Audacious - OLD, PLEASE USE GITHUB DISCUSSIONS/ISSUES - Feature #611

Preserve shuffle history on exit

January 04, 2016 20:45 - Audy Murphy

Status:	Closed	**Start date:**	January 04, 2016
Priority:	Minor	**Due date:**	
Assignee:		**% Done:**	100%
Category:	libaudcore	**Estimated time:**	0.00 hour
Target version:	3.10		
Affects version:	3.7.1		

Description

I have a moderately sized songs library, and I use random play extensively. By its very nature random is just that, random. The likelihood that a song replays several times within a given time interval is unpredictable.
My wish would be for the ability to select say: "Not replay the last <insert number of songs>

I have selected version 3.7.2 as the affected version, simply because this is the version I am currently using.

History

#1 - January 06, 2016 06:40 - John Lindgren

- Affects version 3.7.1 added

- Affects version deleted (3.7.2)

We keep track of the songs that have been played and don't repeat them as long as Audacious is running, but once you close the program, that information is lost. As a workaround, you can randomize the playlist and then play it in (now random) order, without shuffle enabled.

3.7.2 has not been released yet.

#2 - January 06, 2016 17:55 - Audy Murphy

Thanks, I'll do that.

#3 - July 10, 2016 06:33 - John Lindgren

- Subject changed from Pseudo random playback to Preserve shuffle history on exit

#4 - July 10, 2016 06:34 - John Lindgren

- Category set to libaudcore

#5 - March 30, 2017 14:23 - Nadav Har'El

There's another way to implement the desired feature, which I think will be very easy and even more useful:

The idea is add, unrelated to the current "shuffle flag", also an operation on playlist, to "shuffle" (randomly reorder) it. When I "shuffle" a specific playlist, the tracks in it will get permanently reordered in a random way. This ensures that as I play through that playlist, I do not get played the same songs again, and all we need to save on exit is the current position in the (reordered) playlist.

Another benefit of shuffling a playlist (instead of a shuffle flag) is that it allows some playlist to be shuffled, while others are not. I think this is an important feature - for some types of playlists (such a popular music) I usually want to shuffle them, but for other playlists, e.g., classical music, I often want to play by order. The current scheme where we have a global "shuffle flag" is annoying for this.

Finally, yet another benefit of shuffling (reordering) the playlist: when you look at the play list, you can immediately see what songs will play next. With the current shuffle, you know which song is currently playing, but can't see which songs played earlier or will play later.

So I think my suggestion is the best of all worlds :-)

#6 - March 30, 2017 14:45 - John Lindgren

John Lindgren wrote:

> As a workaround, you can randomize the playlist and then play it in (now random) order, without shuffle enabled.

Nadav Har'El wrote:

> The idea is add, unrelated to the current "shuffle flag", also an operation on playlist, to "shuffle" (randomly reorder) it. When I "shuffle" a specific playlist, the tracks in it will get permanently reordered in a random way. This ensures that as I play through that playlist, I do not get played the same songs again, and all we need to save on exit is the current position in the (reordered) playlist.

That feature has existed for years. Did you even read the comments above yours?

#7 - March 31, 2017 07:42 - Nadav Har'El

Sorry, I didn't notice this half-sentence in one of the comments, and I definitely did look for it in the software - I even looked several times in all the submenus and din't find it. But now I see it does exit - as Playlist ~~> Sort -> Random Order. I guess it's not conspicuous enough :~~(

#8 - April 15, 2017 17:30 - me 20 hours and

I'd like to have this behaviour (C pseudocode):

```
struct song {
    […]
    float near_play = 0.0;
};

// When playing the first song, use rand() % songcnt, not this function.
int select_next_song(struct song songs[], int songcnt,
    int previous_song, float ord)
{
    assert(ord >= 0.0 && ord < 1.0);

    for (int i = 0; i < songcnt; ++i)
        songs[i].near_play *= ord;
    songs[previous_song].near_play = 1.0;

    float all_near = 0.0;
    for (int i = 0; i < songcnt; ++i)
        all_near += 1.0 - songs[i].near_play;

    float rv = rand() / RAND_MAX;
    float p = 0.0;
    for (int i = 0; i < songcnt - 1; ++i) {
        float rarity = 1.0 - songs[i].near_play;
        p += rarity;
        if (rv <= p)
            return i;
    }
```

```
    return songcnt - 1;
}
```

The user can configure the value of ord (similar to volume adjustment).
If it's 0, the rarity for the previous song is 0 and the other songs' rarity is the same for each song: 1 / (songcnt - 1) (usual shuffle I think).
If it is set near to 1.0, it will more likely select the song which hasn't been played for a long time.
Example: ord = 0.5, songs: A, B and C

- play B (33.3̅ % chance)
- A: 0, B: 1, C: 0
- A 50%, B 0%, C 50%
- play C
- A: 0, B: 0.5, C: 1
- A 66.6̅ , B 33.3̅ %, C 0
- play B
- A: 0, B: 1, C: 0.5
- A 66.6̅ %, B 0 %, C 33.3̅ %
  […]

#9 - September 07, 2017 06:13 - John Lindgren

- Status changed from New to Closed

- Target version set to 3.10

- % Done changed from 0 to 100