Ruby 1.8 - Bug #5049

[ext/tk] Tk creating double MessageBox's

07/19/2011 04:38 AM - jonforums (Jon Forums)

Status:	Open
Priority:	Normal
Assignee:	
Target version:	Ruby 1.8.7
ruby -v:	ruby 1.8.7 (2011-06-30 patchlevel 352) [i386-mingw32]

Description

When running the following simple test on 1.8.7-p352 built with TDM-1 GCC 4.5.2 and the RubyInstaller recipes, I get two MessageBox's. One empty, one with the message.

ruby -rtk -e "Tk.messageBox :message => 'Hello Ruby Tk'"

Additional working and non-working samples => http://pastie.org/2233258

This does not occur when using MinGW versions of ruby 1.9.2p290 (2011-07-09 revision 32478) [i386-mingw32] or ruby 1.9.4dev (2011-07-18 trunk 32578) [i386-mingw32].

Jon

History

#1 - 07/20/2011 12:26 AM - jonforums (Jon Forums)

Adding email thread as it's missing from Redmine's issue. Comments below.

> > When running the following simple test on 1.8.7-p352 built with TDM-1 GCC 4.5.2 and the RubyInstaller recipes, I get two MessageBox's. One empty, one with the message.
> >
> > ruby -rtk -e "Tk.messageBox :message => 'Hello Ruby Tk'"
>
> It is not a bug. The empty window is a root window of Tk.
> If you want to show a message box only, please call "Tk.root.withdraw"
> at first. Then, a root window will be hidden.
>
> $ ruby -rtk -e "Tk.root.withdraw; Tk.messageBox :message => 'Hello Ruby Tk'"

Your answer makes technical sense, but from a user perspective requiring this extra code to display a message box seems both (arguably) unintuitive and an unnecessary implementation detail.

Are you also saying the root Tk window shouldn't appear in 1.8.7 if the message box is part of a more real-world script? For example, this works as expected:

```
require 'tk'
require 'tkextlib/tile'

root = TkRoot.new
button = Tk::Tile::TButton.new(root) { text 'Hello Ruby Tk!' }.grid

Tk.messageBox :message => 'preliminary'

Tk.mainloop
```

If this 1.8.7 behavior is a corner case that will *never* happen in real-world Ruby Tk apps but only in simple example code, then I agree with you that this isn't a bug and the issue can be closed.

> > This does not occur when using MinGW versions of ruby 1.9.2p290 (2011-07-09 revision 32478) [i386-mingw32] or ruby 1.9.4dev (2011-07-18 trunk 32578) [i386-mingw32].
>
> It depends on the difference between ruby-1.8 and 1.9.
> Although I omit the detail of the reason, Ruby/Tk on recent ruby-1.9

> hides a root widget before calling Tk.mainloop.
> Usually, it has no problem. Because, Tk requires an eventloop to treat
> widnow events. But Dialog methods (e.g. Tk.messageBox) makes its own
> eventloop. So, ruby-1.8 shows a root widget too, but ruby-1.9 doesn't.

Are there any technical limitations in 1.8 that prevent Tk from behaving consistently with 1.9 with respect to Dialog methods and event loops?

Why should 1.8 and 1.9 Dialog methods behave differently from a Ruby code perspective?

Jon

#2 - 07/21/2011 03:33 AM - jonforums (Jon Forums)

I don't know why redmine isn't capturing your answers. They're very helpful so I'm manually inserting them.

> > Your answer makes technical sense, but from a user perspective requiring this extra code to display a message box seems both (arguably) unintuitive and an unnecessary implementation detail.
>
> I said that you didn't get two messagebox windows but got a root window
> and a messagebox window.

Understood and I was imprecise with my language. I meant to convey the idea that the two windows are unexpected, and the extra code required to suppress the root window in 1.8.7 wasn't what I would expect to have to do to show just a messagebox window.

> Maybe, you are misunderstanding about a root window. On Tk, a root
> window is automatically created without calling a constructer method
> (without TkRoot.new) when initializing a Tcl/Tk interpreter and is
> shown as default. If you want to hide a root window, it is a matter of
> course that you must do a method call to hide it.
> I thought that you wanted to show a messagebox window only without a
> root window. So, I wrote a solution to hide a root window.

Understood and thank you for the explanation.

> > Are there any technical limitations in 1.8 that prevent Tk from behaving consistently with 1.9 with respect to Dialog methods and event loops?
> >
> > Why should 1.8 and 1.9 Dialog methods behave differently from a Ruby code perspective?
>
> If changing behavior on 1.8, a compatibility is lost.
> The difference between 1.8 and 1.9 depends on nativethreads.
> 1.8 runs on only one nativethread, but each of threads on 1.9 runs on
> each nativethread.
> Tk widgets will work on the only nativethread on which Tk interpreter
> was initialized. So, on 1.9, Tk library must make a ruby's thread in
> which Tk interpreter is initialized and an eventloop is running.
> Otherwise, Tk.mainloop works on a main thread only (on some
> environments, there is the limitation). Then, "Thread.new{Tk.mainloop}"
> doesn't work.
> Well, when starting an eventloop at "require 'tk'", an empty root
> window is shown at same time. And users will see the steps creating GUI.
> I think that is bad. It is NOT compatible between 1.8 and 1.9 as behavior.
> I did a hack to hide a root window before calling Tk.mainloop.

If I understand you correctly, since in 1.8 all the Tk bootstrap code is running on a single native thread, 1.8 is unable (in some cases) to hide the root window before calling Tk.mainloop because Tk is performing all it's work on that single native thread. But on 1.9 you're able to hide the root window because you can do some of the bootstrap code on a different thread than the thread running the Tk interpreter and eventloop, correct?

> Almost all cases, it works properly. However, as you see, it looses
> compatibility on some minor cases.
> I think the incompatibility will be acceptable, because, without
> Tk.mainloop, it is natural to call withdraw method before calling a
> dialog method (it makes sub-eventloop for it self). 1.8 users will do
> that. And it works properly on 1.9 too.

If your hack works for all the "normal" cases and just loses compatibility for some minor cases like my toy example, I completely agree with you. I'll just have to remember this detail regarding using the withdraw method before calling a dialog method on 1.8.

Is there anywhere in the Tk extension code that makes sense to document this caveat so that it doesn't exist only in this thread?

Of course, maybe it doesn't matter since we're all immediately moving to 1.9 anyway ;)

> Can it be an answer for your question?

Yes, and thank you again for taking the time to provide the details.

Please feel free to close this issue.

Jon