Audacious - OLD, PLEASE USE GITHUB DISCUSSIONS/ISSUES - Bug #147

Audacious opens in multiple instances under Windows

July 08, 2012 23:38 - Jean-Pierre Vidal

Status:	Closed	**Start date:**	July 08, 2012
Priority:	Minor	**Due date:**	
Assignee:		**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	3.5		
Affects version:	3.2.4		

Description

If Audacious is open and running and I open another file externally from audacious (windows explorer), the actual behaviour is to open a new Audacious instance and that results into playing two songs at the same time.
Normal behaviour should be to use the same instance and: replace actual play with the new one or enque the new one.

History

#1 - July 08, 2012 23:39 - Jean-Pierre Vidal

I'm using Audacious for Windows in Windows 7 by the way...

#2 - July 10, 2012 01:13 - John Lindgren

Right now our interprocess communication is implemented using D-Bus, which isn't available in Windows. I'd like to hear suggestions for a cross-platform solution.

#3 - July 10, 2012 01:14 - John Lindgren

By the way, drag-and-drop **does** work in Windows, so I suggest that as an alternative for the present.

#4 - July 13, 2012 14:59 - Carlo Bramini

I have DBUS up and running on Windows, Audacious detects it at configure time, but I do not recommend it because in my opinion it has not a big pratical usage (I must launch myself dbus-daemon.exe first in a separate session before Audacious).
Using Audacious without the daemon server running does not seem to cause problems, in this worst case it seems to work as it worked until now.

Technically speaking, I believe it would not be a big deal to write 20 lines of code for supporting this stuff natively, but if I can say again my opinion:
1) leave 3.2.x as is, by using DBUS
2) on 3.3.x and newer, use the GtkApplication or GApplication classes, which they have been implemented in GTK+3.

#5 - July 30, 2012 05:56 - John Lindgren

See #133 for an explanation of why we can't use GtkApplication. I am interested in the "20 lines of code" solution immensely, though. Do you have a working patch (even a rough one)? I am going to try again later this week to get a working build of 3.3 for Windows as a starting point.

#6 - July 30, 2012 05:59 - John Lindgren

Alternatively, we could implement launching dbus-daemon automatically, perhaps? I wasn't aware that it even worked on Windows; I believe it failed to compile the last time I tried.

#7 - October 07, 2012 16:12 - Carlo Bramini

- File test1.zip added

- File test2.zip added

Actually there are different ways for implementing a single-instance applications, some solutions may look better than others or they may be dependant on the version of the operating system installed on the machine.
I would like to suggest a simple mechanism that it has the advantage to work on all versions of Windows (past, present and I hope future), based on named mutex objects.

In the first attachment I did a very simple example that it works as a standalone application and it should explain the trick: if you will try it on Windows, you will get an empty window on screen and, if you will try to launch a new instance, first it will try to restore it if it was minimized and then it will display a messagebox with the text on command line of the second instance. For simplicity, here I assumed that the application will share plain ascii, but using unicode instead is identical.

Now, let's examine the source: as you can probably imagine, the so called "20 lines of code" are pretty much located into the IsAppRunning() function, the handling of the mutex object in the main loop and the WM_COPYDATA message in the message handler.

With a simplified explanation, a mutex object is created first with ownership requested and three things may happen:
1) The object was not existing and so it will be created.
2) The object exists but ownership cannot be acquired.
3) The object exists and ownership is acquired because it was released by the primary instance.

If we fall in the case (1), then this is surely the primary instance, so we can continue our processing.
If we fall in the case (2), then this is a strange error condition since we are trying to run a secondary instance while the primary instance is still initializing, so don't worry.
IF we fall in the case (3), then we can communicate to the primary instance that the user is trying to run a secondary instance: this may not be really required for solving this bug (we could simply exit here), but it is useful in many situations... for example, in the case of a player like Audacious, the user may double click on an audio file in Windows Explorer and, if that file type is registered to our application with an action verb (like "open" or "play"), the player could change "on the fly" the file it is playing.

The "communication" with the primary is done by sending WM_COPYDATA message: the secondary instance tries to find a known window class and it sends directly some informations.

Now, how to integrate this stuff in Audacious or any other GTK+ application without too much troubles?

I gave a quick look to the sources of GTK+ and unfortunately I discovered that there is no way to get a separation of window classes (all applications have the same class), but this is not a problem... perhaps it even simplifies our job since we can do it directly by ourself, by creating a "shadow" window, not visible on the desktop screen(s) but consistent in the system.
For not touching parts of the current code, in my opinion all this stuff can be implemented with a threaded object that it will work transparently.

The programming interface could be something like this:

```
/* Starts the single instance mechanism.  * It must be called as soon as the application starts.  * Returns TRUE if the application must exit.
*/
int single_instance_start(void);

/* Registers a callback that it will be called  * as soon as a secondary instance will be launched.  * It will receive the command line sent on the secondary instance
*/
void single_instance_register(void *callback);

/* Stops the single instance mechanism */
void single_instance_stop(void);

/* callback type for function single_instance_register() */
typedef void (*single_instance_callback)(const char *);
```

In the second attachment, I re-implemented the sample in this way and with a console based application (like audacious is).
The callback will receive a pointer to a text string, but this could be eventually changed to the best type to be used for a GTK+ application.

As I said, this stuff is very similar to previous code, with the difference to be shared in a different thread for not making troubles to existing GTK+ code.

That's all, friends, with an explanation longer than the code itself...
I hope this stuff could be useful for you.

Sincerely,

Carlo Bramini.

<OT>
Sorry if never replied, but I did not know that you were interested...
For pure case, I discovered that there were some replies to this bug, shouldn't I receive a notification when a change is made to a bug that I commented?
<\OT>

#8 - September 15, 2013 08:56 - John Lindgren

- Status changed from New to Closed

- Target version set to 3.5

- % Done changed from 0 to 100

This is finally fixed in Git by switching to GDBus, which compiles and amazingly "just works" on Windows.

Files

test1.zip	6.51 KB	October 07, 2012	Carlo Bramini
test2.zip	8.51 KB	October 07, 2012	Carlo Bramini