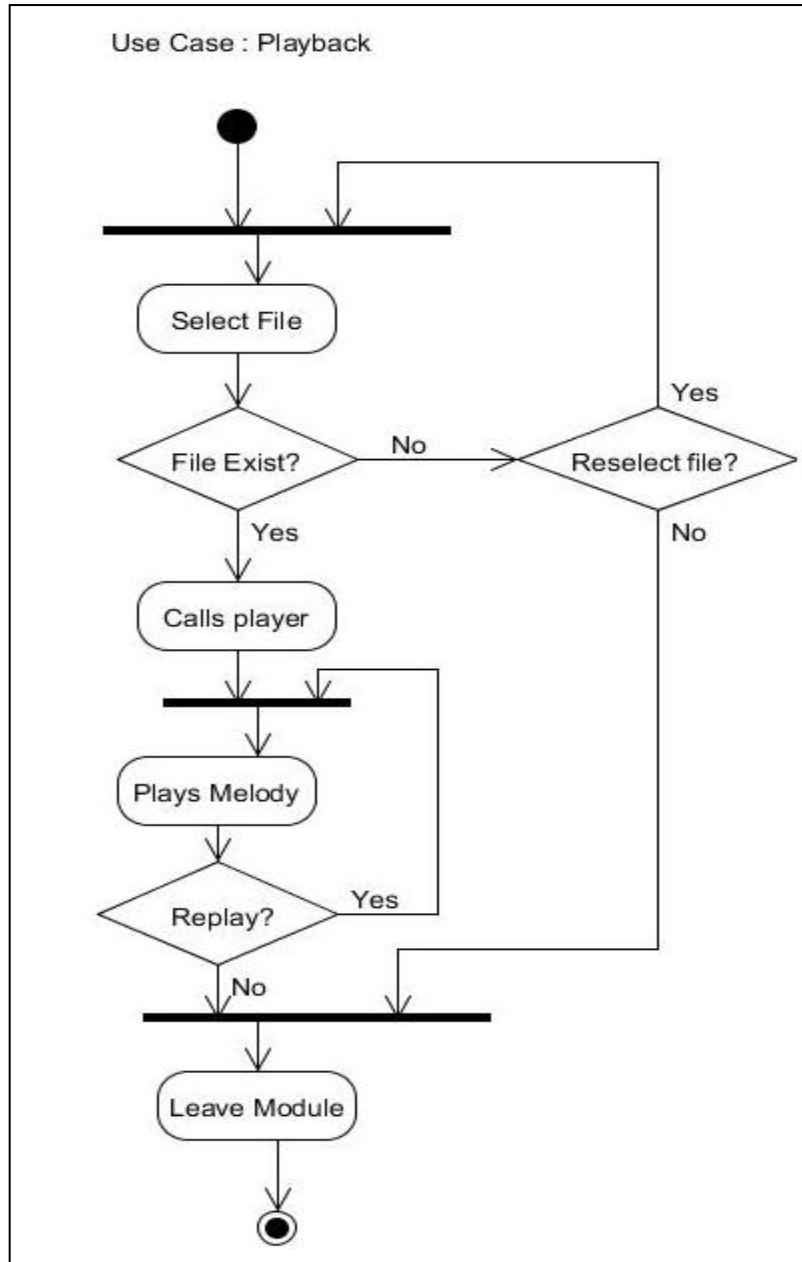


Figure 3.5.4: Activity Diagram for case PlayBack

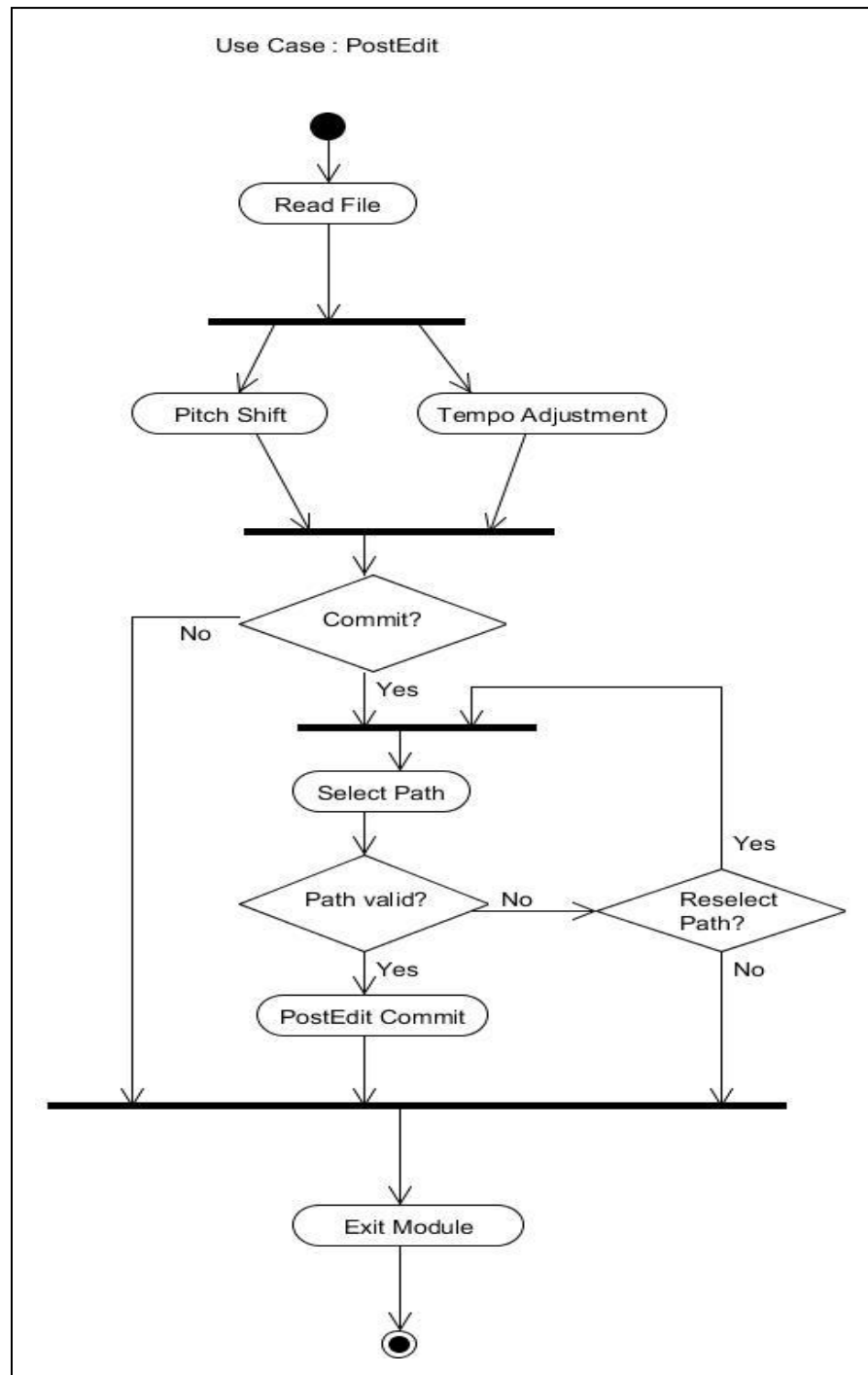


Figure 3.5.5: Activity Diagram for case PostEdit

3.5.3 UML – Sequence Diagram

For each main use case, the corresponding sequence also has been constructed to further elaborate the each process in the use case.

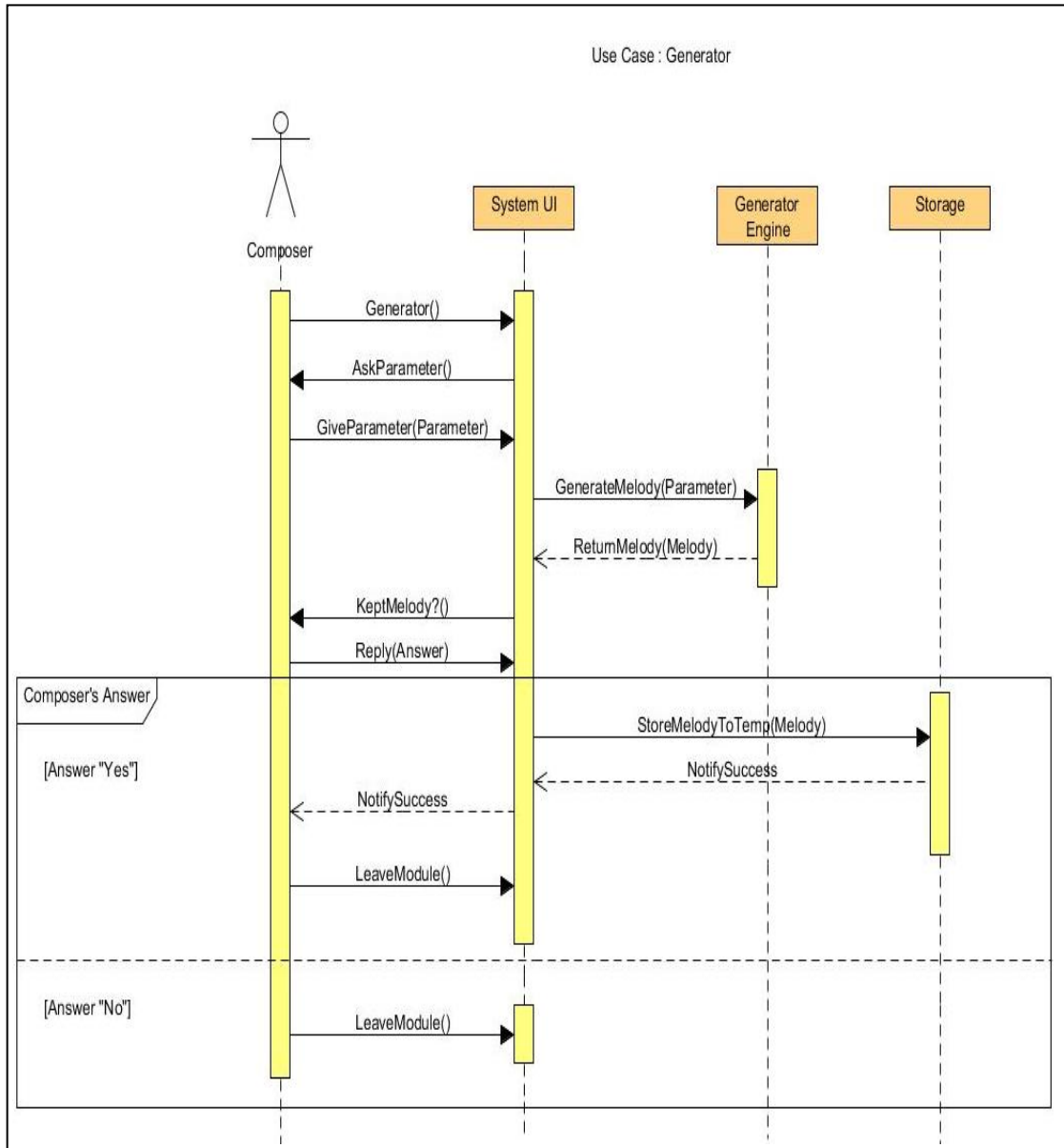


Figure 3.5.6: Sequence Diagram for Case Generator

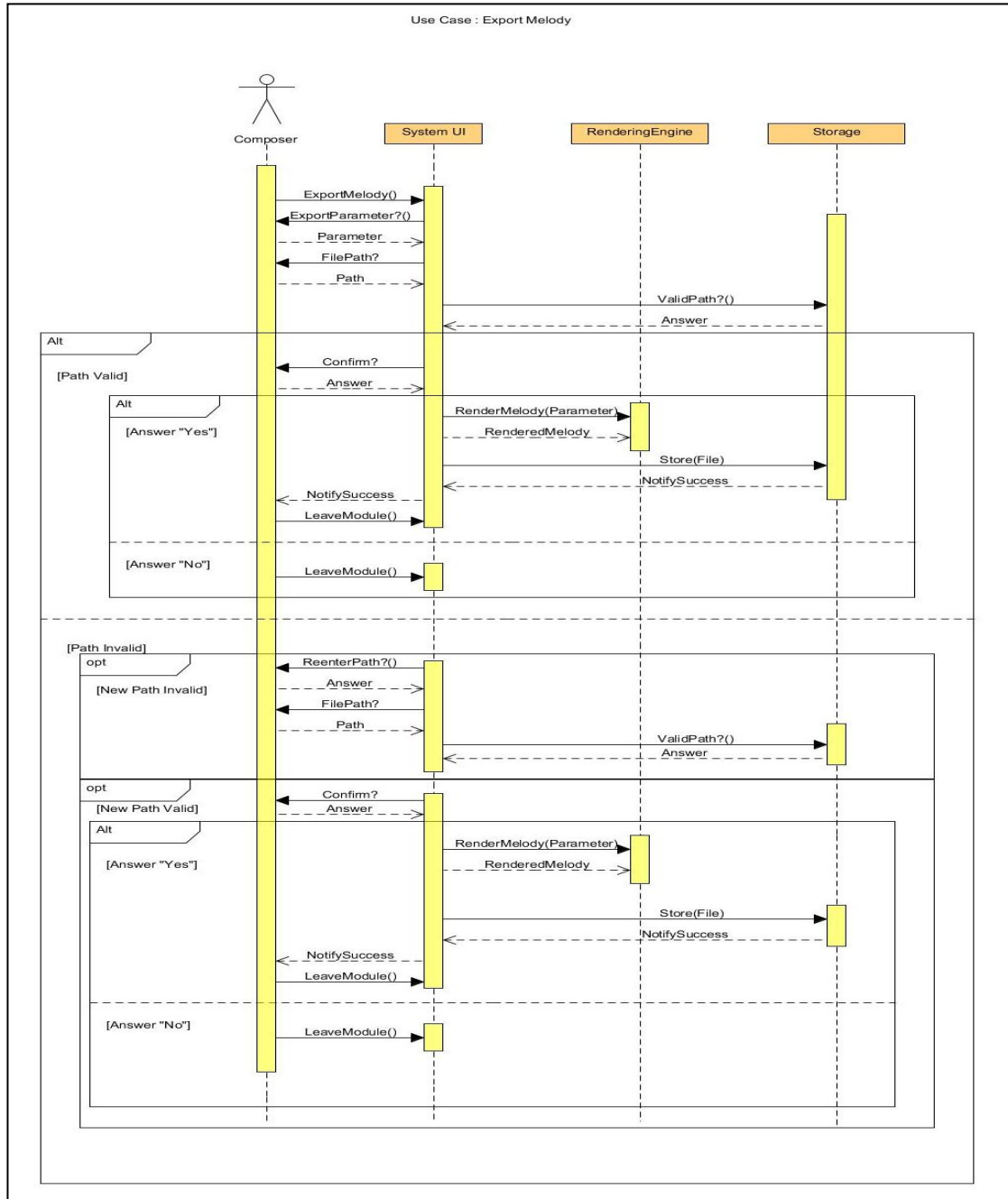


Figure 3.5.7: Sequence Diagram for case ExportMelody

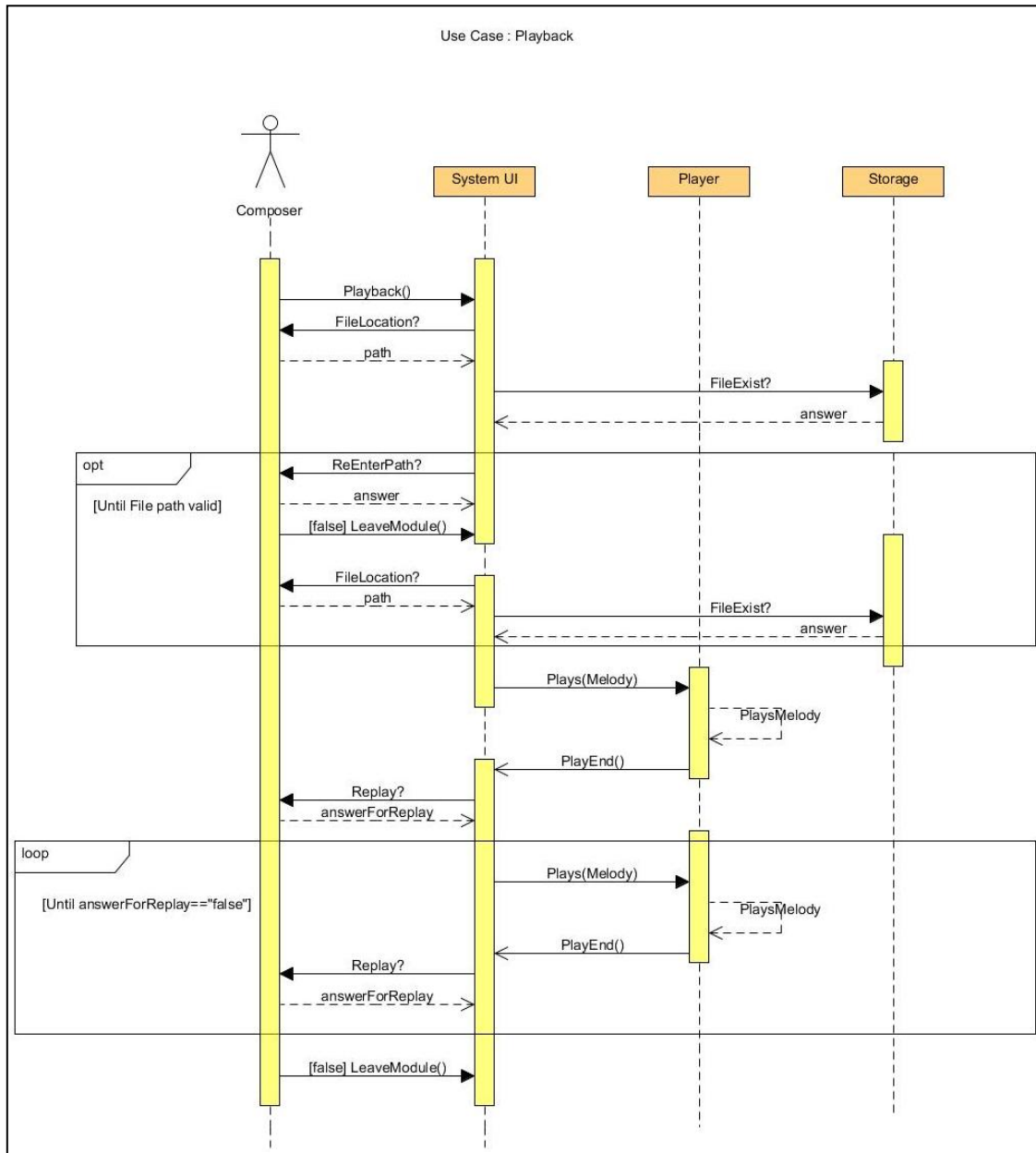


Figure 3.5.8 Sequence Diagram for case PlayBack

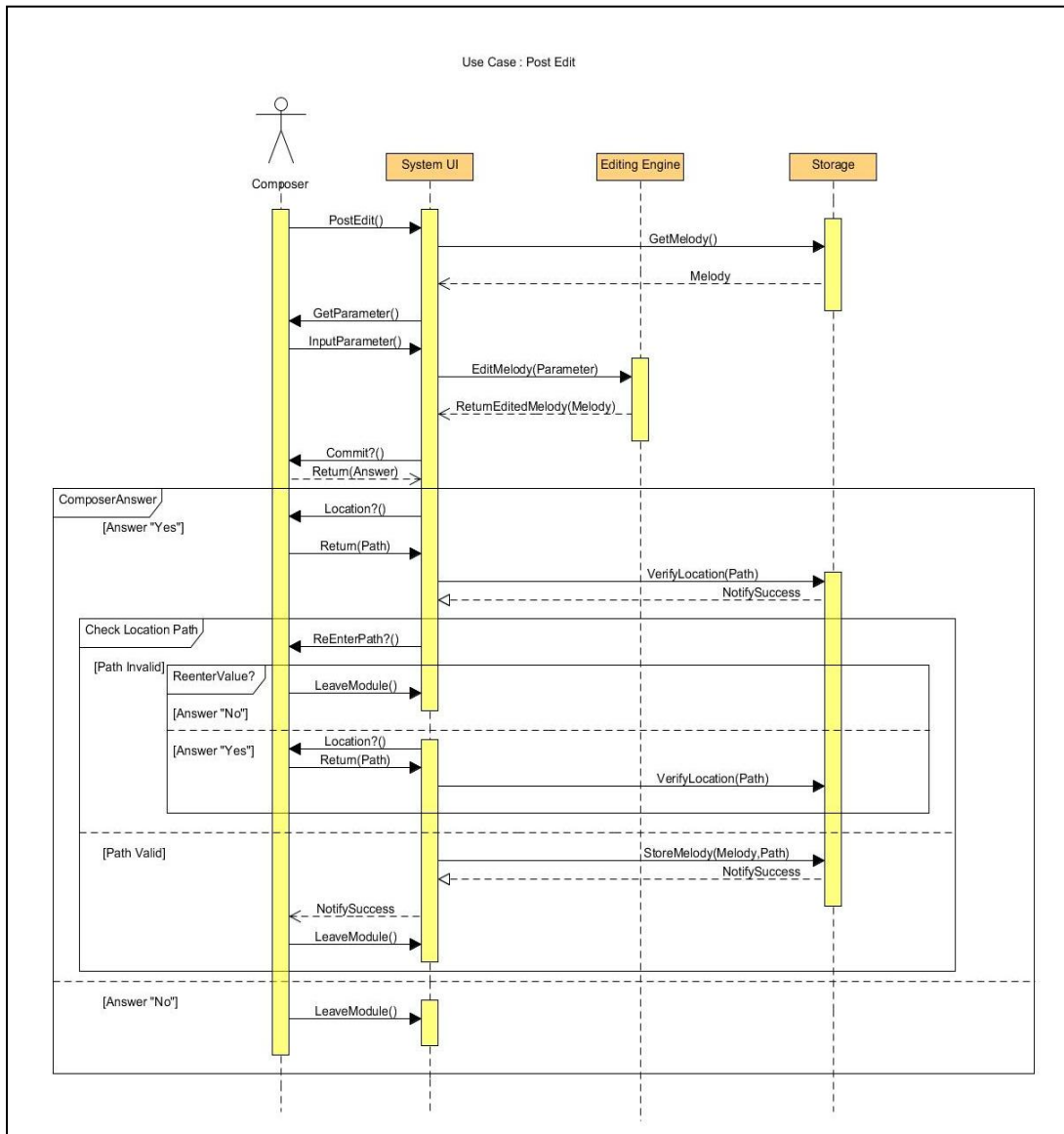


Figure 3.5.9: Sequence Diagram for case PostEdit

CHAPTER 4: INTERFACE DESIGN AND SYSTEM FEATURE

One of the most essential things that developer must do is to have the design of the system before it is being developed. A good system will have a good design to start with, so that the system created is able to cope with the requirements from the users. In this chapter, the interface design of the system would be introduced.

4.1 INTERFACE DESIGN

The interface design must be as simple as possible so that the user would not spend too much time in learning on how to use this system. The design need to be simple yet consistent so that when the user used the system. He or She does not need to learn it twice on the proper way how to use the system.

After some studies on the interface design based on several existed melody generators, an outcome has been produced. The template of the Contrapuntal Generated Song System has been decided, the layout of the interface generally would be divided into three different parts The first part is the menu bar which content some item could lead the user to the particular module, and the exit button. The second part is about the payload of the module. For example, the core module, compose music will have some forms to collect user desired parameter; the content of the export music should have the desired file choose to let user decide location to export the music files.

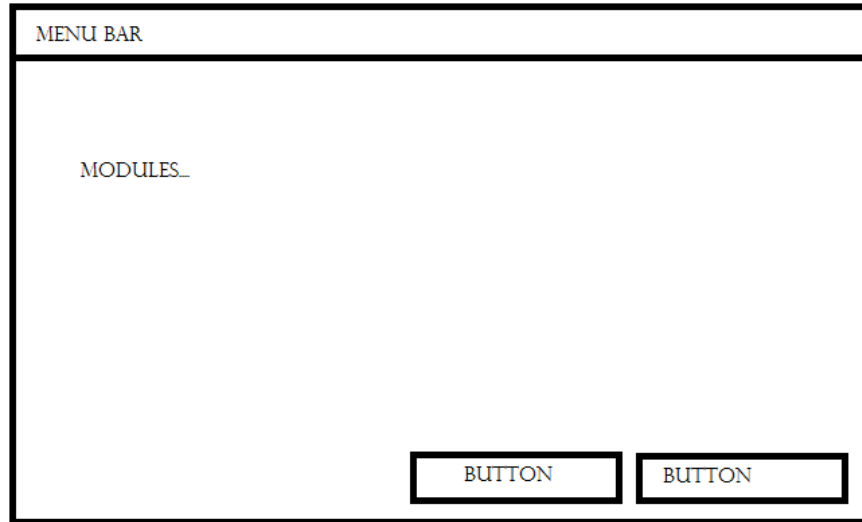
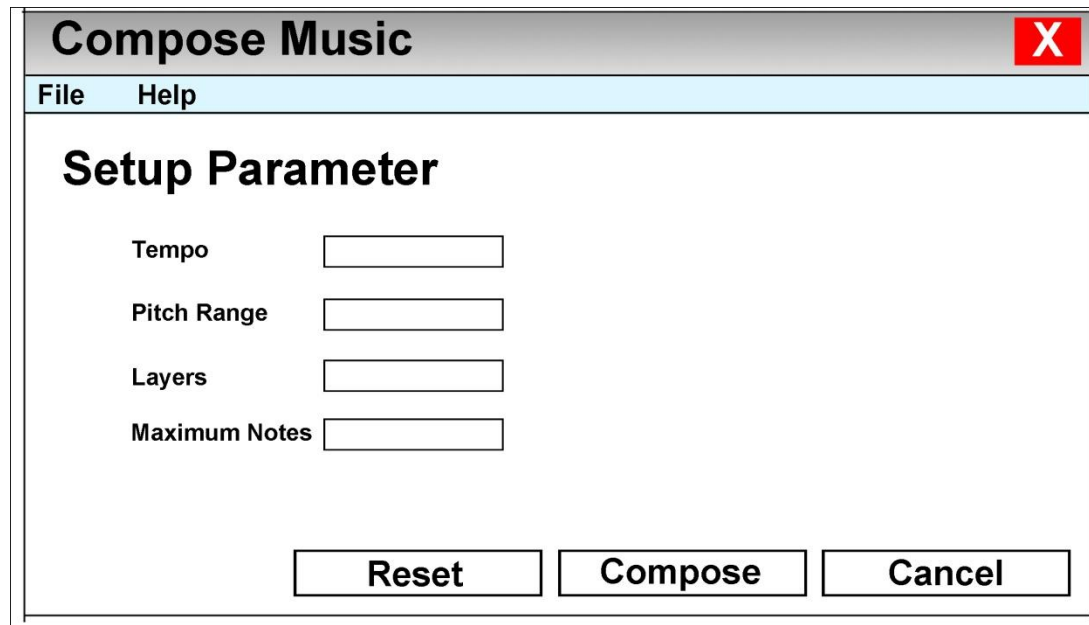


Figure 4.1 General layout of the system

4.2 SYSTEM FEATURES

The system contains four features that are unique and useful for the user so that they will have a better control over the system. One of the features is to compose music.



The image shows a sketch of a user interface window titled "Compose Music". The window has a title bar with a red "X" button in the top right corner. Below the title bar is a menu bar with "File" and "Help" options. The main content area is titled "Setup Parameter" and contains four input fields: "Tempo", "Pitch Range", "Layers", and "Maximum Notes". Each input field is a simple rectangular box. At the bottom of the window, there are three buttons: "Reset", "Compose", and "Cancel".

Figure 4.2.1: (Sketch) User Interface Design of the core module Compose Music

The first user must input the required parameter on the form provided. The system will check for the data input by user is correct and able be processed by the Melody Generator Module. If the data key in by user is incorrect for example they may accidentally input the alphabets in the field which required a numeric data, if such case happened, the system will highlight the incorrect field and prompt out a message to ask user to correct the data input.

Another feature that the system has is the menu bar that lead user to go another module directly without going back to the main menu.

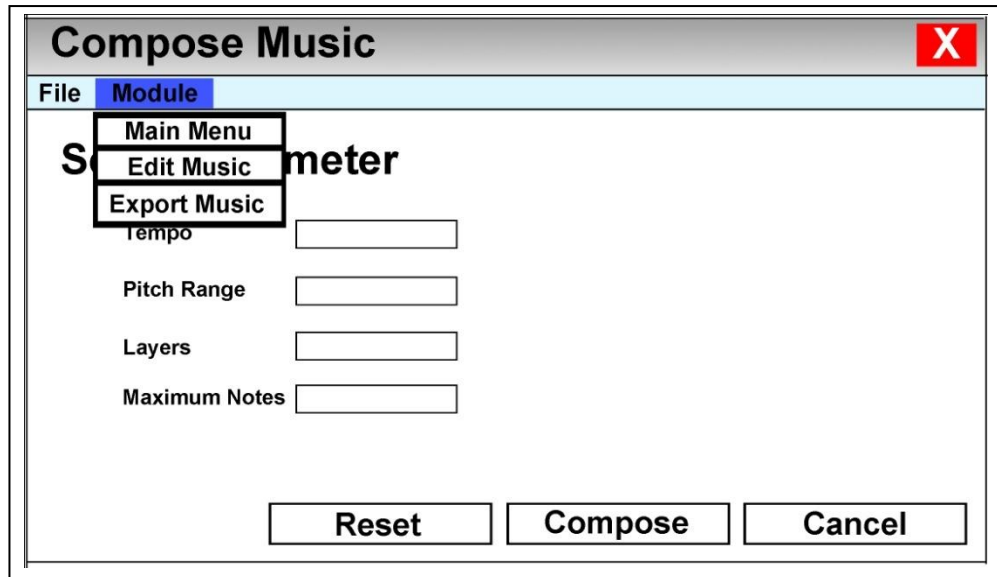


Figure 4.2.2: (Sketch) Menu bar design of the system

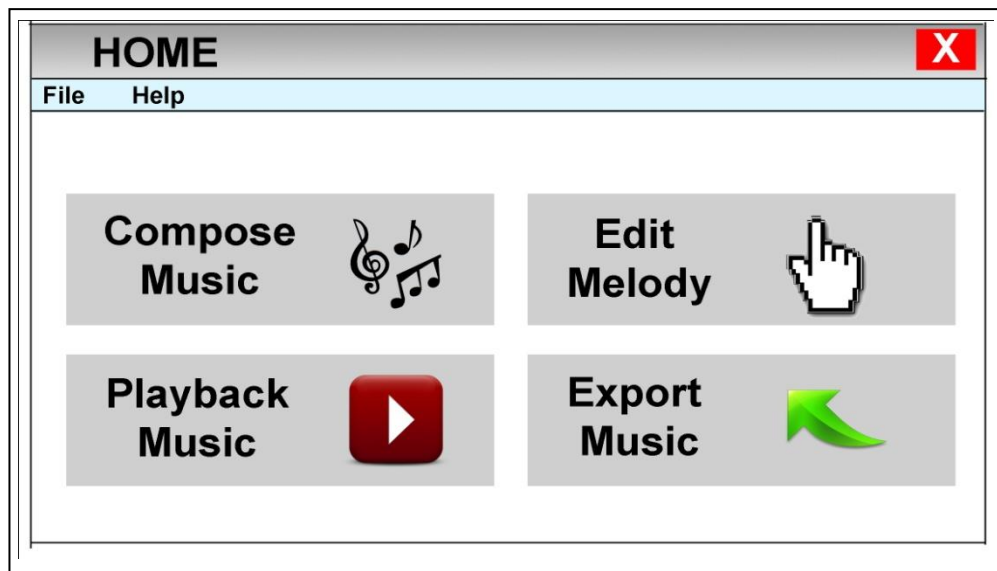


Figure 4.2.3: (Sketch) Main menu design of the system

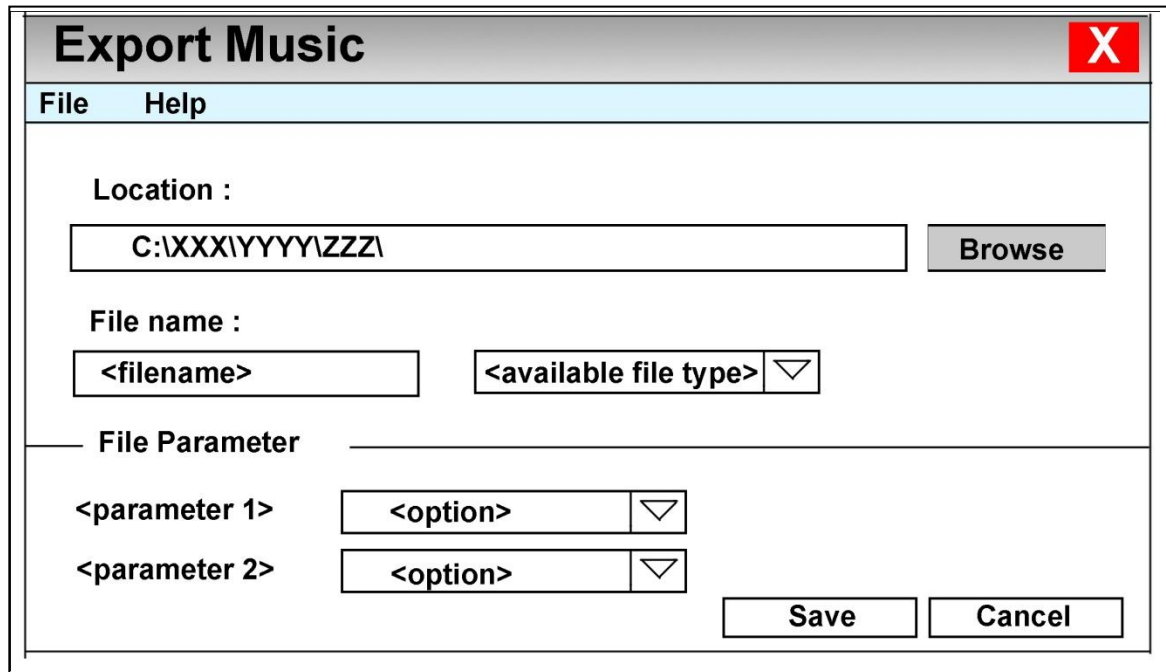


Figure 4.2.4: (Sketch) Interface Design of the Export Music Module

The Export Music will let user to choose where to export the composed music. User need to browse the file chooser to determine the location. After that they need to input the file name, a prefixed file name has been provided in case the users have no idea to create a name.

CHAPTER 5 IMPLEMENTATION AND DEPLOYMENT

The system implementation can only be started after the design work has been done. During this stage, a step-by-step development and installation would be performed on the system. Once the system has been completed, it can be used by the composer in the condition that the minimum requirement to run the system must be met.

5.1 SYSTEM IMPLEMENTATION

The first thing that had been done during the implementation was to download all the development tools as the Netbeans IDE which is a package consist of the JDK, JRE, and Java GUI builder. It is quite trouble if we coding the java program by using only a JDK and type it on a notepad then compile it by using command prompt. And there is a big problem if we do not have a Graphical User Interface Builder, without the GUI builder every parameter of the GUI need to be consider as well, such as import component, state up the container and determine position and so on. With GUI builder everything are much easier as using drag and drop to draw out the UI. Besides that, the Netbeans IDE are able to check error during the code is working out. It will tell the user whenever any error found on the code give suggestion to solve that problem. Without this checking feature, any error/mistake on the coding will only be informed during the compiling time. The next step is downloading the Java Music Programming API, JFugue, as the music library.

To start with the implementation, first thing that need to be done is the installation of development tools. Since the Netbeans IDE is a package consists of almost all the tools needed, the installation can be done in just few minutes. Next, the JFugue library has to be added into the Netbeans IDE.

Once the JFugue has been added into the Netbeans Project, the core module music generator had started to code out. The melody generator module was written out by implementing the Harmony Search Algorithm.

5.1.1 How Harmony Search Algorithm works

In this section the way how the Harmony Search Algorithm really does and how was it derived from musician, has been stated on the table below, step by step.

Step	What Jazz Musician really does	What has been derive into the HSA
1	First musician analyze the music composition order, formulate them into particular genre and determine the requirement for the chosen music genre.	Initialize the Harmony Search Algorithm and the optimization problem parameters. Such as the fitness function, $f(x)$, and the HMCR, PAR, NI, HMS
2	The musician simply write out the music piece randomly, repetitively to produce large number enough of the decision music.(from a library)	Initialize the Harmony Memory, randomly generate a number of decision variable (solution set), based on the HMS (Harmony Memory Size).
3	The musician chooses whether pick a music pieces which wrote on earlier time to improve them, or improvise a new music pieces.	Set up a uniform number (1 to 100) to determine whether to do PA (Pitch Adjustment) or a NI (New Improvisation). The probability to choose PA or NI was based on the PAR and HMCR.

4	After the musician does whether improve the music pieces or improvise new, he need to determine whether this music pieces do better than any of the music he wrote in earlier time, if does, then he replace the worse music piece by the improved/new improvised music	If the considered or new solution set has a better fitness value than any solution set of the harmony memory, the algorithm update the harmony memory by replacing the worst solution set by the considered/new solution set.
5	After a period of repetition, the musician finally submits the music which he feels that is the best among the bunch of music pieces he wrote.	Check the stopping criterion, if reach then output the harmony memory; else repeat the works from step 3 to step 5.

Table 5.1.1 Further Explanations of HS algorithm

5.1.2 How the algorithm implemented into the module

Firstly the decision variable and fitness function need to be determined. The decision variable of course will be the string of musical notes. About the fitness function, because of there no any rules that really applied to all kind of music, difficulty has been found on this section. After a long time struggle, an optimal solution eventually conducted. Some rules which using to write the *cantus firmus* has been used to derive the fitness function.

The next step is to initialize the harmony memory. The data structure that contains harmony memory was initially decided represented by using array, but after considering the flexibility required by the algorithm, the data type was eventually has been changing to vector. By using vector, any new element or dimension could be easily added or replaced. Such flexibility was not implementing well in array. The harmony memory size was prefixed

After initialized the harmony memory, the fitness values for each possible solution set has been calculated and stored at each tail of the possible solution set (add at the place after the last element of the possible solution vector). As convenient of pick out the best or worst solution set, the best/worst Fitness value index has been stated down and updated during each time of iteration.

Next take a random number by using a random number generator. Between 1 to 100, choose whether do the Pitch Adjustment or Create a new Solution based on the Harmony Memory Consideration Rate(HMCR) and Pitch Adjustment Rate(PAR). For example, we set the HMCR as 70, and PAR as 40. If the random number was fall on between 1 to 40 then we will do the Pitch Adjustment, if the random number fall on between 41 to 70 then we will do thing. Else if the random number was fall on 71 to 100 then we will create a new solution set.

After that we will calculate the fitness value of the solution set we do previously, compare with the solution set inside harmony memory. Once we found any one of the solution set inside harmony memory has lesser fitness value than the new solution set, the new solution will takes the worst solution set places as the updates of the harmony memory.

As a last stage of iteration, we will check the iteration counter to see whether it is reach the maximum iteration count. If yes then we output the HM, pick the best solution

set as the main music string, after that choose another few of the solution set as the counterpoint, the total number of counterpoint was determine by user input.

Lastly, we call out the JFugue to render the output solution as the playable music file, stored at a temporary place. To make the music convenient to made modification, the solution will store as word form in a text file.

After the core module has been wrote up, the User Interface has to start to work out. Most of the feature that planned as the previous chapter had successfully implemented. Only some of the module that requires more time that beyond the time given to work it out has been shelved temporarily.

5.2 SYSTEM INSTALLATION

The first development tool that must be installed is the JDK&JRE,JFugue and Netbeans IDE which can be downloaded from their official website in which the URL for would be <https://netbeans.org/> and <http://www.jfugue.org/> for free since it is open source software. The main purpose of installing the JDK&JRE and Netbeans IDE is to have the environment that allow java program to be complied and run. Besides, Netbeans IDE which allows software developer to design the Graphical User Interface with easily uses its built-in GUI-builder.

Another software tool used is the Adobe Photoshop CS4, the software is mainly for design the computer graphic as well as design the logo of the button.

*(*The system requirement has been stated at the previous chapter, Methodology, section 3.3 Technologies involved.)*

CHAPTER 6: SYSTEM TESTING

Once the system has been successfully developed, testing has to be performed to ensure that the system working as intended. This is also to check that the system meets the objectives stated earlier. Besides that, system testing will help in finding the error that may be hidden from the user, there are few of testing which includes the unit testing, functional testing and integration testing. The testing must be completed before it is being deploy for user to use.

6.1 UNIT TESTING

Unit testing is a kind of testing that test on each of the individual component in the system. The testing of the components includes the forms in each of the module to make sure that they are working as intended stated by Don Wells (1999). The main point in unit testing is to make sure that error does not happen during the usage of system. At the same time, if error and or bug do happen, they have to be fixed immediately and this reduces the number of fault in the system. This kind of testing can also be used to check on the input in the form to ensure that the correct format is being entered into the system and to the database thus, producing high integrity records. In order to perform the testing, a test plan has to be used.

Unit Testing 1: Compose Music

Testing Objective: To ensure the Compose Music form is working properly.

No	Test case/test script	Attribute and value	Expected result	Result
1	Input all the music parameter with correct value	Tempo : 90 Pitch Range : 54 , 78 Layers : 1 Notes Length : 30	Successfully compose a new music file and back to main menu	Pass
2	Input some invalid data into the form	Tempo : 90a Pitch Range : 54 , 78a Layers : 1 Notes Length : 30	System prompt out message that ask user to correct the data. The field that contain incorrect data will marks red color	Pass
3	Leave blank in some of the data input form	Tempo : 90a Pitch Range : 54 , NULL Layers : 1 Notes Length : NULL	System prompt out message that ask user to correct the data. The field that contain incorrect data will marks red color	Pass

Table 6.1.1: Unit Testing 1: Compose Music

Unit Testing 2: Playback the music

Testing Objective: To ensure the playback of music work properly

No	Test case/test script	Attribute and value	Expected result	Result
1	Playback music that after compose a new music	Music file exist	The music has been played	Pass
2	Playback music that before compose a new music	Music file does not exist	System will prompt out a message to inform user that there are no music exist and then brings user into the compose music module.	Pass
3	Stop Preview Music	Music is playing	Music playing will be stop	Pass
4	Stop Preview Music	Music is not playing	Nothing happen	Pass

Table 6.1.2: Unit Testing 2 : Playback the music

Unit Testing 3: Edit the music

Testing Objective: To ensure the post-edition of music work properly

No	Test case/test script	Attribute and value	Expected result	Result
1	Change the tempo with correct value	Checkbox : checked Tempo : 140 Music file exist	Tempo of the music has been changed	Pass
2	Change the tempo with incorrect value	Checkbox : checked Tempo : 140te Music file exist	A message has been prompt to ask user to correct the data input	Pass
3	Change the tempo with NULL value	Checkbox : checked Tempo : NULL Music file exist	A message has been prompt to ask user to correct the data input	Pass
4	Change the tempo without check the checkbox	Checkbox : unchecked Music file exist	The textfield will unable to key in value	Pass
5	Change the tempo without compose any music	Music file does not exist	System will prompt out a message to inform user that there are no music exist and then brings user into the compose music module	Pass

6	Change the instrument to “Drawbar Organ”	Checkbox : checked Instrument Type : Drawbar_Organ	Instrument successfully change to Drawbar Organ	Pass
7	Change the instrument to “Church Organ”	Checkbox : checked Instrument Type : Church Organ	Instrument successfully change to Church Organ	Pass
8	Change the instrument to “Harmonica	Checkbox : checked Instrument Type : Harmonica	Instrument successfully change to Harmonica	Pass
9	Change the instrument to “Guitar”	Checkbox : checked Instrument Type : Guitar	Instrument successfully change to Guitar	Pass
10	Change the instrument to “Violin”	Checkbox : checked Instrument Type : Violin	Instrument successfully change to Violin	Pass
11	Change the instrument to “Cello”	Checkbox : checked Instrument Type : Cello	Instrument successfully change to Cello	Pass
12	Change the instrument to “Piccolo”	Checkbox : checked Instrument Type : Piccolo	Instrument successfully change to Piccolo	Pass
13	Change the instrument to “Flute”	Checkbox : checked Instrument Type : Flute	Instrument successfully change to Flute	Pass
14	Edit music without check Instrument	Checkbox : unchecked	Combo box are unable to be selected	Pass

15	Edit music with check any option	Checkbox(Tempo) : unchecked Checkbox(Instrument) : unchecked	A message has been prompt to ask user choose at least one feature to do.	Pass
----	----------------------------------	---	--	------

Table 6.1.3: Unit Testing 3: Edit the music

Unit Testing 4: Export the music

Testing Objective: To ensure the post-edition of music work properly

No	Test case/test script	Attribute and value	Expected result	Result
1	Export the music with after composed music at least one time	Music file exist	Music successfully exported	Pass
2	Export the music with without composed music	Music file does not exist	System will prompt out a message to inform user that there are no music exist and then brings user into the compose music module	Pass

Table 6.1.4: Unit Testing 4: Export the music

Chapter 6: System Testing

Unit Testing 5: open 3rd-party application

Testing Objective: To ensure the 3rd-party application can be fired-up

1	Open the third-party music player	Application exist in “\lib”	Application fired up.	Pass
2	Open the third-party music player	Application not exist in “\lib”	Application unable to fire up	Pass

Table 6.1.5 Unit Testing 5: Open the 3rd-party application

6.2 FUNCTIONAL TESTING

The functional testing will take place after the unit testing. In this functional testing, the functionality of each of the module is tested. This is to ensure that the system produced meets the specifications and requirements as stated earlier (Functionality Testing, 2011). Most if not all the bugs will be uncovered in the functional testing so that less error will occur during the usage of the system. When more bug or error is discovered and fixed in the testing, it improves the overall quality of the system.

Functional Testing 1: Compose a new music

Objective: To ensure that the different kind of contrapuntal music could be produced.

No	Test case/test script	Attribute and value	Expected result	Result
1	Compose a slow tempo music with only one layer of music	Tempo : 60 Pitch Range : 54 , 78 Layers : 1 Notes Length : 30	A slow tempo, one-layered music produced	Pass
2	Compose a moderate, 3 layered of contrapuntal music	Tempo : 95 Pitch Range : 54 , 78 Layers : 3 Notes Length : 30	A moderate tempo, 3 layered contrapuntal music produced	Pass
3	Compose a fast, 2 layered of long contrapuntal music	Tempo : 135 Pitch Range : 54 , 78 Layers : 2 Notes Length : 70	A fast tempo, 2 layered long contrapuntal produced	Pass
4	Compose a fast, wide pitch range, 4 layered of long	Tempo : 135 Pitch Range : 44 , 88 Layers : 4	A fast tempo, 2 layered long and wide pitch range	Pass

	contrapuntal music	Notes Length : 75	contrapuntal produced	
5	Compose a fast, narrow pitch range, 4 layered of long contrapuntal music	Tempo : 135 Pitch Range : 44 , 88 Layers : 4 Notes Length : 75	A fast tempo, 2 layered long and wide pitch range contrapuntal produced	Pass

Table 6.2.1: Functional Testing: Compose Music

Functional Testing 2: Edit a music

Objective: To ensure that the music can be edited.

No	Test case/test script	Attribute and value	Expected result	Result
1	Edit the existing melody to a slow tempo	Tempo : 60	The music tempo has become slower	Pass
2	Edit the existing melody to a faster tempo	Tempo : 120	The music tempo has become faster	Pass
3	Change the instrument to “Drawbar Organ”	Checkbox : checked Instrument Type : Drawbar_Organ	Instrument successfully change to Drawbar Organ	Pass
4	Change the instrument to “Church Organ”	Checkbox : checked Instrument Type : Church Organ	Instrument successfully change to Church Organ	Pass
5	Change the instrument to	Checkbox : checked Instrument Type :	Instrument successfully change to	Pass

	“Harmonica	Harmonica	Harmonica	
6	Change the instrument to “Guitar”	Checkbox : checked Instrument Type : Guitar	Instrument successfully change to Guitar	Pass
7	Change the instrument to “Violin”	Checkbox : checked Instrument Type : Violin	Instrument successfully change to Violin	Pass
8	Change the instrument to “Cello”	Checkbox : checked Instrument Type : Cello	Instrument successfully change to Cello	Pass
9	Change the instrument to “Piccolo”	Checkbox : checked Instrument Type : Piccolo	Instrument successfully change to Piccolo	Pass

Table 6.2.2: Functional Testing: Edit a music

Functional Testing 3: Playback music

Objective: To ensure that the music can be played

No	Test case/test script	Attribute and value	Expected result	Result
1	Plays the music after compose it.	Music File: Existed	The music has been played	Pass
2	Stop the music while it is playing	Music is Playing	The music playing has been stopped	Pass

Table 6.2.3: Functional Testing 3: Playback Music

Functional Testing 4: Export a music

Objective: To ensure that the music can be exported.

No	Test case/test script	Attribute and value	Expected result	Result
1	Export a music to the location "C:\\"	Location : "C:\\" File name: MyContraPuntalSong	The file MyContraPuntalSong.mid has been saving into the location "C:\\".	Pass
2	Export a music to the location "Desktop\"	Location : "Desktop\" File name: MyContraPuntalSong	The file MyContraPuntalSong.mid has been saving into the location "Desktop\"	Pass
3	Export a music to the location "D:\\"	Location : "D:\\" File name: LovelySong_1	The file LovelySong_1.mid has been saving into the location "D:\\".	Pass
4	Export a music to the location "Desktop\"	Location : "Desktop\" File name: LovelySong_2	The file LovelySong_2.mid has been saving into the location "Desktop\"	Pass

Table 6.2.4: Functional Testing 4: Export Music

Functional Testing 5: open 3rd-party application

Testing Objective: To ensure the function of open the 3rd-party application are usable

1	Open the third-party music player	Application exist in “\lib”	Application fired up.	Pass
---	-----------------------------------	-----------------------------	-----------------------	------

Table 6.2.5: Functional Testing 5: open 3rd-party application

6.3 INTEGRATION TESTING

No	Test case/test script	Attribute and value	Expected result	Result
1	Compose a slow tempo music with only one layer of music	Tempo : 60 Pitch Range : 54 , 78 Layers : 1 Notes Length : 30	A slow tempo, one-layers music produced	Pass
2	Plays the music	File Exist	Music played	Pass
3	Edit the music to become faster	Tempo: 100 Checkbox: Checked	Music’s tempo has become faster	Pass
4	Change the instrument to “Drawbar Organ”	Checkbox : checked Instrument Type : Drawbar_Organ	Instrument successfully change to Drawbar Organ	Pass
5	Change the instrument to “Church Organ”	Checkbox : checked Instrument Type : Church Organ	Instrument successfully change to Church Organ	Pass

6	Change the instrument to “Harmonica	Checkbox : checked Instrument Type : Harmonica	Instrument successfully change to Harmonica	Pass
7	Change the instrument to “Guitar”	Checkbox : checked Instrument Type : Guitar	Instrument successfully change to Guitar	Pass
8	Change the instrument to “Violin”	Checkbox : checked Instrument Type : Violin	Instrument successfully change to Violin	Pass
9	Change the instrument to “Cello”	Checkbox : checked Instrument Type : Cello	Instrument successfully change to Cello	Pass
10	Change the instrument to “Piccolo”	Checkbox : checked Instrument Type : Piccolo	Instrument successfully change to Piccolo	Pass
11	Change the instrument to “Flute”	Checkbox : checked Instrument Type : Flute	Instrument successfully change to Flute	Pass
12	Change the instrument to “Piano”	Checkbox : checked Instrument Type : Piano	Instrument successfully change to Piano	Pass
13	Plays the music	File Exist	Music played	Pass
14	Export Music To “C:\testmusic\ With name “MyMelody”	Location : “C:\testmusic\ File Name : MyMelody	The file MyMelody.mid has been saving into the location	Pass

			“C:\testmusic\”	
15	Compose a very fast tempo music with only 3 layer of music	Tempo : 160 Pitch Range : 54 , 78 Layers : 3 Notes Length : 30	A very fast tempo, 3-layered music produced	Pass
16	Plays the music	File Exist	Music played	Pass
17	Edit the music to become even faster	Tempo: 180 Checkbox: Checked	Music’s tempo has become faster	Pass
18	Plays the music	File Exist	Music played	Pass
19	Export Music To “C:\testmusic\” With name “MyfastMelody”	Location : “C:\testmusic\ File Name : MyfastMelody	The file “MyfastMelody.mid” has been saving into the location “C:\testmusic\”	Pass
20	Open the third-party music player	Application exist in \lib	Application fired up.	Pass

Table 6.3.1: Integration testing

CHAPTER 7: FUTURE ENHANCEMENT AND LIMITATION

7.1 SYSTEM LIMITATIONS

The system currently do not have any participation of databases.

The initial intention of purpose this project is to build a music melody generator to assist music composer on music composition works. So there was not necessary that database system should put into the program. Perhaps database should help user to maintain their composed music nicely but it is still not necessary because current version of the system does not provide so much feature to urge the usage of database. The main emphasis of this was still on focus to how the system would produce a new, nice music. Implies that how algorithm functions well, fulfill the composer needs to assist their music composition works.

It is still long ways to go, to tune up a good algorithm.

As the nature of the algorithms, different kinds of problem that are required vary type of the algorithms set up. There is no one absolute solutions that capable to solve any type of problem. The only thing we can do is keep working on improve a algorithm to make it more likely near to perfect, despite that it is nearly impossible to reach the “perfect states”.

The criteria to grade algorithm was not only concern on the time complexity but it is needed to consider the user machine computational power as well. For example like current main stream machine was afford to process particular level of complexity in seconds but on the future time it may capable to reach higher state. That means on future time we may solve a same complexity problem in shorter period, by simply re-calibrate the algorithms.

People in different timing/era might have different kinds of thinking/taste.

This phenomenon is absolutely practical in the field of music. By providing an example a senior citizen may not accept the current genre of music, they may feel the current music consist too many artificial sounds effect and the beats may consider too heavy for them to accept it. On the contrary, the current youth may also not really like the music genre that very popular on past times, they may thinks that is old-fashion and choose not to accept them. **This kind of case has greatly proving a fact that is, as the music there is no absolutely rated as good or absolutely bad but it is just the matters of timing.** A song may only start famous after it has been neglected from a long time ago. This depicted a good song that may not “born” on its own most suitable timing, and it is just yet to getting accepted by others people.

7.2 FUTURE ENHANCEMENT

The Contrapuntal Music Melody generator was initially planned to generate melody by using computer algorithm only. However, for future enhancement, we can try to make the software able to edit the generated melody by human hand. That means we directly open the melody file, make some change on the melody. Besides, we can provide a new module that is let the user remix the music melody, add some more “layer” on the music melody file.

Another potential enhancement on future time is, we can add-in user account module which storing user’s information, and the melodies they generated. Provide them a way to share their “product” on social networking sites.

CHAPTER 8: CONCLUSION

In this 21st century, computer and artificial intelligence plays an important role aiding human performing the tasks which are very tedious and troublesome.

One of the tasks mentioned was the searching and optimization problem. Regarding to the project proposal here was creating the good, new, and original music melodies. There are required a lot of consideration to create a good, new, and original music melody. There may a good melody but similar with existing music, or a new, original music but it is totally unable to accepted by the audiences. The aim of the evolutionary music was creating a brand new, original characteristic, and good to listening music melody.

Since the computer performance has far greater than a human brain, with appropriate AI techniques, the current machine must able to do most of the human job faster than human, including the music composing.

In conclusion, the contrapuntal music melody generator will still have room for improvement. In the future, the task of composing music will not only limit to particular music literacy, may the people who totally know nothing about music theory could also able to become a music composer. By such computer melody generator software emerged, the trend of music composing will totally changed. Just like invention of machine, replacing many traditional jobs which require people does it manually.

REFERENCES AND BIBLIOGRAPHY

Agoston E Eiben & James E Smith, (2003) 'What is an Evolutionary Algorithm?' in *Introduction to Evolutionary Computing*, Springer Verlag
<http://www.cs.vu.nl/~gusz/ecbook/Eiben-Smith-Intro2EC-Ch2.pdf>

Ali Abdolalipour and Ahmad Alibabae 2012, "Harmony Search algorithm". *Int . J. Acad. Res. Appl. Sci* 1.3 : 13-16.

Counterpoint, (n.d.), Available from:
<<http://www.musictheoryhelp.co.uk/comprehensiveguide/1/counterpoint>>. [8 March 2013]

Eduardo R. Miranda , John Al Biles eds. 2007, *Evolutionary Computer Music*, Springer

Engelbrecht, Andries P 2005. *Fundamentals of computational swarm intelligence*. Vol. 1. Chichester: Wiley.

Eric Taylor 1989, *The AB Guide to Music Theory*, The Associated Board Royal School of Music.

Ian Cross 1999, "Is music the most important thing we ever did ? Music, development and evolution." *Music, mind and science: pp 10-39* Available from: Seoul National University Press.

Iterative Application Development. 2012 Available from: <http://www.waterfall-model.com/iterative-software-development/> . [14 March 2013]

References and Bibliography

J. Kim. 2011. *Importance Of Software Development Methodology*. Available From: <http://www.sooperarticles.com/internet-articles/web-development-articles/importance-software-development-methodology-371468.html>. [13 March 2013]

J. Weintraub. 2006. *Implementation of an Iterative and Incremental SDLC (Systems Development Life Cycle) Model Development Project for a Financial Services Organization*. Available From: <http://www.angelfire.com/planet/jweintra/iterativeincrementalprojectdoc.pdf>. [15 March 2013]

K. Kim F. Man, K. S. Tang, S. Kwong 1999. *GENETIC ALGORITHMS: Concepts and Designs, Avec disquette*. Springer Verlag.

M. Rachmadi 2010. *Protootyping*. Available from: <http://rpl-blog.blogspot.com/2010/02/232-prototyping.html>

Perlovsky, Leonid 2011, “*Music. Cognitive Function, Origin, And Evolution Of Musical Emotions.*” Available from: Webmedcentral, ISSN 2046-1690.

Pitch and Clefs, (n.d.), Available from: <http://www.musictheoryhelp.co.uk/fundamentals/2/pitchandclefs>>. [8 March 2013]

Rhythm and Tempo, (n.d.), Available from: <http://www.musictheoryhelp.co.uk/fundamentals/1/rhythmmandtempo>>. [8 March 2013]

Rebecca Morelle, 2012, “*Music evolution: Is this the end of the composer?*” *BBC News* 19 June. Available from: <http://www.bbc.co.uk> . [23 February 2013]

References and Bibliography

S. Dilhan. (2011). *Advantages and disadvantages of Waterfall Model and V Model*. Available from: <http://www.sameeradilhan.com/advantages-and-disadvantages-of-waterfall-model-and-v-model>. [8 March 2013]

Susan Schreibman, Ray Siemens, John Unsworth 2008. *A Companion to Digital Humanities*, ed. Oxford: Blackwell, 2004. Available from: <http://www.digitalhumanities.org/companion/>.

Zong Woo Geem 2010, *State-of-the-Art in the Structure of Harmony Search Algorithm*. Vol 270, pp 1-10, Available from: Springer Link Digital Library. <https://78462f86-a-3a5ac917-s-sites.googlegroups.com/a/hydroteq.com/www/HS_Structure.pdf>.

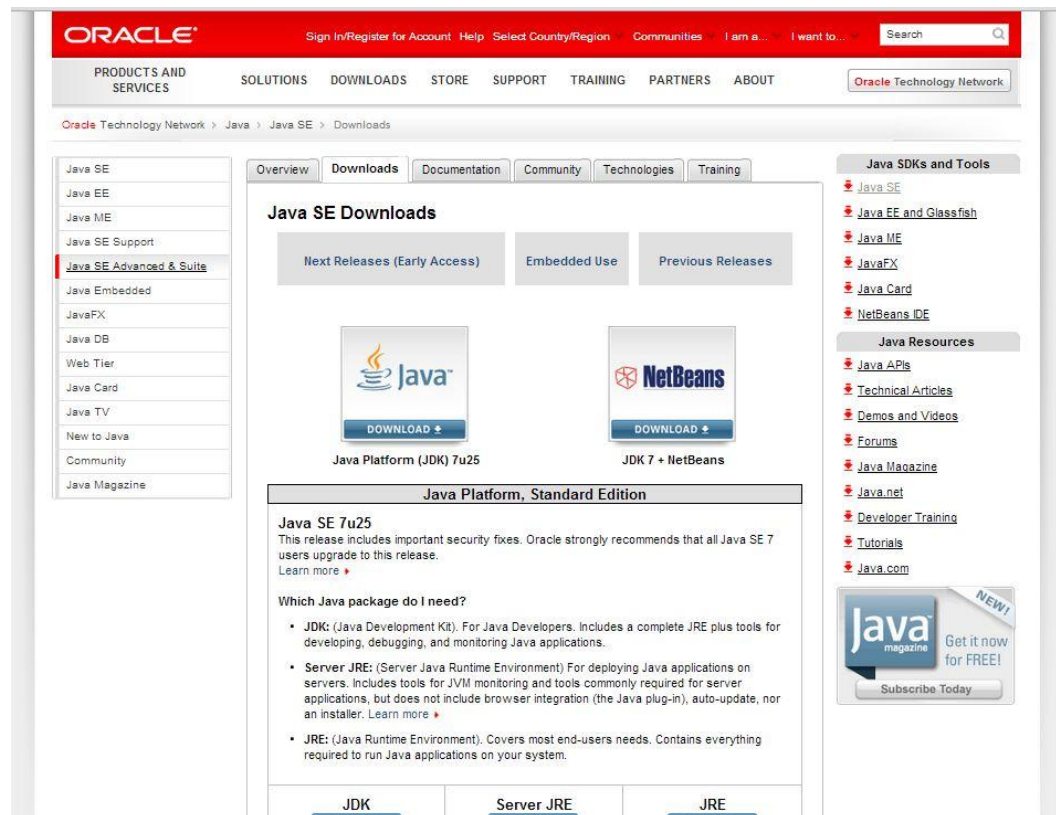
APPENDIX

APPENDIX A: USER MANUAL

1.0 SYSTEM INSTALLATION

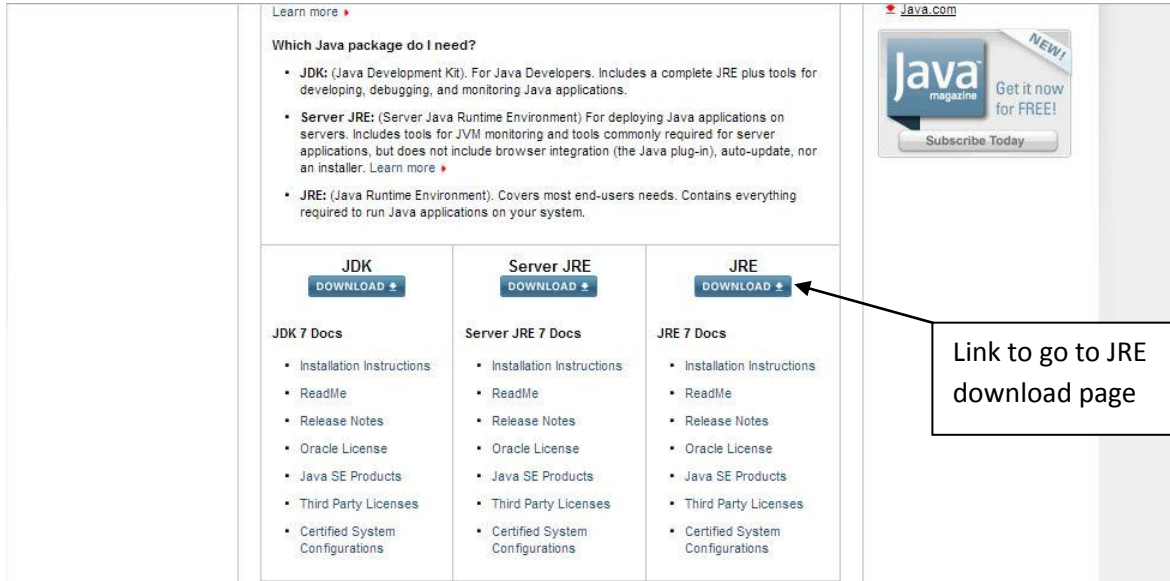
1.0.1 Java Runtime Environment (JRE)

1. The first thing to install before using the system which can be downloaded from “<http://www.oracle.com/technetwork/java/javase/downloads/index.html>”.

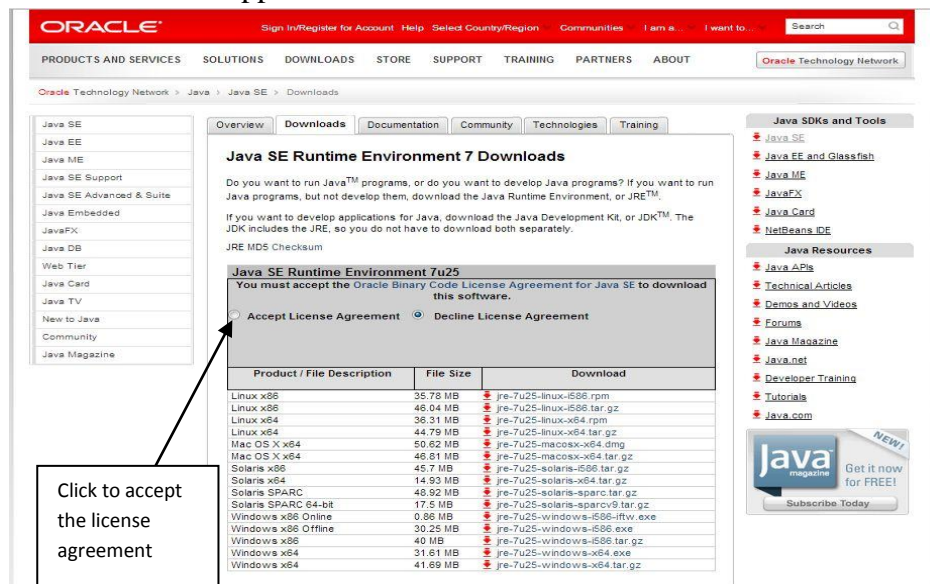


The screenshot shows the Oracle Java SE Downloads page. The page has a red header with the Oracle logo and navigation links. Below the header is a navigation menu with options like PRODUCTS AND SERVICES, SOLUTIONS, DOWNLOADS, STORE, SUPPORT, TRAINING, PARTNERS, and ABOUT. The main content area is titled "Java SE Downloads" and features three tabs: "Next Releases (Early Access)", "Embedded Use", and "Previous Releases". Under "Next Releases (Early Access)", there are two download buttons: "Java Platform (JDK) 7u25" and "JDK 7 + NetBeans". Below these buttons is a section titled "Java Platform, Standard Edition" which includes a "Java SE 7u25" release note and a "Which Java package do I need?" section with three options: JDK, Server JRE, and JRE. The right sidebar contains "Java SDKs and Tools" and "Java Resources" sections with various links.

- After enter the oracle website, scroll it down to find the link which contains JRE installer



- After enter JRE download page, find the radio box which determine whether to accept the license agreement, after press the accept button, the download link for JRE will appear.



Appendix

ORACLE® Sign In/Register for Account Help Select Country/Region Communities I am a... I want to... Search

PRODUCTS AND SERVICES SOLUTIONS DOWNLOADS STORE SUPPORT TRAINING PARTNERS ABOUT Oracle Technology Network

Oracle Technology Network > Java > Java SE > Downloads

Overview Downloads Documentation Community Technologies Training

Java SE Runtime Environment 7 Downloads

Do you want to run Java™ programs, or do you want to develop Java programs? If you want to run Java programs, but not develop them, download the Java Runtime Environment, or JRE™.

If you want to develop applications for Java, download the Java Development Kit, or JDK™. The JDK includes the JRE, so you do not have to download both separately.

JRE MD5 Checksum

Java SE Runtime Environment 7u25

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

Thank you for accepting the Oracle Binary Code License Agreement for Java SE; you may now download this software.

Product / File Description	File Size	Download
Linux x86	35.78 MB	jre-7u25-linux-i586.rpm
Linux x86	46.04 MB	jre-7u25-linux-i586.tar.gz
Linux x64	36.31 MB	jre-7u25-linux-x64.rpm
Linux x64	44.79 MB	jre-7u25-linux-x64.tar.gz
Mac OS X x64	50.62 MB	jre-7u25-macosx-x64.dmg
Mac OS X x64	46.81 MB	jre-7u25-macosx-x64.tar.gz
Solaris x86	45.7 MB	jre-7u25-solaris-i586.tar.gz
Solaris x64	14.93 MB	jre-7u25-solaris-x64.tar.gz
Solaris SPARC	48.92 MB	jre-7u25-solaris-sparc.tar.gz
Solaris SPARC 64-bit	17.5 MB	jre-7u25-solaris-sparcv9.tar.gz
Windows x86 Online	6.62 MB	jre-7u25-windows-i586-iftw.exe
Windows x86 Offline	30.25 MB	jre-7u25-windows-i586.exe
Windows x86	40 MB	jre-7u25-windows-i586.tar.gz
Windows x64	31.61 MB	jre-7u25-windows-x64.exe
Windows x64	41.89 MB	jre-7u25-windows-x64.tar.gz

Choose this if your operating system is 32-bit version

Choose this if your operating system is 64-bit version

Java SDKs and Tools

- Java SE
- Java EE and Glassfish
- Java ME
- JavaFX
- Java Card
- NetBeans IDE

Java Resources

- Java APIs
- Technical Articles
- Demos and Videos
- Forums
- Java Magazine
- Java.net
- Developer Training
- Tutorials
- Java.com

Java magazine Get it now for FREE! Subscribe Today

4. After downloads the JRE installer, double click it to install the JRE. Click next to proceed the JRE installation



- Simply uncheck the “Optional 3rd party Installations” button then click next proceed.



- Wait until the installation progress to be finished. (indicated by the green progress bar)



Appendix

7. After the installation finish, the java runtime environment has been successfully installed into your machine and the contrapuntal generated song is ready to run!



2.0

3.0 GUIDELINE IN USING SYSTEM

3.1 EXECUTE THE MELODY GENERATOR

Double click the `Contrapuntal_Generated_Song.jar` to fire up the melody generator. (Do not remove folder that named “lib” or the generator will fail to work.)

Name	Date modified	Type	Size
lib	8/10/2013 6:01 PM	File folder	
Contrapuntal_Generated_Song.jar	8/11/2013 2:10 PM	Executable Jar File	97 KB
folder lib must put together with the jar f...	8/10/2013 6:04 PM	Text Document	0 KB
README.TXT	8/11/2013 2:10 PM	Text Document	2 KB

Appendix

After the `Contrapuntal_Generated_Song.jar` been executed, the main menu of this program should be come out.



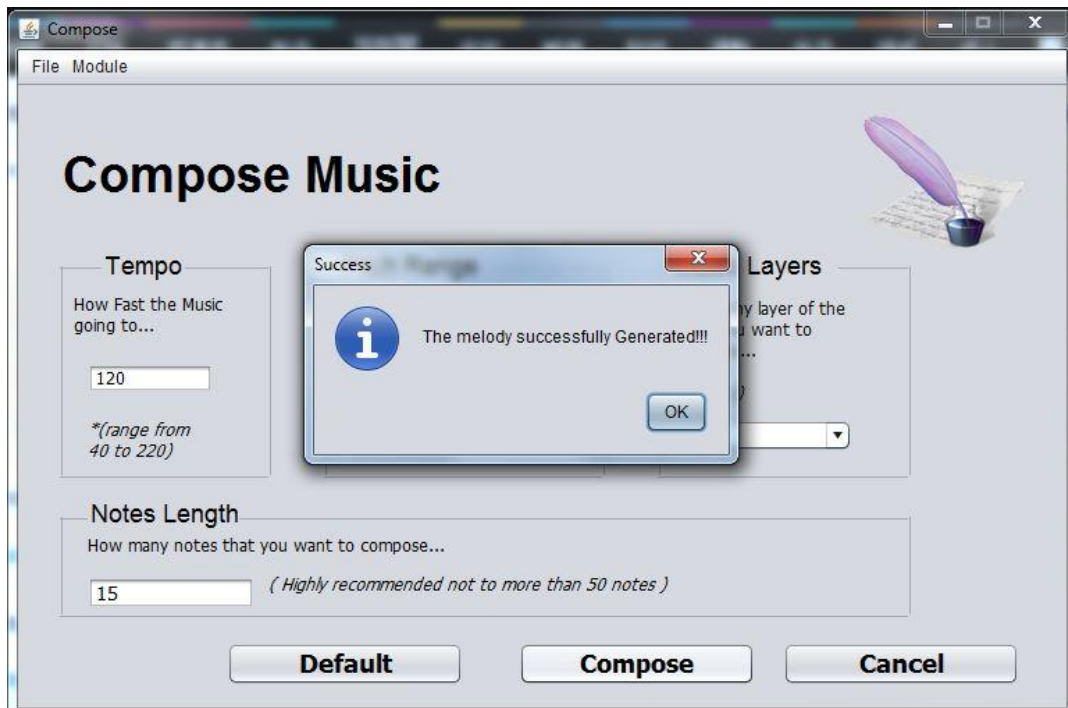
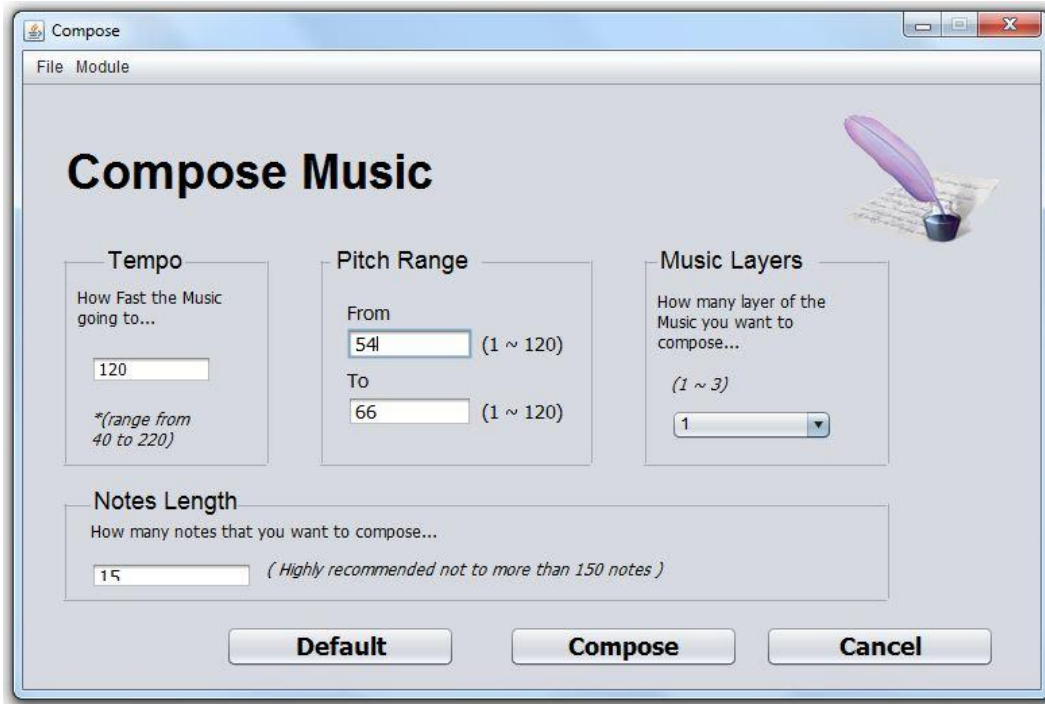
There are six available option appear in the main menu: compose music, preview music, syop preview music, edit music, export music, and Open a third party music player.

3.2 COMPOSE MUSIC

1. Choose the big button that named “Compose Music” in order to enter compose music module.

After the “Compose Music” button click, user should be bring into the compose music interface. Feel free to key in the parameter in the form provided. After that click compose, the program should use up a couple of seconds to compose the music melody. Once the music composition success, the message box will be pop out to inform user. After that the system will bring user back to the main menu.





3.3 PREVIEW MUSIC

To preview the music, simply click the playback music button in the main menu, if the music exist (composed) then the music will be played.



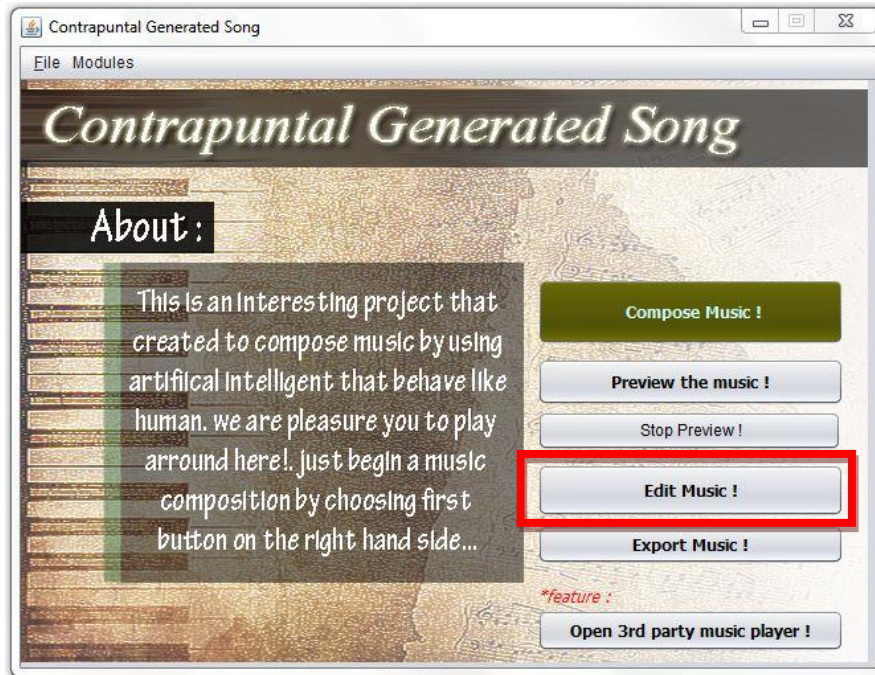


Feel the music this too slow yet boring? Simply click Stop preview button to immediately stop the music playing.

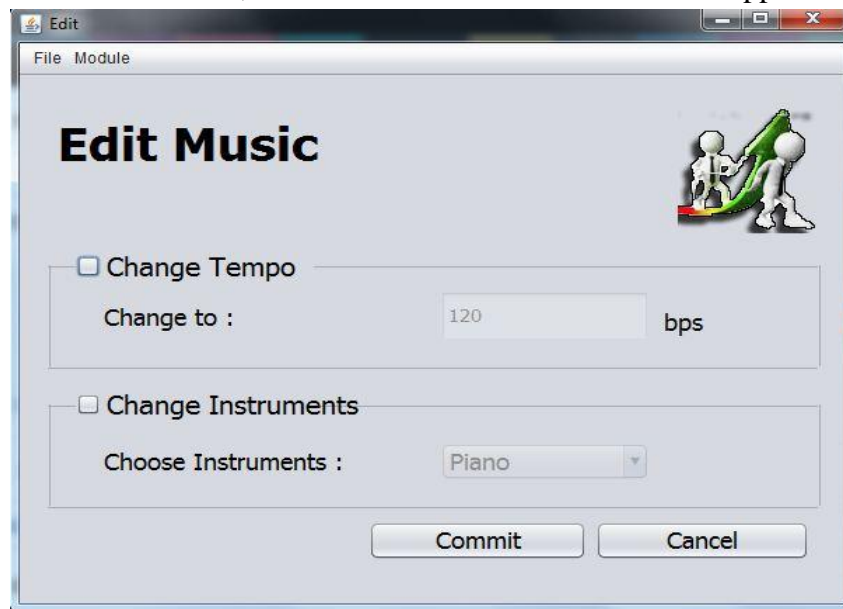


3.4 EDIT MUSIC

To edit the music, simply click the button “Edit music” in the main menu.



After click the button, the interface of edit music should appear in the screen.



Appendix

The application is currently provided tempo and instrument change feature. To perform tempo change on music, simply check the box “Change Tempo” then the form to key in tempo value will be able to be selected.



Edit Music

Change Tempo

Change to : bps

Change Instruments

Choose Instruments :

Commit Cancel

Appendix

To perform instrument change on music, simply check the box “Change Instrument” then the form to choose desired instrument will be able to be selected. This currently supported few type of instrument only.



Appendix

After input the value to the form, simply click the “Commit” button to execute music edits. If success, a message will prompt out to inform user. After it will bring user go back to main menu.



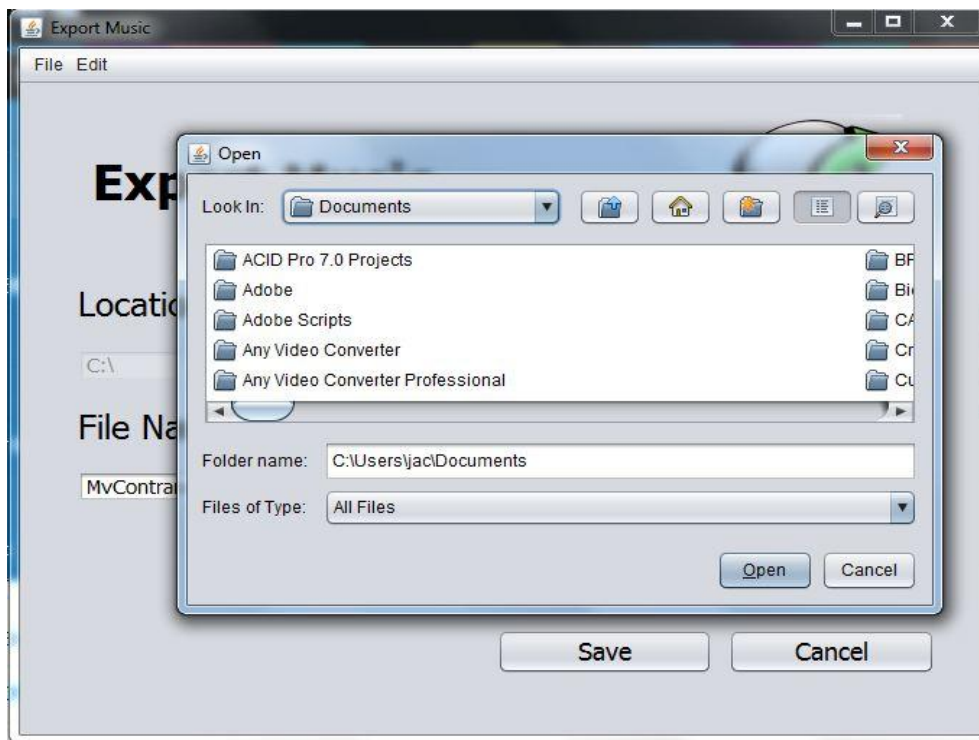
2.5 EXPORT MUSIC

After compose the music and feel it want to export it out, then click the button “Export Music” in main menu to enter export music module.





Click the browse to choose the location that you want to save your music.



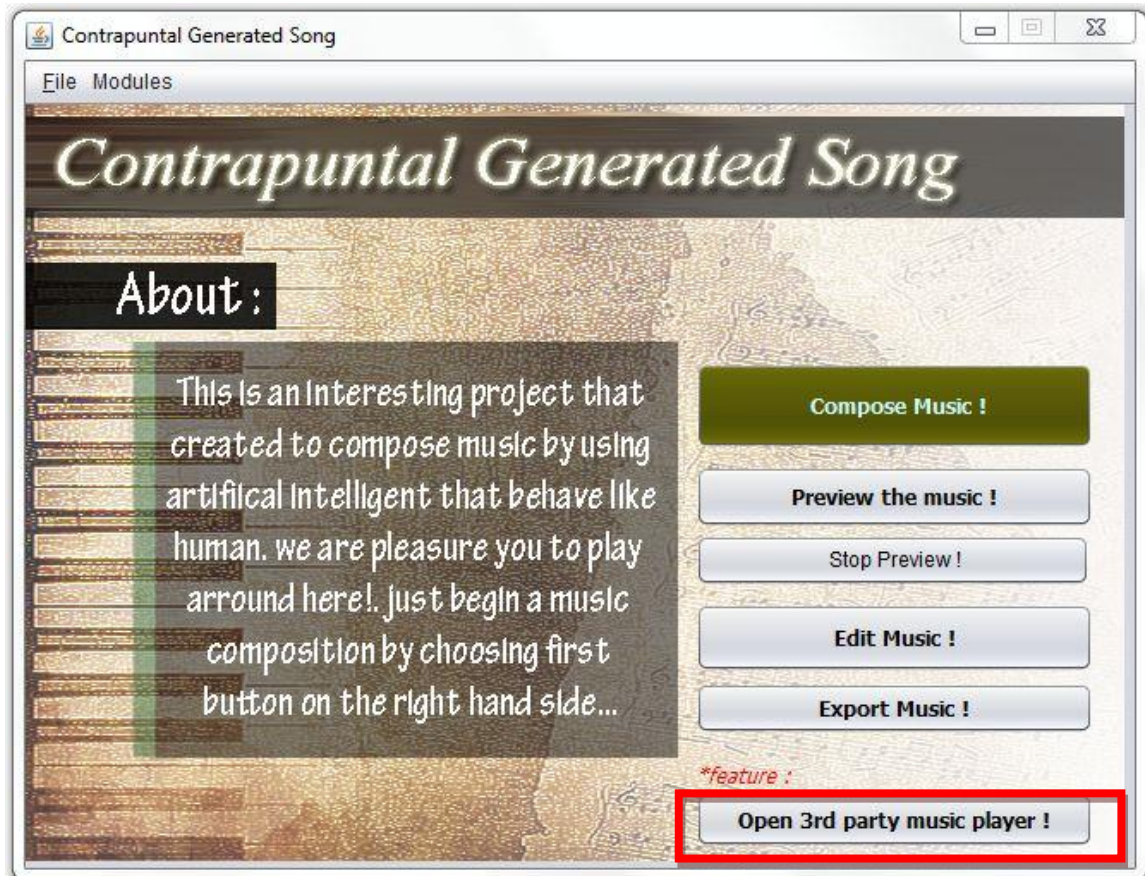
Appendix

After choose the path and file name, simply click save to export out your melody to the location that you chosen. A message box should prompt out inform whether the music exports successful or not.



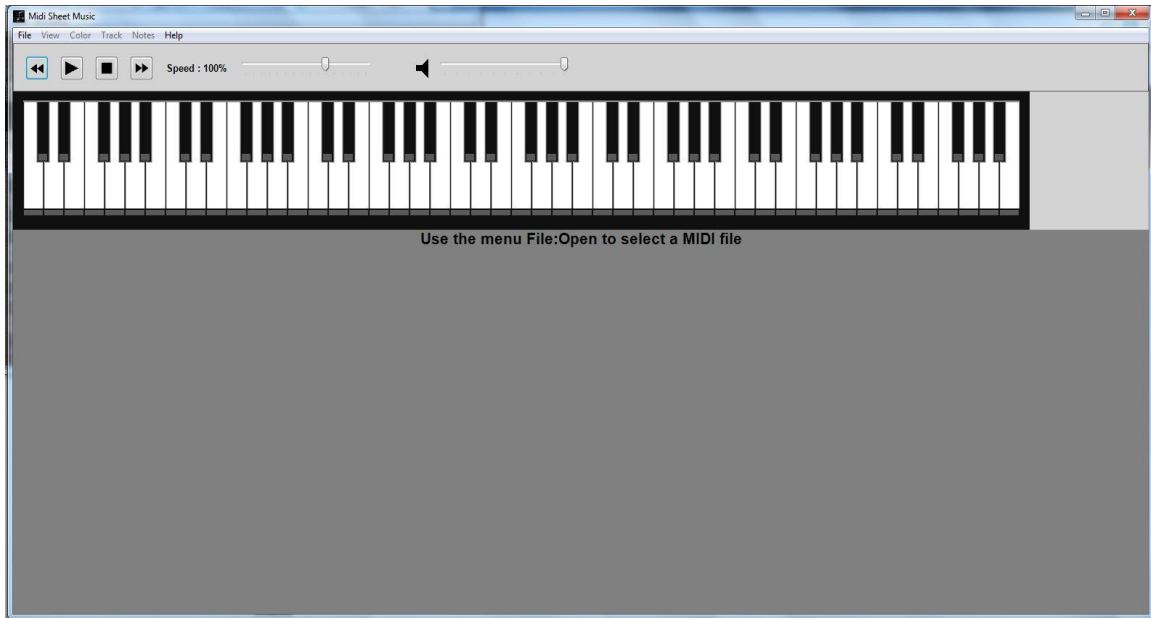
2.5 PLAYS MUSIC WITH READING MUSIC SHEET

This application has included an third party application that let user to listen the music reading the music score. To open the third-party application, simply click the button “Open 3rd party music player”.

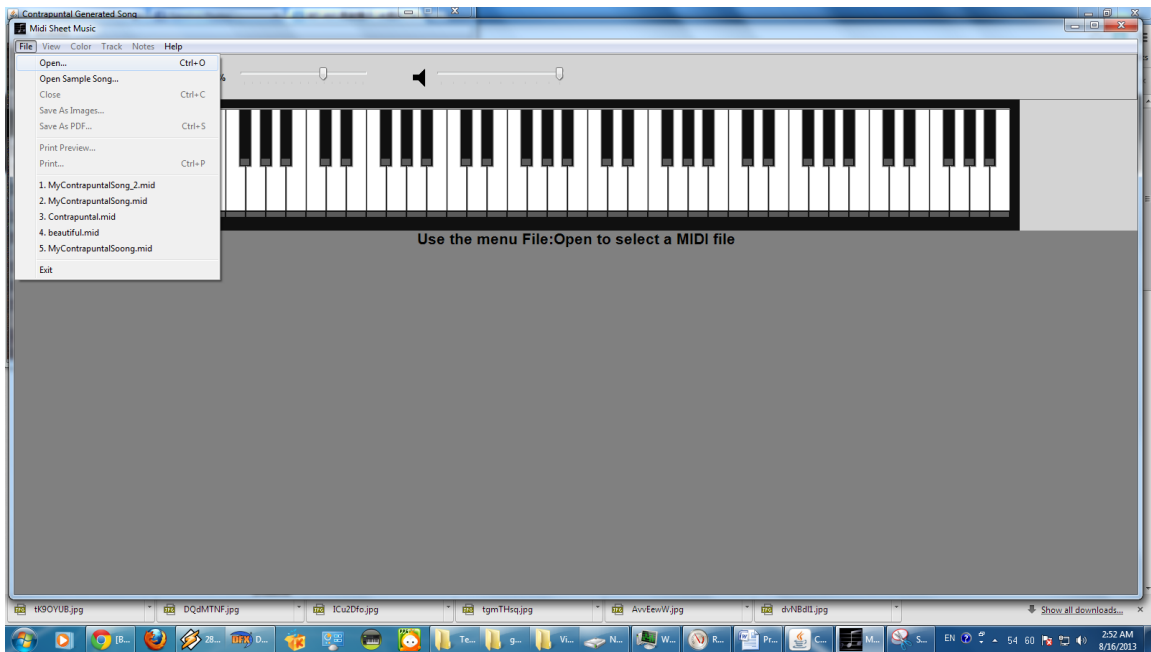


After user clicked the button “Open 3rd party music player”, a new application window should be appearing as below.

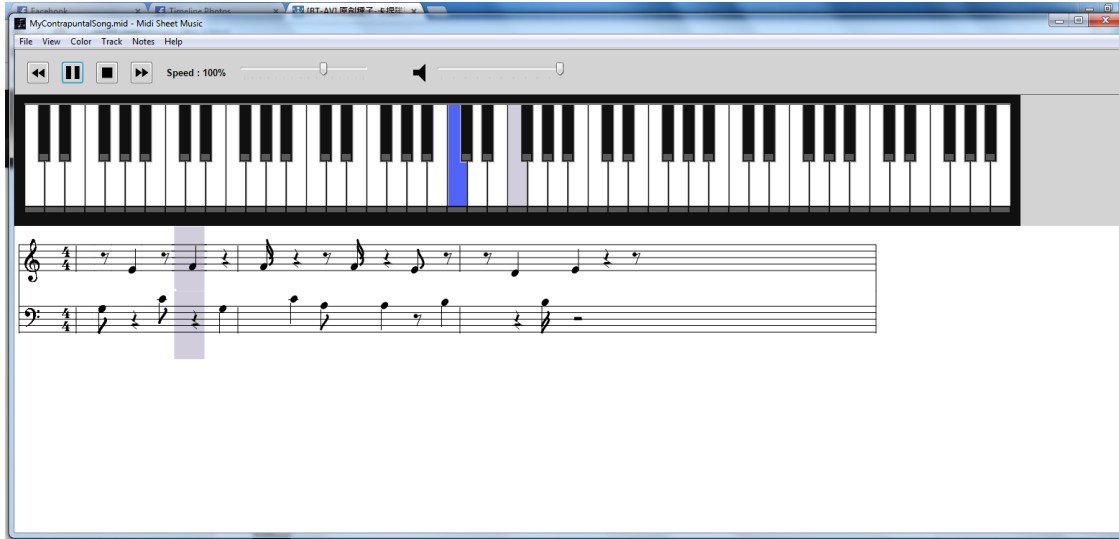
Appendix



To play music by using this application, go File > Open > select the music file then the music file should imported into the application.



Appendix



The music will be played with the score should highlighted the music playing progress accordingly.