Contents

1. Purpose 1
2. System Overview 1
3. Functions 1
   3.1 main 1
   3.2 gui 1
      3.2.1 Design considerations 2
   3.3 input 3
   3.4 guisettings 3
   3.5 output 3
   3.6 midiplayer 4
   3.7 playalong 4
      3.7.1 Design considerations 4
   3.8 globals 5
4. User Manual 6
   4.1 Starting the program 6
   4.2 Loading a file 6
      4.2.1 Troubleshooting 6
   4.3 Starting playback 7
      4.3.1 Troubleshooting 7
   4.4 Stopping playback 7
   4.5 Playing the drum kit 7
      4.5.1 Using the keyboard 7
      4.5.2 Using a MIDI input device 8
5. Sources 9
   5.1 Libraries 9
   5.2 References 9
   5.3 Software 9

MIDI Drum Trainer

1. Purpose

The purpose of MIDI Drum Trainer is to teach new or young drummers how to play the drums without learning drum notation. The user is not limited to a small selection of songs as they would be with a song book; they can learn to play any song, so long as they have a MIDI file with a drum track. The user can load a MIDI file into the program and watch as the song is played on a virtual drum set. In addition to being a learning tool, MIDI Drum Trainer is also fun to watch.

2. System Overview

MIDI Drum Trainer consists of the following files:

- **main**: Passes control to the interface.
- **gui**: Initializes the interface, detects user interaction with the interface, and updates the interface throughout the program run.
- **input**: Reads and analyzes inputted MIDI files.
- **guisettings**: Initializes and detects user interaction with the settings interface.
- **output**: Creates a modified MIDI file for playback according to the inputted settings.
- **midiplayer**: Controls the audio playback of MIDI files.
- **playalong**: Controls the visual playback of the drum track.
- **globals**: Contains variables used throughout the program files.

3. Functions

3.1 main

Function	Args.	Returns	Description
main	-	-	Passes control to the interface.

3.2 gui

Function	Args.	Returns	Description
createInterface	-	-	Creates the main interface. Sets attributes for the interface window and enters the main loop for the interface.
closeInterface	-	-	Closes the main interface. Stops any currently playing songs and deletes the temporary output MIDI file.
createButtons	-	-	Creates the button area of the main interface, which contains a frame, a Load button, and a Play button.
createCanvas	-	-	Creates the image area of the main interface, which contains a frame and a canvas.
initializeImages	-	-	Initializes the drum images on the canvas.
resetImages	-	-	Moves the “off” images and the “user on” images to their starting positions on the canvas.
loadFile	-	-	Triggered when the Load button is pressed. Presents the user with a dialog box and accepts a MIDI

			file. Passes control to the settings interface.
playPause	-	-	Plays or stops audio and visual playback of MIDI file.
noteOn	drumNum	float	Changes the drum numbered **`drumNum`** to the "on" state image. Returns the time taken to update the interface.
noteOff	drumNum	float	Changes the drum numbered **`drumNum`** to the "off" state image. Returns the time taken to update the interface.
showError	message	-	Displays an info box with **`message`**.
pressKey	event	-	Updates the drum image that corresponds to the key specified in **`event`** to the "user on" state image.
releaseKey	event	-	Updates the drum image that corresponds to the key specified in **`event`** to the "off" state image.

3.2.1 Design considerations

3.2.1.1 Drum on/off images

Finding an efficient way to switch drum images between their "on" and "off" states was one of the biggest problems in the development of the program.

The first technique used recreated each image as it was turned on. This consumed too much time, and consequently, the timing of the visual playback was incorrect.

The second technique created the "off" state images in their starting positions and the "on" state images off screen, and then moved the "on" image on screen when its corresponding note was "on". This was an improvement, but not enough for the lag not to be noticeable. Thus, it was necessary to keep track of how long it took to move each image and update the interface, and then take this time into account in the playAlong() function.

In addition to the regular "off" state images and the red "on" state images, there is a third blue "user on" state for user keyboard or MIDI input. While using a GIF image instead of a PNG would be preferable in terms of time, GIF does not support alpha transparency. When a note is played by the user, since the blue images are semi-transparent, when placed on top of the red "on" images, a purple drum indicates that the note was played by the user at the correct time. Unfortunately, using user input during MIDI playback causes a very noticeable lag due to additional time taken to update the interface.

3.2.1.2 User input

Ideally, when a user presses a key on the keyboard or hits a drum on a MIDI input device, the selected drum would be highlighted on the screen and the MIDI note for that drum would be heard. The Pygame library, which handles playback of music files, can also handle playback of individual MIDI notes. Unfortunately, the library does not seem to be able to handle doing both in the same program. Thus, the program does not play any drum notes during user input, since the MIDI file playback feature is more important to the overall program.

3.3 input

Function	Args.	Returns	Description
readMidiFile	guiSelf, midiFilename	boolean	Reads the file **`midiFilename`** as a MIDI file. Finds drum tracks and gets tempo to be used in the settings interface. Returns **`True`** if the file was read successfully.
getDrumTrackTitleList	midiFile	string[]	Finds all drum tracks in **`midiFile`**. Returns a list of drum track titles.
getTrackTitle	trackIndex, titleLine	string	Determines the title of track **`trackIndex`** from **`titleLine`**. Returns the track title.
getTempo	midiFile	int	Finds a SetTempoEvent in **`midiFile`**. Returns the tempo in beats per minute (BPM).
readDrumTrack	midiFile, drumTrackIndex	string[][]	Reads track **`drumTrackIndex`** in **`midiFile`**. Returns a two-dimensional array of events in which each row consists of "On"/"Off", tick length, and the percussion note for one event.

3.4 guisettings

Function	Args.	Returns	Description
createSettingsInterface	-	-	Creates the settings interface. Sets attributes for the settings interface window.
closeInterface	-	-	Prevents closing the settings interface.
updateSettings	guiSelf	boolean	Updates the tempo and drum track using the user's selected settings. Returns **`True`** if output MIDI file was created successfully.
createWidgets	-	-	Creates the settings interface. Creates a frame, labels, and input controls.
ok	-	-	Triggered when the OK button is pressed. Checks for valid input and passes control to updateSettings().
showError	message	-	Displays an info box with **`message`**.

3.5 output

Function	Args.	Returns	Description
createOutputMidiFile	settings	MIDI	Duplicates the original input MIDI file and modifies it according to **`settings`**. Returns the modified MIDI file.
createMutedDrumsMidiFile	settings, outputMidiFile	MIDI	Modifies **`outputMidiFile`** by

			removing the drum track specified in **`settings`**. Returns the modified MIDI file.
createIsolatedDrumsMidiFile	settings, outputMidiFile	MIDI	Modifies **`outputMidiFile`** by removing all but the drum track specified in **`settings`**. Returns the modified MIDI file.

3.6 midiplayer

Function	Args.	Returns	Description
initializeMidiPlayer	guiSelf	boolean	Initializes the music player and loads **`midiFile`**. Returns **`True`** if the file was loaded successfully.
stopOrPlay	guiSelf	boolean	Stops playback if currently playing, and starts playback if currently stopped. Returns **`True`** if playback was started.
stopAll	guiSelf	-	Stops both audio and visual playback.
playAll	-	-	Starts both audio and visual playback.

3.7 playalong

Function	Args.	Returns	Description
initializePlayAlong	guiSelf, drumTrack, tempo, resolution	-	Analyzes **`drumTrack`** to determine the drum image number for each note and the time to wait before and after each note.
getDrumNum	midiNote	-	Returns the image number for the drum that corresponds to **`midiNote`**.
calculateWaitTimes	drumTrack, ticksPerSec	float{}	Calculates the time in seconds to wait before and after every note in **`drumTrack`**, based on the **`ticksPerSec`** of the MIDI file. Returns a dict containing the wait times for each event.
playAlong	guiSelf, drumTrack	boolean	Handles visual playback of **`drumTrack`**. Loops through each event in the track, waiting before and after notes are turned on or off by **`guiSelf`**. Returns **`True`** if the song was played completely.

3.7.1 Design considerations

3.7.1.1 Pre-analyzing drum tracks

The initializePlayAlong() function is called before the user is allowed to play the file. Although it causes a slight delay between loading the file and playing the file, it decreases processing time during the actual playback of the song. By calculating the drum numbers and the wait times in advance and saving these values, redundant calculations are eliminated if a song is played more than once.

3.7.1.2 Wait times

The time to wait between each note is split up into "before" wait time and "after" wait time. The times are split up because note durations are stored as ticks in MIDI note *off* events, while some rests may be stored as ticks in MIDI note *on* events. It was simpler to store the note on event ticks as "before" wait times and the note off event ticks as "after" wait times, rather than combining the two into one wait time.

3.8 globals

Variable	Default	Description
APP_MAIN	-	Used by the interface.
CANVAS_WIDTH **CANVAS_HEIGHT**	635 306	The size of the interface image canvas.
DRUMS[]		The image number corresponding to each drum.
DRUM_NAME[]		The name of each drum.
CURRENT_EVENT	0	The index of the drum track event currently being played.
DRUM_NUM	[]	The image number for the drum corresponding to each event in the currently loaded drum track.
DRUM_TRACK	""	The currently loaded drum track.
DRUM_TRACK_TITLE_LIST	[]	The titles of each drum track in the currently loaded MIDI file.
DRUM_TRACK_INDEX_LIST	[]	The track number of each drum track in the currently loaded MIDI file.
IS_PLAYING	`False`	`True` if audio and visual playback is on.
KEY[]		The keyboard keys corresponding to each drum.
MAX_TEMPO **MIN_TEMPO**	300 10	The range of the user-entered tempo in BPM.
MIDI_FILE	""	The analyzed MIDI file containing MIDI events.
MIDI_FILENAME	""	The complete file path of the currently loaded file.
MIDI_FILENAME_SHORT	""	The shortened filename of the currently loaded file.
NOTES[]		The MIDI notes corresponding to each drum number.
OUTPUT_MIDI_FILENAME	"output.mid"	The filename for the modified MIDI file.
PERCUSSION_CHANNEL	9	The channel used for percussion in MIDI.
POS_TOP[] **POS_LEFT[]**		The top and left pixel positions for each drum image.
ROOT	""	Used by the interface.
SETTINGS	{}	A dict to store user inputted settings (drum track, tempo, and mute settings).
TEMPO	120	The tempo in BPM of the currently loaded file.
WAIT_BEFORE **WAIT_AFTER**	[] []	The time in seconds to wait before or after each event in the drum track.

4. User Manual

4.1 Starting the program

1. Navigate to the **dist** folder.
2. Double click on **main.exe** to run the program.

4.2 Loading a file

Note: Several sample MIDI files are located in the **dist/midis** folder.

1. Press the **Load File** button.
2. Select a MIDI file from the local file directory.
3. Press the **Open** button.
4. Choose your settings:
    a. **Select drum track:**
    If there is more than one drum track, you may choose to use any one of them by clicking on the track's name.
    b. **Enter tempo (BPM):**
    The default tempo is the tempo used by the MIDI file. Enter an integer between 10 and 300 to change the tempo. The higher the tempo, the faster the file will play; the lower the tempo, the slower the file will play.
    c. **Mute drum track:**
    You may select this checkbox so that the drums will not be heard during file playback. If this checkbox is selected, the following checkbox may not be selected.
    d. **Drums only:**
    You may select this checkbox so that the drums will be the only instrument heard during file playback. If this checkbox is selected, the previous checkbox may not be selected.
5. Click the **OK** button.

4.2.1 Troubleshooting

4.2.1.1 ERROR: _____ is not a MIDI file.

Problem: The inputted file does not have a .mid or .midi extension, or the file is corrupted.
Solution: Choose another MIDI file.

4.2.1.2 ERROR: Could not find any drum tracks in _____.mid.

Problem: The inputted file does not contain any drums tracks.
Solution: Choose a MIDI file that contains at least one drum track.

4.2.1.3 ERROR: _____.mid is in the wrong MIDI format. Please use a MIDI format 1 file.

Problem: The inputted file is in MIDI format 0. MIDI Drum Trainer can only accept files that are in MIDI format 1 or 2.
Solution: Choose a MIDI file in format 1 or 2.

4.2.1.4 ERROR: Tempo must be between 10 and 300 BPM.

Problem: The number entered for the tempo is either less than 10 or greater than 300.
Solution: Pick a number that is between 10 and 300.

4.2.1.5 ERROR: Please select one check box only.

Problem: The **Mute drums** and **Drums only** checkboxes have both been selected.
Solution: Select only one of the two checkboxes, or select neither checkbox.

4.2.1.6 ERROR: Please enter an integer.

Problem: The input in the tempo box is not a number (eg. "fifty", "100.5", "120 BPM").
Solution: Enter a number in the tempo box (eg. 60, 120, 200).

4.3 Starting playback

1. Load a file (see section 4.1: **Loading a file**).
2. Press the **Play ____.mid** button (where ____ is the name of the MIDI file).

4.3.1 Troubleshooting

4.3.1.1 ERROR: Could not read wait times. Please try another MIDI file.

Problem: The inputted file may not contain a drum track, or the file is not in proper MIDI format.
Solution: Choose another MIDI file.

4.4 Stopping playback

1. Load a file (see section 4.1: **Loading a file**).
2. Play the file (see section 4.2: **Starting playback**).
3. Press the **Stop ____.mid** button (where ____ is the name of the MIDI file).

4.5 Playing the drum kit

The drum kit can be controlled by the keyboard or a MIDI input device. Note: The drum kit does not play drum notes; it only highlights the selected drum on the screen in blue.

4.5.1 Using the keyboard

Each drum corresponds to a key on the keyboard. This key is indicated on the image of each drum.

Key	Drum
a	bass drum
s	snare drum
d	high tom
f	low tom
g	floor tom
h	crash cymbal
j	closed hi-hat

Pressing any of the above keys will cause the corresponding drum to be highlighted on the screen.

4.5.2 Using a MIDI input device

Note: MIDI Drum Trainer has only been tested using the Rock Band drum controller.

Since the Rock Band drum controller is detected by the computer as a joystick, a third party program such as JoyToKey is required to translate the drum controller input into keyboard input.

1. Plug the drum controller into the computer.
2. Download the Joy2Key zip file from the JoyToKey website.
3. Unzip the downloaded file.
4. Run the JoyToKey.exe file within the unzipped directory.
5. Configuring which drum pad corresponds to which key on the keyboard may take some trial and error. It is likely that the drum pads will correspond to one of Button 1, 2, 3, etc. One possible configuration for a right-handed Rock Band drum controller is as follows:

Button	Key	Drum Pad	Drum Sound
Button 1	F	blue	low tom
Button 2	J	green	closed hi-hat
Button 3	S	red	snare drum
Button 4	D	yellow	high tom
Button 5	A	orange (pedal)	bass drum

   To configure the key for each button:
   a. Double click on the Button row.
   b. Press the key on the keyboard that you want that button to correspond to (eg. **a**).
   c. Press the **OK** button.
   d. Repeat for each drum pad.

Note: The JoyToKey program must remain running while MIDI Drum Trainer is running.

Hitting any of the configured drum pads will cause the corresponding drum to be highlighted on the screen.

5. Sources

5.1 Libraries

Pygame: http://www.pygame.org/
Python Imaging Library: http://www.pythonware.com/products/pil/
Python Midi: https://www.github.com/vishnubob/python-midi/
Tkinter: http://docs.python.org/library/tkinter.html (standard Python)

5.2 References

General MIDI. (2012). Retrieved from Wikipedia:
http://en.wikipedia.org/wiki/General_MIDI#Percussion
(drum note numbers)

MIDI File Format. Retrieved from Personal Computer Technical Reference:
http://www.piclist.com/techref/io/serial/midi/midifile.html
(information about MIDI files)

The MIDI File Format. Retrieved from FileFormat.Info:
http://www.fileformat.info/format/midi/corion.htm
(information about MIDI files)

Midi Timestamp in Seconds. (2011). Retrieved from Stack Overflow:
http://www.stackoverflow.com/questions/7063437/midi-timestamp-in-seconds
(formula for converting MIDI ticks to seconds)

.MIDI with PyGame. (2011). Retrieved from Pastebin: http://pastebin.com/zrgjRSwY
(code for playing music files with Pygame)

Shipman, J. (2010). *Tkinter 8.4 Reference: A GUI for Python*. Retrieved from New Mexico Tech:
http://infohost.nmt.edu/tcc/help/pubs/tkinter.pdf
(GUI-related code snippets and guides)

Tkinter Classes. Retrieved from Pythonware:
http://www.pythonware.com/library/tkinter/introduction/tkinter-classes.htm
(GUI-related code snippets and guides)

5.3 Software

Ableton Live: http://www.ableton.com/
(which keyboard key controls each drum)

Joy2Key: http://www.electracode.com/4/joy2key/JoyToKey%20English%20Version.htm
(enables MIDI device input to the program)

py2exe: http://www.py2exe.org/
(creates executable)