CSC3002 Introduction to Computer Science: Programming Paradigms
Prof. HUANG, Rui

EMOsician Software

By Kaicheng Zhou, Tianling Wang, Jie Ji,

Ruihan Wu, Jin Rao, Lanxuan Deng

Executive Summary

Software Updates From Proposal:

1. Carefully Designed Parameters: Style
2. Re-generating Function

Module Updates From Proposal:

1. Clearer User Interface Design & Advanced Music Player
2. More Delicately Designed Rules for Melody Making
3. 'HIFI' Music Generating Method

Project Highlights:

1. **Unique Idea:**

EMOsician idea is in essence a principle that some part of music language can be interpreted as human sensation and computer can be the intermediate to link these two. There is no similar project existing currently.

2. **Creative Approach:**

we design our own software architecture and datatype to decomposing large problem(composing 3-min song) into same form of sub-problems(composing several notes).

3. **Valuable Application:**

This software is not only interesting in idea but also practical in market. There are three potential users (Child, Adult, Professionals) that will be interested and benefited by EMOsician

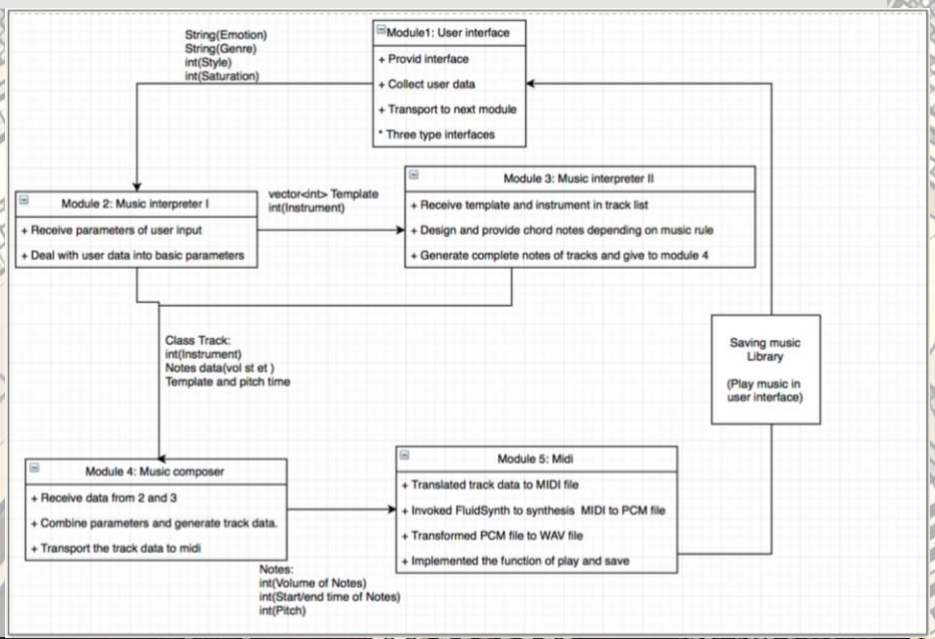
4. **Challenging Project:**

We design complex data structure and datatype to translate user input and music parameters. Moreover, trackdata is written. We also have to deal with hex, binary coding and I/O.

Sample Output:

https://pan.baidu.com/s/1qg9-p7ESS_5PsWXhipNhFg

Warning: Since FluidSynth has system dependency, it may encounter problems when running in some systems.



EMOsician Software

Contents

1.User Guide

Step 1:Choose Parameter

Step 2:Generate and Play New Song

Step 3:Re-generate New Song

Step 4:Save Your Song

1.1 Updates From Proposal

Carefully Designed Parameters

Re-generating Function

1.2 Sample Output

2.Project Highlight

Unique Idea

Creative Approach

Valuable Application

Challenging Project

3.Team Member and Task Distribution

4.Final Project Structure

General Structure

Module 1: User Interface

Module 2: Music Parameter Interpreter

Module 3: Melody Composer

Module 4: TrackData Generator

Module 5: Song Generator

4.1 Updates From Proposal

Clearer User Interface Design & Advanced Music Player

More Delicately Designed Rules for Melody Making

‘HIFI’ Music Generating Method

5.Challenge, Solution&Gains

Implementation of UI functions

Track Class Design

MIDI File Writing

Compilation of Midi Synthesizer

Understanding of Different Music Formats

File Input/Output

6. Expectations For CSC3002

7. References

EMOsician Software

1.User Guide:

Welcome to be an EMOsician! You can also be a musician with your emotion, music preference and even professional music understanding.

EMOsician will provide you with three versions to compose and generate music.

Step 1: Choose Parameter:

1.Simple Version:

If you are just new users, this version is the simplest one and can let you get familiar with EMOsician. You can choose your preferred emotion including sad, happy, excited, nervous, lonely, relaxed and blue, which will be the basic tone for your EMOsician song.

2.Advanced Version:

If you are advanced users, you further specify your preferences including choosing your preferred genre, saturation(the density of instruments), and style (the complexity and speed of rhythm).

3.Professional Version:

If you are professional user who wants to DIY your music, professional version is best fit. You can choose your preferred Tempo (BPM for the song), Melody(your root note progression), Lead Instrument, Template(Music Composition Template), Dynamic Plus(Volume Fluctuation).

Step 2: Generate and Play New Song:

After choosing parameter, EMOsician will translate your input into self-created music parameters. After careful calculating, you can click ‘generate new song’ and get a brand new original songs, which is **wholly composed by our software**. When the ‘play’ button is working, you can click it and listen to your song in the music player board.

Step 3: Re-generate New Song:

Music is a journey of trial and error. A good song takes time to write and choose. If the music generated is not satisfying, you can just click generate new music and get another new piece of music. Your old music file will be replaced but your trackdata will be kept, which is more efficient and also creating a back-up.

Step 4: Save Your Song:

If you like this song, you can save it by clicking ‘save’ button.

Hope you can enjoy it!

1.1 Updates From Proposal:

For EMOsician Software Function, we have done several updates:

1. Carefully Designed Parameters:

Initially style is too close to genre, which makes this parameter unclear. We redefined genre as complexity and speed of rhythm, which makes this parameter more distinctive.

2. Re-generating Function:

Initially, we haven't included re-generating function in EMOsician. However, since rewriting is one of the most important thing in music composing, we added this function and carefully design the data flow behind it. We will remain track data as back-up for old songs. However, once the user choose to re-generate, the pcm and wav file of that song will be covered.

1.2 Sample Output:

Here we provides a link to our sample output:
https://pan.baidu.com/s/1qg9-p7ESS_5PsWXhipNhFg

2.Project Highlight:

1. Unique Idea:

EMOsician is, at first place, interesting in its uniqueness of the idea. There is currently no such program can take users's emotion wholly based on programming to compose, generate and play music. There are only some programs generating MIDI and do not take users's emotion. Therefore, EMOsician is actually the first music-emotion software. This uniqueness of idea is in essence a principle that some part of music language can be interpreted as human sensation and computer can be the intermediate to link these two.

2. Creative Approach:

EMOsician is creative in its huge quasi-recursion approach to achieve its vision. EMOsician's uniqueness leads to a problem that there is no sample project. Therefore, we have to design our own software architecture and datatype to compose music. Since, composing a 3-min song is a complex project, we borrow the idea from recursion—decomposing large problem into same form of sub-problems. Using templates, we decompose writing the whole song into writing piece of melody for each component. Using chord progression, we decompose writing for component into writing for a single section. Using melody generating rule, we decompose writing for single section into writing for several notes.

3.Valuable Application:

This software is not only interesting in idea but also practical in market. There are three potential user types that will be interested and get benefits in EMOsician. 1) For children, this software can help them get an early-stage understanding of the relationship between music and their sensation. Thereby, help them gain interests and intuition in music composition. 2) For adults, DIY music for their emotion is a fun way to relax and they will get the feeling of a private song composer. 3) For professional or amateur musician, this software will quicken their music composing process. They can just press one button and get random generated music, which helps them find the right riff and melody faster.

4.Challenging Project:

EMOsician is also challenging not only in writing over 6000 lines in short time but also in its range of areas. We have to design complex data structure and datatype to translate user input and music parameters. Moreover, trackdata is written. We also have to deal with hex, binary coding and I/O.

3.Team Member and Task Distribution:

Ruihan Wu(115020139):

Design and code Module 1 User Interface (three versions).

Lanxuan Deng(117010048),Kaicheng Zhou(115020314), Jin Rao(117010221):

Co-work on Module 2&3&4. Developing Input Interpreter I, Music composer, and Melody Maker

Jie Ji(117010104):

Develop Module 5.1 Music Generator — MIDI generator + Music Composer

Tianling Wang(115020129):

Develop Module 5.2 Music Generator — pcm to wav convertor + Music Player

4.Final Project Structure:

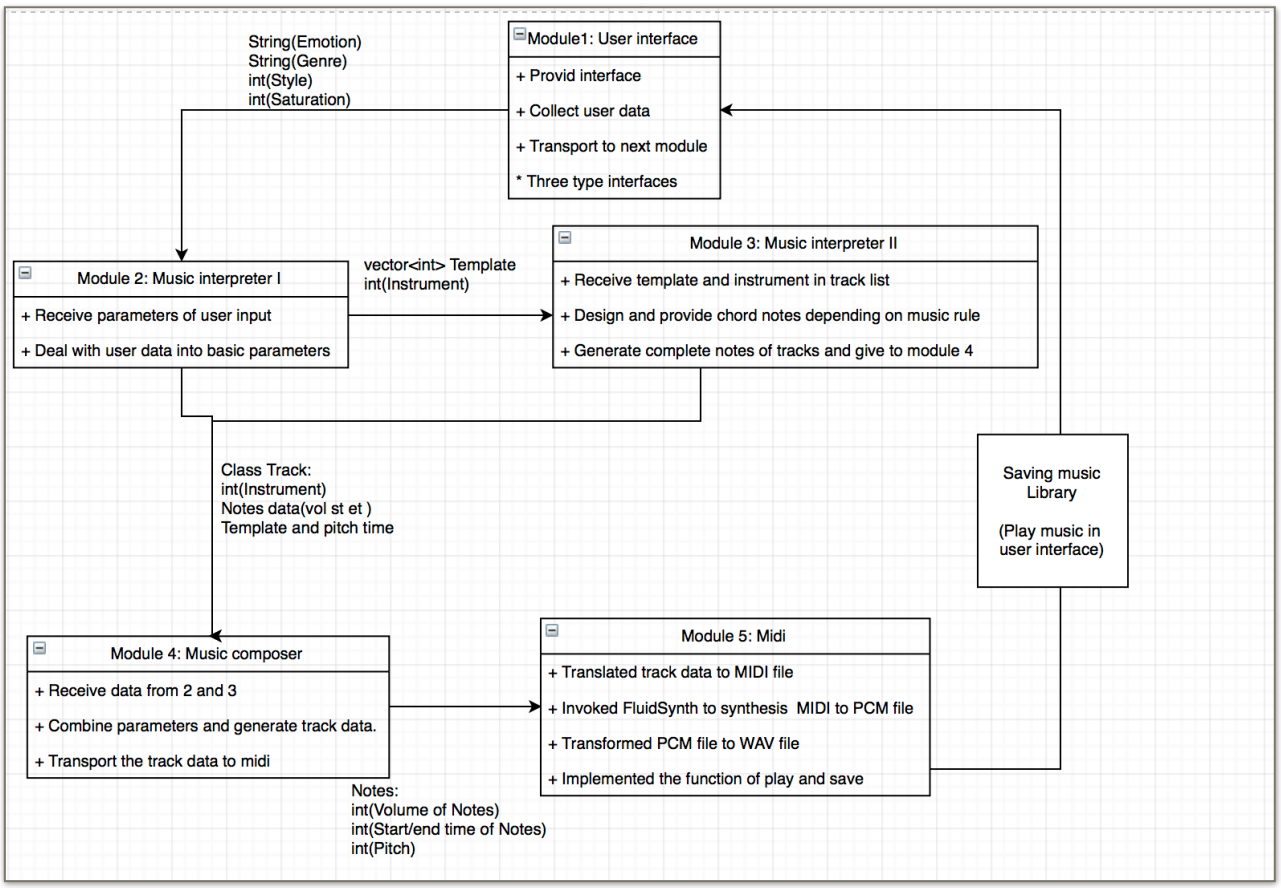
The whole structure of our project is depicted in the UML picture in the next page.

General Structure:

We have set five modules to take user input and output wav file.

Module 1 UI is interacting with users to get user input parameters and implement music player functions through clicking.

Module 2 Music Parameter Interpreter is to take user input parameter and return important music composing parameters including template, tempo, instruments set and volume curve.



Module 3 Melody Composer is to take template, emotion, genre, style and return the core part of song, accompany and lead melody.

Module 4 TrackData Generator is to take all note data from module 2&3 and return in class trackdata, write text file to save all composing history data.

Module 5 Song Generator is to take trackdata, transform it into MIDI, using fluidsyth to compose it into pcm file. After this, it will convert pcm file into wav file and interact with UI to play, save the music.

Module 1: User Interface:

The user interface is implemented by Qt designer, a Qt tool for designing and building graphical user interfaces (GUIs) with Qt Widgets. We make changes from proposal because using user interface with Qt designer is more convenient and efficient where users can customize windows or dialogs in a what-you-see-is-what-you-get (WYSIWYG) manner.

There are four windows displayed including main window, standard dialog, advanced dialog and professional dialog. main window is the basic main user interface which will be shown at the very beginning when someone uses this application. Buttons controlling selections of different versions and basic music player functions like playing, pausing, saving, volume and progress bar are displayed on it. Moreover, users can also open local wav files from the computer through the

“File” button in the menu bar on the top of the main window interface. Standard dialog, advanced dialog and professional dialog are windows for different versions which will appear after users click the corresponding version buttons. In standard dialog, there is only emotion type to be chosen. In advanced dialog, apart from emotion types, different genre types, style types and saturation level are also provided as choices. Because the professional version is designed for people more proficient in music theory, users do not choose emotion types on professional dialog, instead, they choose template types, tempo types, instrument types and dynamic-plus types to generate music.

Module 2: Music Parameter Interpreter

To transform the music basic parameters, this module mainly uses mapping function and random function to provide mapping rules, switch functions to consider different situations and provide suitable functions. The random function will automatically generate random music parameters such as template, tempo, rhythm and instrument within a setting range. The volume remark function is changed by time in sinusoidal function to provide randomly seed for random function.

Module 3: Melody Composer

This module takes input from UI and Module including two kinds of case. Case 1: directly taking root note progression from professional version. This module will directly write chord progression based on it. Case 2: taking input from UI and Module 2 including template, instruments, genre, emotion and style to write melody. Firstly, genre will determine a certain domain of root note progression. By random choosing one root note progression, emotion will determine the form of chord based on that root note. With chord progression, different instruments will decide different ways to play accompany parts. For lead melody, chord progression will determine the applicable domain for lead melody and lead melody will be decided following certain rules.

Module 4: TrackData Generator

Considering that the data is transported inside the project, we designed the *track* class. The *track* class can easily collect the data of the notes (pitches), and can be easily traversed with a serial number *index*. We constructed the *track* class to transport data between modules. We created the track and preset the instrument into the track list, and then module 3 will call for instrument and give notes data into track, and lastly, module 5 will read the notes data through get-function.

Module 5: Song Generator

This Module has been recomposed to 4 sections: track to midi, midi to pcm, pcm to wav, wav file to music player(UI).

Module 5.1:

This module is to receives track data from upstream, interpret it to standard GM1 MIDI file, synthesize it into PCM audio file, and pass it along the line.

We made two passes on tracks, created a time-indexed cache vector at the first pass, and encoded the vector into legit MIDI file on the second pass. Then, we called FluidSynth and utilized external .sf2 sound bank to provide better sound quality.

Module 5.2:

For the pcm-to-wav part, we fully analyzed the data structure of wav file and structured the inner classes to create a new wav file, then wrote the wav head and body according to the parameters in the pcm. Afterwards, we adjusted the parameters based on our needs, for example, setting the channel number to achieve the double-channel stereo audio effect.

For the music player part, we need to realize two main functions which are Play and Save. There are multiple ways to play a wav file, among which we choose the one that comes with Qt, so it can be better integrated with the GUI. Users have two ways to play a wav. For one, by directly clicking the Play button, it will read the absolute path where the new piece of music was generated and play it. For another, by clicking the File button in the upper left corner, user can choose any wav file from the disk and play it using our player. As for the Save function, the realization relies on the QFile, QDir, QString, QFileDialog classes in Qt to identify the original file, read the content and copy it to a new file, read the new address the user selects, and finally save the copied file.

4.1 Updates From Proposal:

1. Clearer User Interface Design & Advanced Music Player

Instead of putting all the selection buttons of different versions on the main window, separate dialogs are utilized for different versions to make the interface more clearly and ordered,.

Initially, there is no time recording, volume control bar and open-local-file function on the main window. But to make the user interface more user-friendly and realize all the basic functions that ordinary music players have, these functions are added to the final user interface.

2. More Delicately Designed Rules for Melody Making

Initially, we planned to write the lead melody first and write accompany melody later. After discussion, we found writing accompany first is a better way to decompose complex tasks.

3. ‘HIFI’ Music Generating Method:

We created the intermediate music format pcm, because we found that it’s quite a mess when converting a MIDI music file directly to wav, but better to implement through a pcm file since it has

better quality and fits for several HIFI players. Besides, a txt file trackdata.txt is also provided to save all songs the program has written.

5. Challenges, Solutions & Gains:

Implementation of UI functions

The synchronization of the slider and spin-box when implementing some functions like choosing the saturation level and the tempo level is kind of difficult. This problem was solved by searching relative solutions on the internet. I learned how to implement user interface by Qt designer, how to find solutions through internet resources.

Track Class Design

Designing the track class to connect three modules is hard to achieve. We learned the way that using index to traverse the notes data from the internet and successfully achieved it in our project.

MIDI File Writing

MIDI file has to be written in raw binary format, leading to difficulties in debugging: we have to read MIDI output in binary in any debugging attempt. This challenge becomes more difficult because our project is trying to structure musical parameters in a self-defined class called trackdata. To translate this trackdata with 10 channels into MIDI standard file, we have to design mapping functions and the binary transformation functions. By solving this problem, I get familiarized with binary details of MIDI files.

Compilation of Midi Synthesizer

The compiling for FluidSynth needs a mountain of dependencies, which leads to much pain in the process of compiling, since the developers didn’t provide binary release for win32 platform. Writing in raw into a file is also troublesome. I got familiarized with compiling binaries with dependencies on Windows.

Understanding of Different Music Formats

We have to identify which music format is suitable for notes generation and the conversion capabilities between different formats. Therefore, we researched in the structures of midi, pcm, wav to identify their inner relations for conversions. For example, a wav file includes a RIFF chunk descriptor which identifies the file format as WAVE, a fmt sub-chunk which describes the format of the sound information in the data sub-chunk, and a data sub-chunk which indicates the size of the sound information and contains the raw sound data.

File Input/Output

The I/O methods supported by Windows system are not compatible with Mac system. In order to solve this, we abandoned the system-specific coding methods and turned to the more general I/O methods developed by Qt Creator.

6. Expectation For CSC3002:

1. Advanced string and stream operations, such as padding to a certain width, show/translate into string as Dec/Hex/Bin
2. More content about GUI design
3. Deeper study on program paradigm and the methods behind them
4. More content about the utilization of IDE and the packages provided

7. References:

1. The Qt Company Ltd. (n.d.). Qt Documentation. Retrieved from: https://doc.qt.io/qt-5/qtdesigner-manual.html
2. Zhi Eng, Lee (2018), Hands-On GUI Programming with C++ and Qt5. Birmingharm, UK: Packt Publishing Ltd.
3. ProgrammingKnowledge (2013). QT C++ GUI Tutorial 5- How to open a new window from a pushbutton in Qt. Retrieved from: https://www.youtube.com/watch?v=tP70B-pdTH0
4. boywgw. (n.d.). PCM转WAV. Retrieved from CSDN: https://blog.csdn.net/boywgw/article/details/49099473
5. mcgrady_tracy. (n.d.). wav音频文件格式解析. Retrieved from CSDN: https://blog.csdn.net/mcgrady_tracy/article/details/52502263
6. ymc0329. (n.d.). Qt拷贝文件、文件夹. Retrieved from CSDN: https://blog.csdn.net/ymc0329/article/details/7975654/
7. FluidSynth(2019). Retrieved from https://github.com/FluidSynth/fluidsynth