MIDI Conversion to Musical Notation

Rikip Ginanjar
Faculty of Computing
President University
Bekasi, Indonesia
rikipg@yahoo.com

Ivan Iskandar
Faculty of Computing
President University
Bekasi, Indonesia
ivanisk87@gmail.com

Abstract— **Nowadays, music has become an integral part of human life. People live with music surrounding them. Almost everywhere in the world there is music playing. Playing music is no longer the work of musician only, as all humans like music. Musical Instrument Digital Interface (MIDI) was created to help people play numerous musical instruments alone and to help musicians enrich their music. As the development of music, people can only rely on hearing to know how to play/sing a song. However, not everybody is a capable singer. If the singer sing a wrong pitch, everybody learning from him will be false as well. To solve this problem, musical notation is needed. It is the same with MIDI. People cannot reproduce a song in MIDI if they do not know the chords to play it. Thus, musical notation is the solution. Converting MIDI file into musical notation is the goal of this project. The musical notation produced will help people who want to learn a certain song contained in a MIDI file which can be found across the internet for free or for a small fee.**

Keywords-component; *MIDI Conversion, musical notation*

I. INTRODUCTION

As music becoming an inseparable part of human life, music can be found anywhere. Playing music is not only for musicians, but also for all humans alike. With the development of music, a standardized form is needed to prevent any differences in pitch and tempo. The solution is by implementing musical notation. In the past, musical notation could not serve as a learning sheet, since no exact pitch was agreed. However, as music progresses, the musical notation become more complicated as to show exact pitch everywhere in the world. With a musical notation, even if someone does not know a particular song, they can read at it and know how to play it. Song is intangible, people cannot touch it, cannot read it, people can only hear it, feel it. Musical notation is the tangible form of song, letting people to know a song without hearing it.

Musical notation is a popular representation of music in a standardized form. In a musical notation, a note is defined as a note relative to its clef and how a musical instrument is played. A “C” in treble clef will have a different sound than a “C” in bass clef.

MIDI is popular because unlike other digital audio files (wav, mp3, etc) MIDI does not need to capture and store actual sounds. MIDI can be just a list of events that describe specific steps that a soundcard or other playback device must take to generate certain sound. This way, MIDI files are much smaller than other digital audio files.

Musicians have treated MIDI as their helper whenever they are short on player. When there is not enough people to perform a music, they can use MIDI to cover the hole left by those unavailable players, either in a music performance, or in sound recording. MIDI can also be used as a source of inspiration for musicians.

However, when they hear a MIDI file playing and they want to play it on their own, they might be hindered as they do not know how to play the song because they do not know the chords to play it.

The solution is to convert MIDI file into musical notation, so it is playable and producible. Knowing the musical notation hidden in MIDI file can greatly improve a musician’s skill and knowledge.

This paper intends to convert a Musical Instrument Digital Interface (MIDI) file to the form of musical notation. A MIDI file contains music data in performance instructions, making it possible to translate it into a complete musical notation with staves, clefs, tempo, and notes.

II. SYSTEM ANALYSIS

The MIDI Converter is about taking in a MIDI file, reading it by bytes and interpreting the information inside, and then converting it into a readable musical score. Every MIDI file follows a format that has been discussed in chapter 2, so it is possible to make an application to access all pertaining information inside. Accessing MIDI file will be done by reading the file word per word to find the pattern. The data read will be in object file so it is passed by reference. After the data is read, it will be sent immediately to be processed by another method and the graphical representation will be created.

Figure 2.1 clearly shows the flow that the user might do with the application. The user will start by giving an input to the application regarding the location of the MIDI file. This can be done in two ways: using open dialog box, where the user can browse to the location of the file and choose it; and by entering the file path manually to the provided text box. After the file path is entered, the user must check the file validity. If the path provided refers to a MIDI type 2, a non-MIDI file, or a non-existent file, an error dialog box will appear to notify the user of this failure. If a correct path has been confirmed, the user will be enabled of running the converting process. The application will then run the process of converting MIDI and give result to the user. From the Use Case Diagram, an Activity Diagram is then derived.

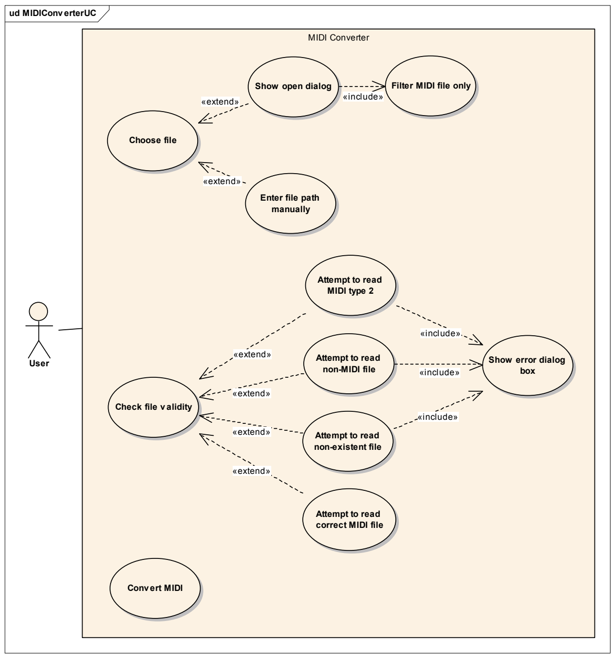

Figure 2. 1 MIDI Converter Use Case Diagram

The Activity Diagram in figure 2.2 shows the flow of the application. After the user chooses a MIDI file, the application will read the header of the file and check whether it is a type 0 or 1 MIDI file. If yes, the application will start reading the events contained in the file until it reached the end of file. After reaching the end of file, the application will use the data to make a score and show it to the user.

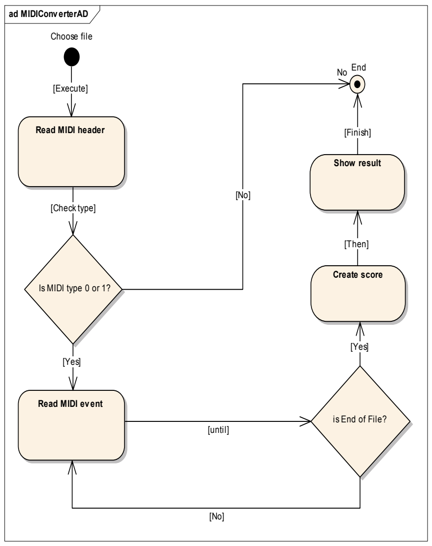

Figure 2. 2 MIDI Converter Activity Diagram

After seeing the Activity Diagram, the next part is conveying the information into coding part. The Package Diagram in figure 2.3 will show a glimpse on what the coding will cover.

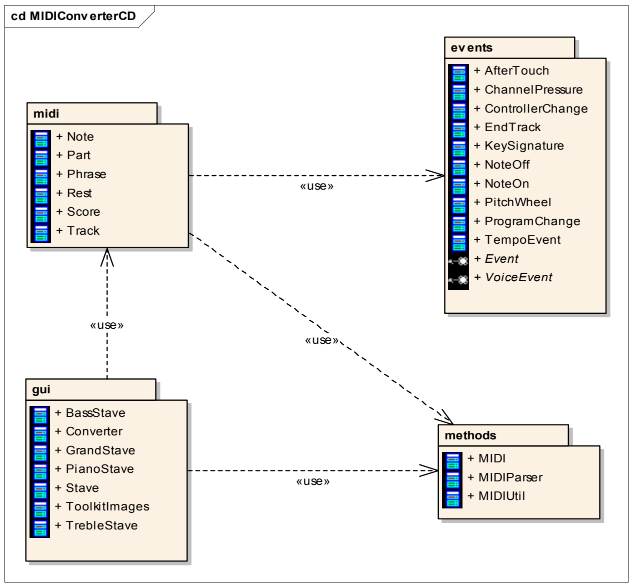

Figure 2. 3 MIDI Converter Package Diagram

III. SOFTWARE IMPLEMENTATION

A. User Interface

The MIDI Converter is about taking in a MIDI file, reading it by bytes and interpreting the information inside as shown in figure 3.1.

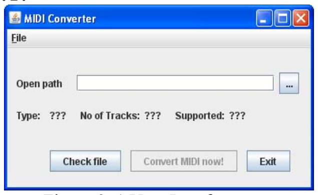

Figure 3. 1 User Interface

First of all, when the user clicks the browse button, the file filter will be set to show the only files with ".mid" extension. Everything else besides a folder and MIDI file will not be shown. After the user selects a file, the text box provided in the main form will be filled with the file's absolute path. Yhe codes run are stated in Figure 3.1.

```
private void btnOpenActionPerformed(ActionEvent evt) {
FileFilter filter = new FileNameExtensionFilter("MIDI
file", "mid");
fileChooser.setDialogType(JFileChooser.OPEN_DIALOG);
fileChooser.setFileFilter(filter);
fileChooser.showOpenDialog(null);
txtOpen.setText(fileChooser.getSelectedFile().getPath())
;}
```

Figure 3. 2 the converter.btnOpenActionPerformed code

After that, the user should click the Check file button that will retrieve the file path and read the data inside. The data will be checked to see whether it is a type 2 MIDI file. If it is not, the check method will state that the file is supported.

Read method will read the file header to check if it is truly a MIDI file by checking the header. If it is

"0x4D546864" then it is truly a MIDI file. The button check will run read method as in figure 3.3:

```
public void read(InputStream is) throws IOException
{
final byte[] fileData = new byte[is.available()];
is.read(fileData);
final ByteArrayInputStream bais = new
ByteArrayInputStream(fileData);
final DataInputStream dis = new
DataInputStream(bais);
// clear any SMF data
if (!trackList.isEmpty()) {
trackList.removeAllElements();}
// check header for MIDI file validity
if (dis.readInt() != 0x4D546864) {
JOptionPane.showMessageDialog(null, "This is NOT a
MIDI file!", "Error", JOptionPane.ERROR_MESSAGE);
throw new IOException("This is NOT a MIDI file!");
} else { dis.readInt();}
try {fileType = dis.readShort();
numOfTracks = dis.readShort();
ppqn = dis.readShort();
} catch (final IOException e) {
e.printStackTrace();}
for (int i = 0; i < numOfTracks; i++) {
readTrackChunk(dis); }
is.close();
dis.close();}
```

Figure 3. 3 MIDI.read

If it is a MIDI file, it will continue to readTrackChunk method. ReadTrackChunk method will start by checking whether the track starts in the right place by checking the next offset. If it is "0x4D54726B", it will continue to read the deltaTime. The stream will then be set to this position so if the status is smaller than 0x80, it will be resetted. Subsequently, it will check whether it is a meta event, system exclusive (which is not supported), or voice event.

After the file has been concluded as a valid MIDI file, the Convert MIDI now button will be enabled.

Afterward, it will continue by checking and drawing the time signature and pitches. The pitch will again be checked for its individual accidentals, sharp or flat. Since there are 7 natural pitch, there are 5 accidentals in total. C, C#, D, D#, E, F, F#, G, G#, A, A#, B (or in flat). For example, C4 = 60, CS4 = 61, DF4 = 61, D4 = 62, DS4 = 63, EF4 = 63, E4 = 64, ES4 = 65, FF4 = 64, F4 = 65, FS4 = 66, GF4 = 66, G4 = 67, GS4 = 68, AF4 = 68, A4 = 69, AS4 = 70, BF4 = 70, B4 = 71. The pitch number will be divided 12 and checked if the remainder is 1, 3, 6, 8, 10. Those are the numbers that is a sharp or a flat, while the rest are natural pitch.

The method will next continue the painting of the score by drawing lines and putting appropriate images according to a particular pitch. It will continue to check the height of the stave and see if it needs additional line outside the standard ledge, be it above it or below it.

IV. Conclusion

Since music has become an inseparable part of human life, Musical Instrument Digital Interface (MIDI) has been a help to musician and music lovers. Even some video games use MIDI to enhance the experience of players.

To play a song, mere people can only rely on hearing to know how to play it. However, not everybody is expert in interpreting songs. If that person interprets the pitch wrongly, it will lead on playing a wrong song. To solve this problem, standardized musical notation is needed. It is the same with MIDI, people cannot play it by themselves unless they know the chords behind.

To help people understand the music inside the MIDI file, and to help them reproduce the music by themselves, they have to know the chords of the song. This paper shows that MIDI file can be converted into musical notation because all the information needed is contained inside the file itself. The application shows that conversion is possible, as long as the file inputted is the correct MIDI type 0 or 1 file.

References

[1] MIDI Manufacturers Association. http://www.midi.org/index.php. February 8, 2009.

[2] Java Sound Programmer Guide. http://java.sun.com/j2se/1.4.2/docs/guide/sound/programmer_guide/chapter1.html. January 21, 2009.

[3] Java Sound Programmer Guide. http://java.sun.com/j2se/1.4.2/docs/guide/sound/programmer_guide/chapter8.html. January 21, 2009.

[4] Notation website. http://www.notation.com. January 29, 2009.

[5] Website for learning music. http://www.treblis.com. January 21, 2009.

[6] MIDI File Parsing. http://www.ccarh.org/courses/253/assignment/midifile. March 14, 2009.