impromptu

interactive music notation

Team members: Ivy Malao, Zoë Naidoo, Annie Chen (Mengyuan Chen), Rachel Pinsker, Zakir Gowani, Sofia Wyetzner

MILESTONE 4.B

Code can be checked out from our Github repository at https://github.com/rpinsker/impromptu

Library dependencies to be downloaded and installed:

Front end: LilyPond, Abjad 2.17, Flask (please refer to Milestone 3.B for installation instructions)

Backend: Python-midi (please refer to Milestone 3.B for installation instructions)

** New library: aubio **

Download aubio (http://aubio.org/download) and follow the instructions in the aubio readme file:

./waf configure

./waf build

sudo ./waf install

If waf is not found in the directory, you can download and install it with:

make getwaf

(1) and (2) how to compile and run code

When inside the `src` folder, type '`Python app.py`' into the terminal in order to run the program.

Go to the URL http://127.0.0.1:1995/ where the web application should be running on your localhost.

You can also run the backend functionality independently by typing '`Python Tune.py -f [filename]`' into the terminal, where [filename] is a MIDI file path, e.g. '`c-major-scale-treble.mid`'. The program will print out the MIDI file pattern and all of the Notes in the MIDI or Wave file to the terminal.

Note that the "obsolete" folders contain older versions of code that we are keeping as reference.

Downloaded and saved JSON / WAVE files are saved to the "Downloads" folder for easy access for uploading into the program

(3) how to run the unit test cases

The verbose flag (-v) allows you to run the test with more detail to see which tests pass and fail.

Backend tests: When inside the src folder, run: `python ../tests/backendtests.py -v`

Frontend tests: When inside the src folder, run: `python ../tests/flaskFrontendTests.py -v`

(4) please suggest some acceptance tests for the TA to try (i.e., what inputs to use, and what outputs are expected)

Main acceptance tests are uploading valid MIDI files and Wave files. First, download MIDI files provided in tests/MIDITestFiles and Wave files provided in tests/WAVTestFiles. Other MIDI files can be downloaded from: https://www.basicmusictheory.com/major-scales if you would like to test more. Wave files can be searched on the internet.

Once a file is uploaded and a pdf of the sheet music is displayed, you can edit the title, contributors, etc, add and delete events, and edit events.

The correct PDFs for each of the MIDI files are also in the MIDITestFilesFolder with a name matching the name of the corresponding MIDI file to be compared to for accuracy. All of the provided MIDI and Wave Files should render.

(5) and (6) text description of what is implemented (can refer to the use cases and user stories) and who did what and who was paired with whom

We split into frontend and backend groups as did “pair programming” in our separate groups. The frontend group was Sofia, Rachel, and Zakir; the backend group was Ivy, Zoë, and Annie. This iteration focused on rendering chords, editing sheet music, downloading and uploading sheet music as a json file, recording audio, and generating sheet music from Wave audio files (please note that mp3 files are not supported as explained in part (7) changes.

All team members worked on writing their relevant tests and debugging both the code and the tests.

- Upload WAVE audio file (**frontend** -- Zakir)
- all UI, including editing sheet music (**frontend** -- Sofia)
- Implement clef, key, time signature on PDF display (**frontend** -- Rachel)
- Process and render chords (**frontend** -- Rachel and **backend** -- Ivy, Zoë)
    - Chord, Note, and Rest will now be subclasses of the superclass Event
- Support variations within MIDI format 1 type, e.g. when no note off event (**backend** -- Annie)
- Support one-track/single-instrument WAVE audio file by generating Tune object directly from wav audio file (**backend** -- Zoë) using Aubio, specifically Aubio’s method *aubionotes*
    - This extracts MIDI note, onsets, and offsets for each note from the wav audio file
    - Note convert frequencies to pitch method is unused as Aubio can extract MIDI note directly
- Record instrument and store as WAVE file (**frontend** -- Zakir)
- Edit sheet music functionality (**frontend and backend**):
    - Adding/Deleting notes -- Zakir, Ivy
    - Editing a note’s pitch and duration -- Rachel, Ivy, Sofia
    - Displaying notes of a given measure for editing -- Rachel, Sofia, Zoë
- Save files as impromptu sheet music. This means downloading as a json file to later upload and re-convert into a Tune object for display. Frontend for creating json file, downloading json file, and

saving uploaded json file and backend for reading a json file to make a Tune object. (**frontend** -- Zakir and **backend** -- Annie)

(7) changes: have you made any design changes or unit test changes from earlier milestones?

Design Changes:

- **Event superclass comprising Chord, Note, and Rest events:** To implement chords, which have similar structure to notes, we created the Event superclass. We also separated Rests as their own subclass whereas in iteration 1, Rests were Notes with pitch letter ‘r’ and frequency 0.
- **Wave audio file support (extension .wav) rather than mp3 audio file support:** We chose to use Wave audio files instead of mp3 files because mp3 files are a compressed format and are harder to work with (and is a derived format from the more direct sound translation Wave format). Wave file functionality is sufficient as one can record to the Wave format and there are sufficient libraries
- Frontend now supplies measure information to abjad to allow for editing by measure.

Unit Test Changes:

Minor Frontend unit test changes due to functionality changes:

- Using Event class instead of Note (setEventsList instead of setNotesList, etc.)
- test_tune_to_notes and test_making_chord adjusted because return of this function now deals with measures
- Providing a measure number when editing pitch and duration requests
- Test_tune_to_measures adjusted because the the format of the output was rethought
- Elimination of mp3 upload test because mp3’s are no longer supported
- Creation of wav upload test because wave files are supported

For the backend, Frequency-related methods (i.e. frequency to pitch) are still irrelevant as Aubio can give either MIDI notes or frequency, and MIDI note support is already built from iteration one. Unit tests were added for completeness and minor changes were made to unit tests to fix logical errors in the unit test or update method names.