CocoMUD client - Feature #140

Use another library to play sounds

12/01/2018 12:48 PM - Vincent Le Goff

Status:	Closed	**% Done:**	100%
Priority:	Normal		
Assignee:	Vincent Le Goff		
Category:	Customization		
Sprint/Milestone:	16		

Description

In the past, CocoMUD used Pygame, a reasonable choice to play sound but much too powerful and heavy for this very simple task. The audio support in pygame also turned out to be not very extended. This all encouraged to find an easier and lighter alternative with more power.

For this new library, the basic requirements are as follow :

- Ability to play at least WAV, MP3 and OGG file formats.
- Ability to run on several platforms (though for the time being, CocoMUD only works well in Windows and is only partially ported on Linux).
- Ability to work with Python 3 (new requirement, since we've moved to Python 3.6).
- Optional ability to play sounds and keep a reference to them, so we can stop them (interrupting before the sound is over).
- Optional ability to handle the volume on individual sounds (perhaps some other features might be a nice addition to CocoMUD itself).

Please post alternatives as feature comments below. If you feel like doing it, try to work on the library implementation in a separate branch (one branch per library), though discussing the library in advance is probably better.

History

#1 - 12/01/2018 12:59 PM - Vincent Le Goff

So far, one library seems to stand out: Pybass. It seems to answer to requirements. Unfortunately, it would require further investigating, the library itself being somewhat sparsely documented (as far as I could see) and the sound mechanism being mostly low-level Py/C functions. We'll have to come up with a wrapper, something very simple that probably allows to only play audio from the file system. Recording is not needed and handling most sound effects doesn't seem necessary for the time being.

One successful implementation of the library is TWBlue, which has successfully used it in the accessible Twitter client. Most of the features used in the wrapper aren't needed for CocoMUD, and reading the source code both of TWBlue's sound_lib package and Pybass extention seems necessary, though I still believe we can simplify the wrapper to one or two classes, to begin with anyway. You might look at src/sound_lib/output.py which provides an example of wrapper around the Pybass library.

#2 - 12/01/2018 07:43 PM - Vincent Le Goff

- File wx_ctrl_phoenix.py added

I was able to play sound (WAV, MP3 and OGG files) and keep a handle on them with Pybass. I still have a long wrapper and will try to shorten it, but it works perfectly well on Windows anyway.

The included pybass/wx_ctrl_phoenix.py file worked with minor modifications: wx has been updated and on Python 3, few changes need to be done (mostly str/byte conversions in Pybass calls). You will find my attached script here, that you can test:

1. Just copy or clone the Pybass library.
2. From within the pybass/pybass directory, paste the libraries from pybass/lib (particularly the DLL files on Windows).
3. Put the script attached to this issue in this directory and run it with no argument (use Python 3.6, no dependency needed except for wxPython).

#3 - 12/01/2018 09:38 PM - Vincent Le Goff

- % Done changed from 0 to 30

I've reduced the code of the wrapper to a test that doesn't take more than 20 lines. It's really a small bridge between Python code and C code, and (to me) it doesn't look that nice, so I'll hide it. The idea is that, after initializing the library, there's a function to call, which returns handles on the audio files. This handle (a big ugly int) is the way to reach the library for further processing. One of the functions is to play this handle. Sounds simple, but there are still some unanswered questions.

For the time being, you can pull the pybass branch of CocoMUD from Github. It contains a pybass directory, which is pretty sparse: a pybass.py file which is the authentic library, and a test.py module which contains my small test. From the root directory, paste a sound (sound.ogg file) and execute the test script to play the 15 first seconds of it:

pipenv run python pybass/test.py

This very basic test performs a play and pauses (using time.sleep) the program, because it's completely asynchronous. Other methods, to change volume, pause and stop files (as handles) are also supported.

#4 - 12/02/2018 06:47 PM - Vincent Le Goff

- Status changed from In Progress to Closed

- Assignee set to Vincent Le Goff

- % Done changed from 30 to 100

A very basic wrapper has been set up to play sounds. It's still very basic and needs improvement, but I close this issue as the goal is reached.

Files

wx_ctrl_phoenix.py	21.6 KB	12/01/2018	Vincent Le Goff