Busic - Installation Guides and User Manual

Bolong Zeng

School of Electrical Engineering and Computer Science
Washington State University, Pullman, WA 99164
{bzeng}@wsu.edu

This manual includes brief introductions on the requirements for running the project (Busic), basic use cases, and currently known issues, as well as potentail future enhancements and updates options.

1 Running Requirement

Busic is built upon Python 3.4.1, Kivy 1.8.0, pygame 1.9.2. It has only been built and run successfully on Windows 7/8.1 machines (both 32 and 64 bit platforms), therefore it also requires available corresponding MinGW installation.

1.1 Python 3.4.1

It is assumed the user have at least Python 3.4+ successfully installed. Available instructions can be found at: Python-3.4.1-Download.

1.2 Kivy 1.8.0

Official complete installation instructions can be found at: Kivy-Installation-Windows. However, there are several short cuts for the process. With existing Python 3.4 installation, the user could refer to Kivy-with-Existing-Python for instructions. Specifically, Kivy-exes provides the exe files that can be quickly installed for 32 and 64 bit windows machines.

A small module from Kivy Garden also needs to be installed: FileBrowser. Instructions can be found at: Kivy-Garden-Instruction. To summarize, first install Kivy Garden by running *pip install kivy-garden*. Then, install the FileBrowser package by running *garden install filebrowser*.

1.3 pygame 1.9.2

Kivy itself requires certain features from pygame packages to be fully functional. Busic also uses these features. Similarly, pygame installation files for windows machines can be found at pygame-exes.

1.4 MinGW

Information on how to install MinGW on a Windows machine can be found at MinGW-Installation-Guide. Note that for manual installation, the PATH environment variable of the system has to been changed to include the MinGW bin directory.

1.5 Note on support on Linux

Linux installation and execution has been successfully test. Yet details on support varied among distributions and are waiting to be updated.

2 Basic Use Cases

The key features of Busic 1.0 include music playing, a simple playlist of songs, and most importantly, the ability to support synchronized lyrics, both displaying and editing. This section describes the basic steps of several simple use cases. Figure 1 shows the main interface of Busic 1.0. All the essential operations will update the status bar on the bottom of the window, for clarity and error tracking purposes.

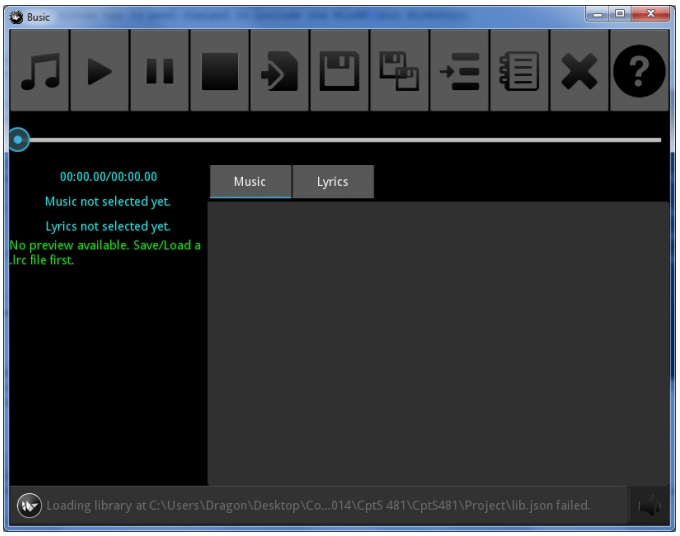

Fig. 1. Main GUI

2.1 Playing Music

Initially the program would have an empty playlist while no library is available. The user can click the "Load Music" button with the music icon to load songs. At the moment only .ogg files are fully supported, while .wav files can be played but with potential issues. The package contains several sample music file in the *music* directory. Once a music file is loaded, it will be added to the play list, and can be played using the basic playback control buttons on the toolbar. Figrue 2

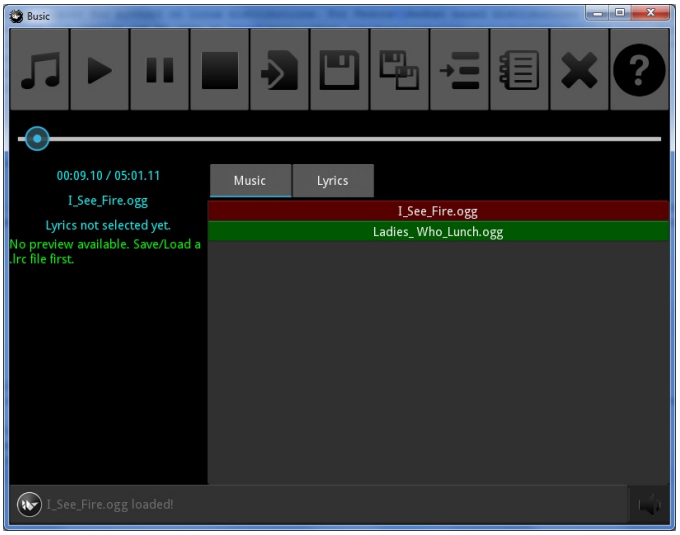

Fig. 2. Playing

shows the program while it is playing a song. On the bottom right corner, Busic also provided a single button that can be used to mute/unmute the music.

2.2 Managing Playlist

Once there are songs available in the playlist, the user can save the list to a *lib.json* file by clicking the "Save playlist" button with the notepad icon. The library file is automatically loaded in future runnings of the program. A selected song can also be deleted from the list by clicking the "Delete from playlist" button.

2.3 Lyrics Displaying

Busic 1.0 supports both regular lyrics (.txt) and synchronized lyrics (.lrc) files. Synchronized lyrics labeled each line of lyric with a time stamp of the song. Busic is able to read in the time stamp information and display the lyrics accrodingly while the music is playing. Figure 3 shows a regular lyrics is loaded when using the "Load lyrics" button.

If the loaded lyrics file is a legitimate .lrc file, then the preview area shows the lyrics with a karaoke effect by highlighting the current line of lyrics, as shown in Figrue 4.

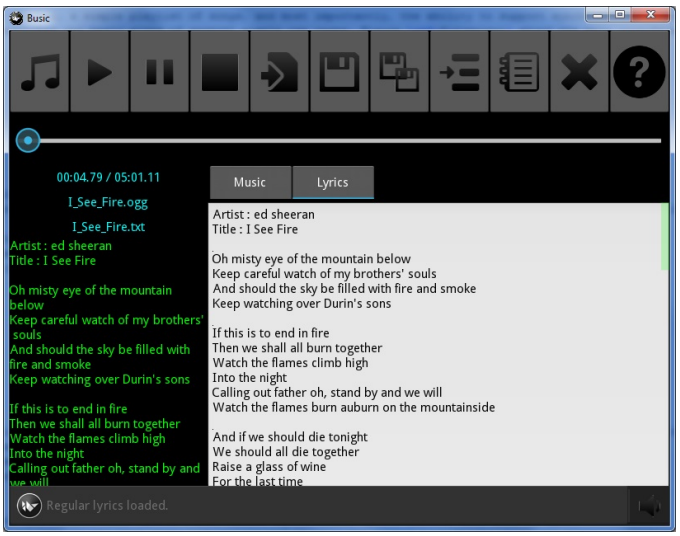

Fig. 3. Displaying regular lyrics.

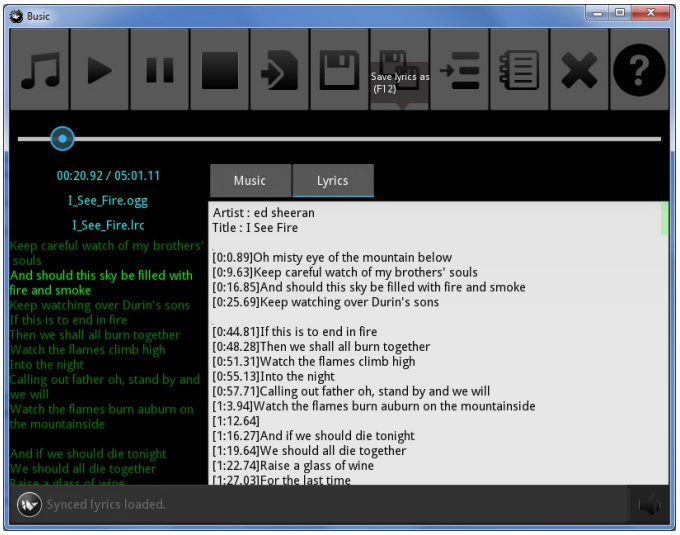

Fig. 4. Displaying synchronized lyrics.

2.4 Lyrics Editing

The core feature of Busic 1.0 is that it enables the user to edit a regular lyrics file and turn it into a synchronized .lrc file. First, the user can enter lyrics either by hand, by copying a lyrics file into the text input area, or load a regular .txt file. The file content is then displayed in the text input area.

The user can now select a music to be played. Once the music starts, the user can move the cursor to the start of the lyrics, and wait until the music played to the current line. At that point, the user can either press F5, or use the "Insert time stamp" button in the toolbar to insert the current time stamp. The cursor is then automatically moved to the head of the next line, so the user can keep inputting time stamps. The process is shown in Figure 5.

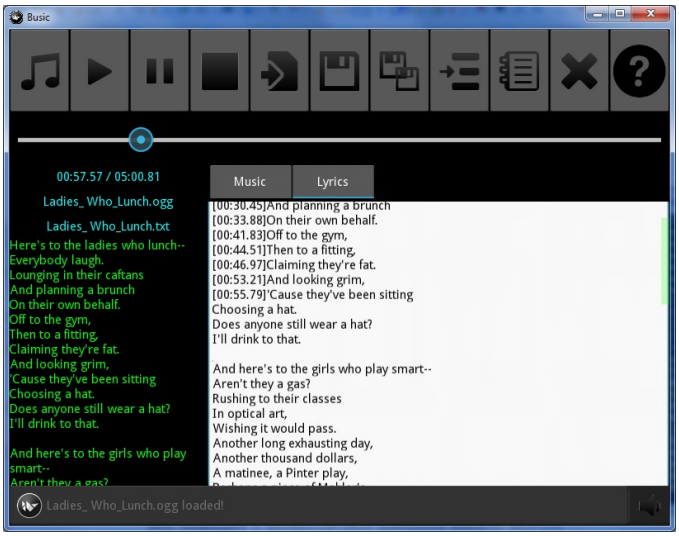

Fig. 5. Displaying synchronized lyrics.

Afterwards, the user can save the lyrics by using either the "Save" (F6) or "Save as" (F12) button into .lrc file. Once it is saved, the user can then review the edit by playing the music again.

3 Known Issues and Potential Updates

This section summarizes several issues that existed in Busic 1.0, and possible updates directions for future improvements.

1. The most important issue, the support for audio format is minimal at the moment, with only full support on .ogg files. Reasons for this is because Kivy

1.8.0 has a bug in its audio module, and we have to retreat to pygame's audio support, which has only limited support for popular types such as .mp3. Meanwhile, large .wav files, while can be played, will have incorrect behavior or even noises throughout playing. This is definitely one direction that is worth improving.
2. Since .mp3 file and other audio types are not supported, the playlist function at the moment is also as simple as it can be. Once Busic can support most audio types, the next milestone would be to parse the ID Tag information with the .mp3 files, and make the music library and playlist more powerful. Besides, it would be desirable to implement a drag-and-drop features that supports dragging music files from the file explorer directly, instead of using the load functionality.
3. Seeking function during playback is not implemented at the time, due to the developer being unable to solve a touch down event related to the slider widget in Kivy. This would make the playback control feature much more user friendly.
4. Many details re-tuning could be done in the future. For instance, multi-language/unicode characters are currently not supported. The volume control is only limited to mute and unmute as well. These are all features that can be enhanced to improve the usability of the software.
5. Insufficient time for testing and coding documentation.
6. Simple GUI design, both externally and internally. The GUI programming is the developer's most significant weakness. For this reason, both the layout and internal logic is not well designed, especially the logic control within the program are highly coupled among classes. Certainly can be improved with more OO focus.