Scripting Audacity with Python

Maxime Casara

LIACS, Leiden University

Abstract. This paper describes the implementation of scripted macros for Audacity. By using macros, the user not only saves time but also bypasses the GUI of the software. A flowchart is provided to help the user navigate within the different levels of the program. The current version supports the quick modification of any audio sample's volume, pitch and tempo, and allows to add effects such as wahwah, echo, fadein, fadeout, reverb and paulstretch.

1 Introduction

Audacity is a free and open-source digital audio editor and recording application software created in 1999 by Dominic Mazzoni and Roger Dannenberg at Carnegie Mellon University and released on May 28, 2000. It is one of the most popular audio editor with a total of 100 million downloads. Audacity won the SourceForge 2007 and 2009 Community Choice Award for Best Project for Multimedia. The software accepts scripts for almost every built-in commands. The architecture of Audacity is shown in the figure below:

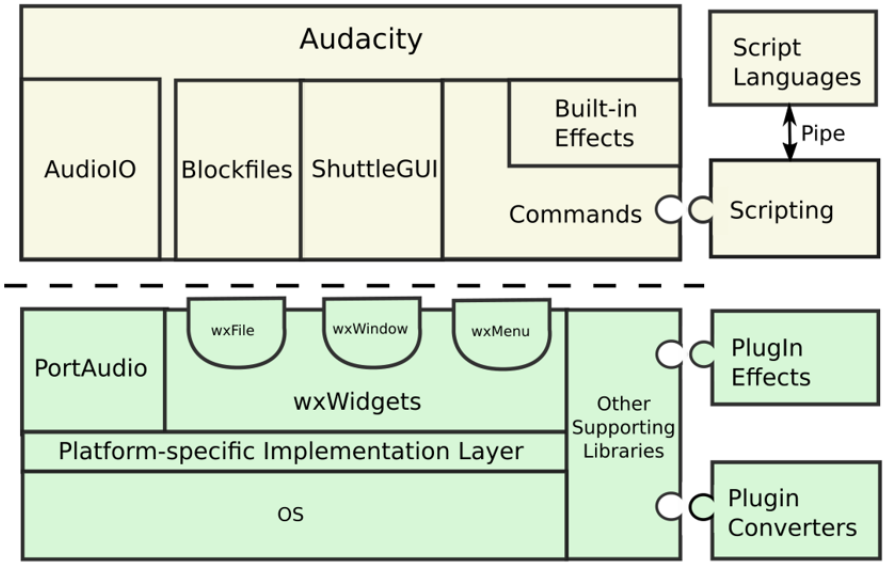

Fig. 1. Software architecture of audacity

1.1 Goal

The goal of the project is to develop a list of macros to quickly perform different sound effect on an audio sample, bypassing any user-interface interaction. The main motivation was to apply these macros in a context where there is not much time to fix and tune audio samples, for example, in a theater where technical preparations must be carried out on the very same day as the representation.

1.2 Previous work

Previous work on this subject is scarce or unpublished. Only the method to build the pipeline to connect the software to the script language [2] and the implementation of some basic commands can be found. All the scripting commands are referenced [1] by the team in charge of the software. However, the commands are not always accurate, and using them is not as straightforward as it looks to be. Some scripting syntax are deceiving, which might come from the fact that the reference page was automatically generated.

2 Design

The algorithm can be described with a flowchart with different levels. The flowchart syntax can be found in [1].

[1] https://manual.audacityteam.org/man/scripting_reference.html

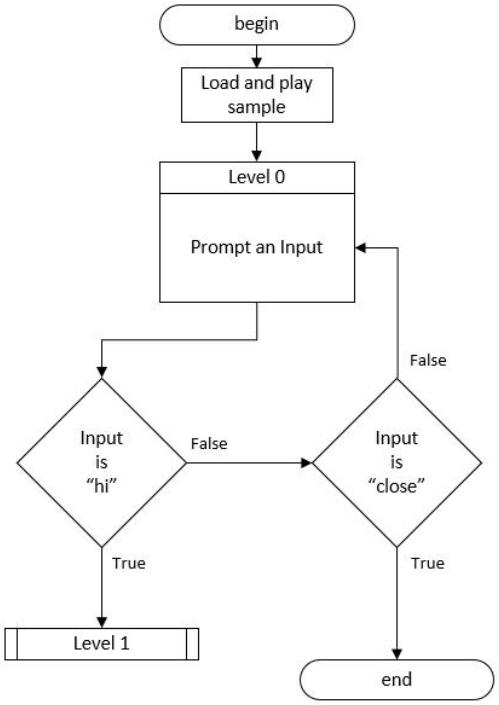

Fig. 2. Flowchart level 0

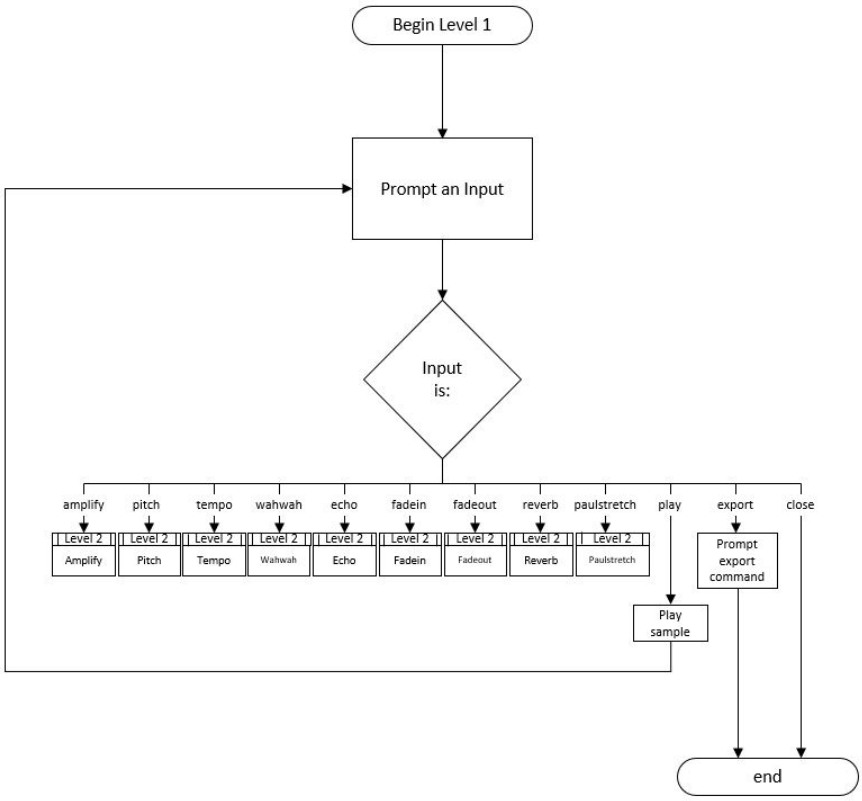

Fig. 3. Flowchart level 1

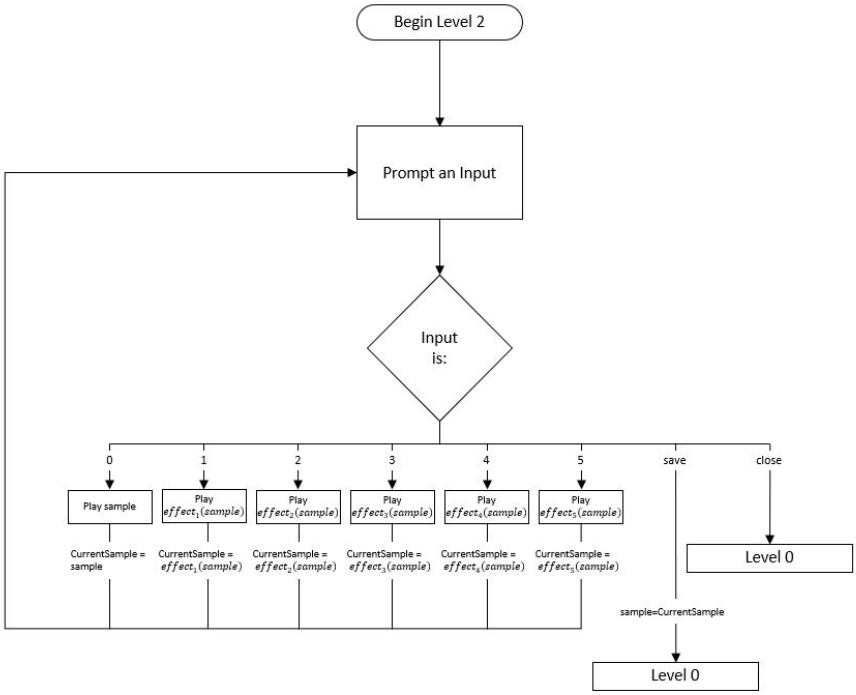

Fig. 4. Flowchart level 2

3 Implementation

The program is entirely coded in python and does not require any specific libraries. The 3 levels described in the flowchart serve different purposes:

Level 0:

– In this level, the sound sample is loaded and played once. There, the user can either go to level 1 by entering "hi" or end the program by entering "close".

Level 1:

– This level contains all the actions that can be performed on the sound sample. Either applying an effect, by entering the name of the effect, playing the sample, closing the program or exporting the sample. The command to close and to export both end the program. Playing the sample loops back to level 1. Applying an effect brings to the level 2.

Level 2:

– Entering this level commands Audacity to duplicate the sound sample 5 times and apply on each copy the effect with arbitrarily varying intensity. The user can now chose to play each sample by entering a number from 0 to 5, where 0 is the original sample, 1 the least altered sample and 5 the most altered sample (with few exceptions for some effects). The 'save' command overwrite the original sample with the most recently played one and brings the user back to the level 0, with the new sample loaded. Finally, the user can quit and discard any change with the command 'close' which brings back to the level 0, keeping the original sample.

4 Experimentation

The metric that can be used to validate the performance of the solution is the time we save by using the program rather than Audacity's GUI to fix and tune sound samples. In the case where the user does not know how to use Audacity, the program comes in handy. Indeed, during the presentation of the solution, I allowed people from the audience to test the program and apply effects on a given sound sample. The results were obtained rapidly and easily by issuing simple commands from the program, by people with no prior knowledge of Audacity. In the case of experimented users, they would notice limitations such as not being able to fine-tune or customize the parameters of an effect. For example, the reverb effect has 9 parameters (room size, pre-delay, reverberance, damping...) that require tuning. In the current version of the solution, all parameters are linearly proportional to the sample copy number.

Future improvements therefore include the option for the user to fine-tune parameters. All effects are also not yet implemented.

5 Software Requirements

The program is run bug-free on Python 3.6 and Audacity 2.3.3.

References

1. https://fulmanski.pl/zajecia/ics/materialy/compalgorithm.pdf
2. https://github.com/audacity/audacity/blob/master/scripts/piped-work/pipe_test.py
3. https://manual.audacityteam.org/man/scripting_reference.html