A note on Linger

Ken Nakatani
Last update: June. 2017
Back home

Preface

Linger is a simple sentence presentation program written by Doug Rohde during his post-doc years at Ted Gibson's lab at MIT (of which I was also a member at that time). Its latest version is v2.94, updated in 2003. So it's a very old program! However, because it is a relatively simple Tcl/Tk script specifically designed for self-paced reading tasks (as well as auto-paced reading and listening tasks), which runs on various Unix-based platforms (including Windows with Cygwin), many researchers still use it.

However, since it's old, it may not run on the latest computers "as is". Inside the researchers' communities where Linger is being used on a regular basis, they hand down the tips so there are no problems. However, I think it is also a good idea to share the knowledge beyond the communities, over the internet. I felt that way when a friend of mine found that Linger would record rediculously huge reading times when used under Mac 10.9 (Mavericks) or higher. I eventually found out a workaround, but as a lay person in programming, it took me hours to find out the solution. I couldn't find information specific to this Linger problem on the web, so I had to look for relevant information in Tcl/Tk programming in general, which was the reason why I had to spend so much time resolving the problem. (To jump to the discussion of the problem with Mavericks or higher, click here.)

Yes, so I really wished at that time that I could have found Linger-specific trouble-shooting on the web. I

guess some people may still be looking for it. That's the reason why I decided to leave a note here.

However, I do not have any intention to provide a comprehensive guide. I am just a psycholinguist that has little knowledge in programming, especially in Tcl/Tk. I just leave here what I know. Period. I'm using a Mac, and I won't provide any information regarding other platforms, nor try to find solutions for anybody unless the problem is relevant to my research environment. So please do not ask me how to do X in Windows or Linux. Sorry but I lack skills. On the other hand, comments are very welcome. I believe that this page contains inaccuracies and errors. Oh, so please don't blame me if anything wrong happens by following suggetions provided here.

This is not a replacement for the official manual

Doug's own official manual contains detailed and very useful information. Don't miss it.

Some notes on the Linger startup with a Mac

Downloading linger.tgz

In the official installation guide, downloading Tcl/Tk Aqua is suggested, but it's not necessary under newer versions (dunno since which one) of OS X. All you need is to download the linger.tgz package. (Newer versions of OS X do not come with StuffIt Expander, but .tgz files can be unpacked by a double-click, without StuffIt Expander.)

The Linger folder

When linger.tgz is unpacked, a folder called Linger appears. This folder itself can be placed anywhere.

The directory (folder) structure **inside** the Linger folder, however, should not be altered.

linger.tcl

The actual Linger script is linger.tcl, which is a plain text file written in Tcl/Tk language. linger.tcl is executed through a program called wish (named after "window shell"), which comes with OS X by default (located in /usr/bin/). wish is a Tcl language interpreter, and it reads through the commands written in linger.tcl, realizing a GUI interface that interacts with the signals from the keyboard a human operates.

To run linger.tcl, you may use Terminal.app ("Terminal"). In Terminal, go ("cd") to the directory where the linger.tcl is located, and simply type "wish linger.tcl". This simple command boots Linger.

(BUT some problems will be incurred -- workarounds are explained below.)

Linger launcher

The Linger folder also contains an application called Linger (without an extension). This is an applescript application that is supposed to launch linger.tcl by a simple double-click, not though Terminal. However, since this application is created by an old, PowerPC-based applescript editor, it won't open in newer versions of OS X (from OS X 10.7 Lion and onwards, where Rosetta is unusable).

If you still wish to have a handy launcher, you can create one, using AppleScript Editor that comes with your newer machine. There are many ways of implementing the launch in an applescript, but one of the simplest ways is to write the following single line in the AppleScript Editor:

```
do shell script "wish " & POSIX path of
((path to me as text) & "::") &
"/linger.tcl"
```

If you are using Mavericks or higher, you might want to run linger.tcl with an older version of wish (v8.4), for the reason I will mention later. In that case, the AppleScript command would be:

```
do shell script "wish8.4 " & POSIX path of
((path to me as text) & "::") &
"/linger.tcl"
```

Then save or export(*) it as an "application" in the same directory as linger.tcl is in. This application itself must be in this directory, but an alias of it can be placed anywhere. You can now hire an assistant who does not have an experience in Terminal shell.

> (*)With macOS Sierra 10.12.5, saving the script as an application would lead to an error-prone launcher, for which I don't know the reason. Exporting it as an application works fine.

For your convenience, here is the AppleScript application described above (the wish8.4 version): Linger.app.zip

These executable scripts are also editable. To edit the script, right-click on the script and select "パッケージの内容を表示" ("Show the content of the package" or something like that in English), then browse through Contents > Resources > Scripts.

How to run a self-paced reading experiment in Japanese using Linger on a Mac

I am a Japanese native speaker working on Japanese, so here's what I know about using Linger in Japanese.

How to use Japanese characters

Encoding

Whenever you set up a new experiment, you need to create a new folder right under the Experiments

folder (let us call it an experiment folder, with a non-capitalized e). In other words, Linger recognizes each experiment folder (but not sub-folders) in the Experiments folder as an independent experiment.

In order to use Japanese characters, you need to specify the encoding in the file called "preferences" located in each experiment folder. In the offical manual, euc-jp or jis0208 is recommended for Japanese, but I would recommend utf-8. All the files in an experiment folder should be saved using the encoding as specified in the "preferences" file. In order to specify the encoding, include the following line in the "preferences" file:

```
set LangEncoding utf-8
```

Setting up the font name in Japanese

As the official manual states, you cannot use Japanese characters in the "preferences" file where the encoding is specified, because the specified encoding is effective only after the "preferences" file is read by the program. If you want to specify the font name in Japanese, you need to:

1. Include the following line in the "preferences" file:
   `encSource preferences-jp`
2. Then create a file named "preferences-jp" and specify the font names like the following:
   ```
   set TextFont "-family {ヒラギノ角ゴ Pro W3} -size -18"
   set BigFont "-family {ヒラギノ角ゴ Pro W3} -size -24"
   set ContinueFont "-family {ヒラギノ角ゴ Pro W3} -size -18"
   ```

You can also set up the Japanese versions of various phrases in this "preferences-jp" file (or alternatively, in the "introduction" file), such as the following:

```
set WrongAnswer "残念！ 不正解です"
set PressKey "次に進むにはスペース・バーを押してください"
set QuestionKeys "「はい」には ¥"[string
```

```
toupper $YesKey]¥" を、「いいえ」には ¥"[string
toupper $NoKey]¥" を押してください。"
```

To be spaced or not to be spaced

In linger, regions are usually masked and presented with spaces. However, Japanese orthography does not utilize spaces. If you want to eliminate spaces in the presentation, you can do so by separating regions with | in the "items" file, in place of spaces.

An issue with Mavericks or higher

As mentioned at the beginning of this document, Linger has an issue with MacOS 10.9 (Mavericks) or higher: it would record rediculously huge reading times. I am not familiar with Tcl/Tk nor am I a computer expert. So I am unable to provide precise explanation on the cause of this issue, but at least, I figured it out that this problem is related to the issues on clock resolution.

Solution 1: Run linger.tcl with older wish

A quick and dirty workaround is to run linger.tcl with an older version of wish (special thanks to Michinao Matsui sensei for bringing this up). Fortunately, Mavericks (and newer OSs including Sierra) comes with several versions of wish (located in /usr/bin/): `wish`, `wish8.4`, and `wish8.5`. linger.tcl in its original form works fine with wish8.4, so cd to the linger directory and type:

```
wish8.4 ./linger.tcl
```

Then the problem goes away.

Dan Grodner has suggested another method (thanks Dan!), by modifying line 3 of linger.tcl, which reads:

```
exec wish "$0" -- "$@"
```

This could be changed into:

```
exec wish8.4 "$0" -- "$@"
```

Then linger.tcl can be executed by wish8.4. (Note that this doesn't work if you want to use an AppleScript application to execute linger.tcl.)

Solution 1b: Run linger.tcl with older wish

Solution 2: Modify linger.tcl

If you want to run linger.tcl with the newer version of wish installed in Mavericks, you need to modify linger.tcl (although there is one reason why we might want to stick with wish 8.4, which is mentioned in the next section). This turns out relatively easy (although it took a while for me to figure it out). The problem resides in the following command, which is the most crucial in recording the reaction times:

```
clock clicks
```

According to the manual of Tcl (https://www.tcl.tk/man/tcl/TclCmd/clock.htm), this command is supposed to return "a high-resolution time value as a system-dependent integer value." Furthermore, the manual states that "[t]he unit of the value is system-dependent but should be the highest resolution clock available on the system such as a CPU cycle counter." I found that this `clock click` command returns a figure a thousand times as large under the default wish in Mavericks. From what can be inferred from the manual of Tcl, the problem appears to be incurred from the resolution of the CPU; however, from the fact that different figures are returned by different versions of wish with the same CPU, it can be concluded that the wish (Tcl/Tk) versions also matter.

In any case, here's what you could do if you want to run linger.tcl with the default wish that comes with Mavericks:

- Replace all the instances of
  ```
  clock clicks
  ```
  in linger.tcl with
  ```
  clock milliseconds
  ```

Note that this `clock milliseconds` command does not work in wish 8.4. If you want to modify linger.tcl so as to work properly in both older and newer versions of wish, you can alternatively:

- Replace all the instances of
  ```
  clock clicks
  ```
  in linger.tcl with
  ```
  clock clicks -milliseconds
  ```

This latter command, however, is "obsolete" according to the documentation of the latest wish, so there is a possibility that this will stop functioning in the future. For now, this works fine.

Full screen mode

Another issue with running linger.tcl under Mavericks is that if it is run with the default (newer) wish in Mavericks, the window, which is supposed to be in the fullscreen mode, is drawn in a tiny size. In linger.tcl, the size of the (fullscreen) window is calculated with the `winfo` command, which is part of the `buildMainWindow` procedure (subroutine):

```
proc buildMainWindow {} {
global BgColor PadX PadY GoKey KillKey
RightShift tcl_platform
set width [expr [winfo screenwidth .] +
$PadX * 2]
set height [expr [winfo screenheight .] +
$PadY * 2]
wm geometry .
${width}x${height}+-${PadX}+-${PadY}

...snip...
}
```

This does not seem to work with the default wish in Mavericks. A very simple modification, however, will solve the problem: simply comment out the original calculation part and use the `-fullscreen` option for `wm`.

```
proc buildMainWindow {} {
global BgColor PadX PadY GoKey KillKey
RightShift tcl_platform
# set width [expr [winfo screenwidth .] +
$PadX * 2]
```

```
# set height [expr [winfo screenheight .] +
$PadY * 2]
# wm geometry .
${width}x${height}+-${PadX}+-${PadY}
wm attributes . -fullscreen 1

...snip...
}
```

This works just fine with the newer wish that comes with Mavericks, but it works even better with older wish 8.4: a "real" fullscreen is fully realized with wish 8.4, while it is not perfect in the newer wish because the upper window frame remains visible. So I think that using wish 8.4 is probably the best option for now.

(There used to be an error in this section, which came to my attention thanks to Dan Grodner. It has been fixed. Thanks a million, Dan!)

[End of document]

If you find errors or if you know of better solutions, let me know via here. Please do not ask questions since I'm no expert!