BREYDON's

turntab

turner of tablet

Version: 0.4.2 extremely α

October, 2020

—

K.F. Breydon

www.breydon.id.au

Table of Contents

1 Acknowledgement of Country

Acknowledgement of Country[1] ≡

```
{turntab has been written on Boon Wurrung land.
 I pay my respects to the Boon Wurrung people,
 especially their Elders, and to all Indigineous
 communities within whose lands, waters or skies
 this software is read, adapted or run.

                          \342\237\244\342\237\245
}
```

This macro is invoked in definition 22.

`\342\237\244` and `\342\237\245` are C octal escaped UTF-8 representations of modal operator symbols "was always" and "will always be" — used here in allusion to the phrase *Always was and always will be Aboriginal land.*

2 Overview

2.1 Cataloguing

Reference, in the script, to version number, update dates and so on draws on the following few macros. These macros haven't been connected to their TEX equivalents — I still edit footline and coverpage mentions separately.

Version[2]**M** ≡
```
{v0.4.2 extremely alpha}
```
This macro is invoked in definitions 9 and 22.

Dates[3]**M** ≡
```
{Written October 2020.
 }
```
This macro is invoked in definition 22.

Credits[4]**M** ≡
```
{Author: Kermie BREYDON (they/them/their).
 }
```
This macro is invoked in definition 22.

URL for turntab[5]**M** ≡
```
{https://www.breydon.id.au/puting/turntab/}
```
This macro is invoked in definitions 9 and 22.

Should this web address fall out of date, kindly change it to a freely available record held by a public library or similar. (If nowhere has `turntab` these days, consider donating this copy you are reading). Thanks :)

Copyright status[6]**M** ≡
```
{This software is in the public domain. It is
 free for use by anyone and for any purpose. The
 software is offered "as is", without warranty of
 any kind.
 }
```
This macro is invoked in definitions 7 and 22.

Gonna pop that last passage into in a file named `COPYING`, for those folks relying on automated searches for one.

COPYING[7] ≡
```
{Copyright status[6]
 }
```
This macro is attached to an output file.

I'd rather `COPYING` consist of a Creative Commons CC0 statement, but for a regrettable dependence of that notice on "he or she" phrasing (as of CC0 1.0).[1] Alternative standardised statements that boil down to the above — public domain dedication, fallback licence, warrantilessness — don't seem to be worth their disadvantages. I think I might have borrowed the "free for use by anyone and for any purpose" phrasing from the SQLite website.[2] While I intend that middle statement as a fallback licence, I may too be open to issuing individual licences, for umpteen dollary-doos, in special cases.

[1] http://creativecommons.org/publicdomain/zero/1.0/
[2] https://www.sqlite.org

2.2 What is turntab?

README[8] ≡
```
{turntab is a programme,
 which is in these files:

 README describes the files.

 turntab.sh is the file that computers use to run turntab.

 turntab.pdf is a book for people to read turntab.

 COPYING says that turntab belongs to everybody.

 turntab.tex is for putting turntab into other publications.

 turntab.fw is what the files are made out of.
 }
```
This macro is attached to an output file.

Introduce self[9] ≡
```
{#!/usr/bin/env sh
 #      turntab Version[2]
 #         written by Kermie BREYDON
 #          on Boon Wurrung country
 #                 in 2020
 #           for the public domain
 #
 # URL for turntab[5]

 printf "Hello, turntab speaking. I get the orientations
 of tablet input and display to match each other.\n"
 }
```
This macro is invoked in definition 10.

If you'll excuse a few paragraphs' jargon, `turntab` is a work of literate programming. Its executable component — `turntab.sh` — is a script designed to be run by POSIXy shells, either at the command-line or when triggered by a shortcut. The documentation (which you are reading now) and the script are both derived from the same source file — `turntab.fw` — which follows the formatting conventions of the FunnelWeb macro preprocessor[3] (but eschews FunnelWeb's extremely 1990s HTML dialect for a slightly heavier investment in the TEX typesetting system[4] than is quite standard for FunnelWeb).

I wrote `turntab` as an accessibility aid to using the X Window System[5] on a touchscreen device. `turntab`'s script probes and very slightly manipulates `xsetwacom` (a utility for the `wacom` input driver for X)[6] and `xrandr` (an interface for the RandR extension to X), to align their respective rotation settings. Perhaps most distinctively, `turntab` also co-ordinates the system console and virtual TTYs with that graphical stuff. `turntab` presumes an interface for this last part at `/sys/class/graphics/fbcon/rotate_all`; some systems might require a different approach, or lack the capacity. Note that the method given here for

[3] http://www.ross.net/funnelweb/
[4] https://tug.org/
[5] http://www.x.org/
[6] http://linuxwacom.sourceforge.net

adjusting frame-buffered consoles depends on superuser privileges — so do take appropriate precaution if you are out to exploit this particular function.

At version 0.4.2 extremely α, `turntab` is likely to behave over-exuberantly in systems that involve multiple displays and/or multiple `wacom`-compatible tablet surfaces. I expect it could adjust more of them than necessary, and in some cases take the wrong readings. It is also likely to mess up around some custom RandR efforts to compensate for projector distortion, and so on. But in a standalone unit — for communication boards or basic tablet computing — I find it effective, versatile, and uncommonly simple to implement. May you benefit from this work, in some way, too.

2.3 Features

`turntab.sh` can perform the following tasks. Their definition numbers are indicated in square brackets, for ease of cross-reference.

turntab.sh[10] ≡
{*Introduce self*[9]
Have functions[11]
Follow command line instructions[12]
Finish[23]
}
This macro is attached to an output file.

"Have functions"? No, it's not throwing a party; it is being ready to do a range of things in various sequences. These things:

Have functions[11] ≡
{`# Functions!`

Cite origins[22]
Offer help[21]
Translate terminology[16]
Increment rotation goals[17]
Suss out orientation of graphical display[13]
Find all the forms of tablet input[14]
Suss out orientation of tablet input[15]
Orient graphical display[18]
Orient tablet input[19]
Orient frame-buffer consoles[20]}
This macro is invoked in definition 10.

3 Follow command line instructions

`turntab` commands aim to suit a few demographics, with contrasting needs.

People using stenography, text-to-speech or (some) predictive-text utilities will typically find it easiest to issue a command in ordinary words — like `turntab help`. The word "turntab" will of course still need to be added to lexicon of the appropriate software. Fuller support of text-to-speech users might require an interactive mode, something `turntab` 0.4.2 extremely α lacks.

Serial typists (entering one letter at a time), are sometimes better served by a concise command without punctuation — like `turntab h`. This group might like copies of `turntab.sh` reassigned to a shorter name; but I'll leave that step to you.

Sticklers for tradition go in for symbols. One or two hyphens and a full word — as in `turntab --help` — facillitates the writing of intelligible scripts. One hyphen and initial(s) offers consistency on a wider range of operating systems (and for some users is easier to input) — as with `turntab -h`.

Obviously the approach outlined below is very English Language, but it's easy to bung in additional terms. Certain phrases might clash with instances of shorthands like `*cw`, necessitating replacement with `--cw|cw|-cw` patterns (in the macros `Following command line instructions` and `Translate terminology`).

Follow command line instructions[12] ≡

```
{# Making sense of commands!

 case "$1" in
   ""|*help|h|-h)
     printf "\nThings you can write after $0:\n"
     help
     ;;
   *version|v|-v)
     software
     ;;
   *increment|*cycle|c|-c)
     sussxoutputori
     incrementgoals
     orientxoutput
     orientxinput
     orientfbc
     ;;
   *input|i|-i)
     sussxoutputori
     orientxinput
     orientfbc
     ;;
   *output|o|-o)
     sussxinputori
     orientxoutput
     orientfbc
     ;;
   *0|*1|*2|*3|*normal|*right|*inverted|*left|*none|*cw|*half|*ccw)
     goal="$1"
     translate
     orientxoutput
     orientxinput
     orientfbc
     ;;
```

```
  *x)
    case "$2" in
      *increment|*cycle|c|-c)
        sussxoutputori
        incrementgoals
        orientxoutput
        orientxinput
        ;;
      ""|*input|i|-i)
        sussxoutputori
        orientxinput
        ;;
      *output|o|-o)
        sussxinputori
        orientxoutput
        ;;
      *0|*1|*2|*3|*normal|*right|*inverted|*left|*none|*cw|*half|*ccw)
        goal="$2"
        translate
        orientxoutput
        orientxinput
        ;;
      *)
        printf "Sorry, do what?"
        ;;
    esac
    ;;
  *)
    printf "
Sorry, I do not know what you mean by $1.
Here are some other things we could try next:\n"
    help
    ;;
esac
}
```

This macro is invoked in definition 10.

There have got to be more efficient ways of allowing X-only behaviour. We could check for an `x` on beginning an `orientfbc` step — skipping the rest of the step should there be an `x`. Problem is, I don't know how to neatly ask a shell script to follow the same procedure regardless of whether a command-line option is tacked on in the 1st or 2nd position. Shame. Maybe one day.

4 Figure things out

4.1 Suss out orientation of graphical display

Often, our reference will be the orientation of the main graphical display.

`xrandr` doesn't seem to allow us to ask after a datum directly. Present orientation is left out of the default query response, and I haven't managed to get `xrandr` to report in detail on fewer than ALLLL OF THE SCREENS at once. With some regret, let us wade into `xrandr --verbose`.

The resulting output does not appear cleanly delineated. (Although maybe it uses some kind of invisible record-separator?) Furthermore, identifying features are inconsistent; there's not always a "`primary`" marker on the entry we want, for example.

Have a squiz at this first five lines of a (pretend) `xrandr --verbose` result:

```
Screen 0:  minimum 8 x 8, current 600 x 400, maximum 12345 x 12345
LVDS1 connected primary 600x400+0+0 (0x2c) normal (normal left inverted right x axis y axis) 120mm x
        Identifier:   0x57
        Timestamp:    54321543
        Subpixel:     horizontal rgb
```

We could take a stab that the first non-"`Screen`" line is practically always the one for us.

We could try something like `|grep "Screen" -v -m 1`, saying invert match (thereby *skip* lines announcing "`Screen 0`" etc) and return only the first result. Left clutching just the first line of the first monitor's data and the vain hope that a whole bunch of things *do* stay consistent, we might go to something like `|awk -F " " "{print $6 }"` or "show us the 6th word of the remaining line".

But if we're gonna be so sketchy anyway, we might as well straight-up lie to `awk`, for improved accuracy and less piping.

In any case, we should see if we got a plausible-sounding result, and if not, bow out.

Suss out orientation of graphical display[13] ≡

```
   {sussxoutputori () {
      printf "\nSussing out display (via xrandr)...\n"
      xoutputori="$(xrandr --verbose | awk -e 'BEGIN {RS ="\\(normal left inverted"} $0 ~
"Screen" {print $NF}')"
      if test "$xoutputori" = "normal" -o "$xoutputori" = "left" -o "$xoutputori" =
"inverted" -o "$xoutputori" = "right"; then
        goal="$xoutputori"
        translate
        printf "OK. I reckon the display is $orientation.\n"
      else
        printf "
    Sorry, but this time I could not tell which way
    the graphical display has been turned.
    (Assuming that there even is one?)

    Please note that I depend on the X Window System
    for graphical matters and am not very clever.\n"
        exit
      fi
    }
    }
```

This macro is invoked in definition 11.

4.2 Find all the forms of tablet input

In practice, `xsetwacom` treats each type of tablet input, or "device", separately — despite the driver's manual claiming that changing one input device's orientation will change them all. One implication of this is that we must identify every form of contact that we wish to reorient. Another implication is that reading the orientation of one device at random does not guarantee the orientation of whatever device(s) the user has observed. For instance, they might be exclusively using the eraser end of a stylus, whereas we might happen to base our assumptions on the "device" for generic touches.

We should be able to automate the referencing of a complete set of Wacom devices, in a single line, surely? `xsetwacom list |sed -E 's/.*id:  //; s/\t.*//'` returns ID numbers that `xsetwacom` will be happy with.

Find all the forms of tablet input[14] ≡

```
{findtab () {
 printf "\nIdentifying forms of tablet input (via xsetwacom)...\n"
 devices="$( xsetwacom list | sed -E 's/.*id: //; s/\t.*//')"
 }
 }
```

This macro is invoked in definition 11.

4.3 Suss out orientation of tablet input

Suss out orientation of tablet input[15] ≡

```
{sussxinputori () {
 findtab
 printf "\nSussing out tablet orientation (via xsetwacom)...\n"
 for device in $devices; do
   xinputori="$(xsetwacom get $device rotate)"
   break
 done
 if test "$xinputori" = "none" -o "$xinputori" = "ccw" -o "$xinputori" = "half" -o
"$xinputori" = "cw"; then
   goal="$xinputori"
   translate
   printf "OK. I reckon the tablet input is $orientation.\n"
 else
   printf "
 Sorry, but this time I could not tell which way
 any tablet input is oriented.

 Please note that the only tablets I handle are
 Wacom hardware in use by the X Window System.\n"
   exit
 fi
 }
 }
```

This macro is invoked in definition 11.

Evidently, we are risking the surprise reading outcome (by only examining one `$device`'s orientation).

5 Change things

5.1 Translate terminology

In order to match orientations, we connect a piece of `xrandr`, `xsetwacom` or `fbcon` terminology with its equivalents. While we are at it, let's express the terms' shared meaning more clearly.

Translate terminology[16] ≡

```
{translate () {
 case "$goal" in
   *normal|*none|*0)
     xoutputori="normal";
     xinputori="none";
     fbcori="0";
     orientation="upright (not rotated)";;
   *left|*ccw|*3)
     xoutputori="left";
     xinputori="ccw";
     fbcori="3";
     orientation="turned on its left";;
   *inverted|*half|*2)
     xoutputori="inverted";
     xinputori="half";
     fbcori="2";
     orientation="upside-down to usual";;
   *right|*cw|*1)
     xoutputori="right";
     xinputori="cw";
     fbcori="1";
     orientation="turned on its right";;
 esac
 }
 }
```

This macro is invoked in definition 11.

5.2 Increment rotation goals

For cycling through orientations, we reassign those meanings.

Increment rotation goals[17] ≡

```
{incrementgoals () {
 case "$goal" in
   normal|none|0)
     xoutputori="right";
     xinputori="cw";
     fbcori="1";
     orientation="turned on its right";;
   right|cw|1)
     xoutputori="inverted";
     xinputori="half";
     fbcori="2";
     orientation="upside-down to usual";;
```

```
      inverted|half|2)
        xoutputori="left";
        xinputori="ccw";
        fbcori="3";
        orientation="turned on its left";;
      left|ccw|3)
        xoutputori="normal";
        xinputori="none";
        fbcori="0";
        orientation="upright (not rotated)";;
    esac
    printf "\nLet's make it $orientation.\n"
    }
    }
```

This macro is invoked in definition 11.

5.3 Orient graphical display

Orient graphical display[18] ≡

```
   {orientxoutput () {
    printf "
    Trying to orient graphical display (via xrandr)...\n"
    xrandr -o $xoutputori
    if test "$(xrandr --verbose | awk -e 'BEGIN {RS ="\\(normal left inverted"} $0 ~
"Screen" {print $NF}')" = "$xoutputori"; then
      printf "Seems OK, from my perspective.\n"
    else
      printf "Sorry, I could not correct the graphical display.\n"
    fi
    }
    }
```

This macro is invoked in definition 11.

This manner of testing does not strike me as terribly efficient.

5.4 Orient tablet input

Of course, `xsetwacom set $device rotate $xinputori` is a task that need only be done once per `$device`. So, at a glance, enclosing it in a `for device in $devices;` loop which is itself inside another `for device in $devices` loop might appear redundant.

The outer loop we don't actually care to repeat. The main point is to see if we seem to be handing `xsetwacom` back the right sort of datum: a likely-sounding id number. As a programme inclined to loose assumption, `turntab` figures that this'll do, this vague spot check.

However, with the knowledge that we cannot necessarily rely on `xsetwacom` to sort the rest out for us, we do need to be prepared to repeat the inner loop should there be more than one id-number in `$devices`.

Orient tablet input[19] ≡

```
   {orientxinput () {
    findtab
    if test "$devices" = ""; then
      printf "Sorry, I can't find any.
```

```
Please note that I can only do so within the
X Window System and for certain Wacom devices.\n"
else
  for device in $devices; do
    if test "$device" -gt 0 -a "$device" -lt 100; then
      printf "Trying to correct tablet orientation (via xsetwacom)...\n"
      for device in $devices; do
        xsetwacom set $device rotate $xinputori
      done
      printf "Done. Hopefully.\n"
    else
      printf "Sorry, this time, I cannot make sense of
the system's response.\n"
    fi
    break
  done
fi
}
}
```

This macro is invoked in definition 11.

5.5 Orient frame-buffer consoles

When successful, the command `echo $fbcori |sudo tee /sys/class/graphics/fbcon/rotate_all` spits out the value of `$fbcori`. By catching that output in `$fbcresult`, we stop a decontextualised number from popping out onto the user interface. If we don't catch a number, we can be confident of failure. Nice.

Orient frame-buffer consoles[20] ≡

```
{orientfbc () {
 printf "
 About to orient the text consoles that use a
 frame-buffer. This requires special permission,
 so you might be asked for a password...\n"
 fbcresult="$(echo $fbcori | sudo tee /sys/class/graphics/fbcon/rotate_all)"
 if test "$fbcresult" = "$fbcori"; then
   printf "OK. I think that worked.\n"
 else
   printf "\nSorry, I could not correct them.\n"
 fi
 }
 }
```

This macro is invoked in definition 11.

6 Supply information without accompanying action

6.1 Offer help

Offer help[21] ≡

```
{help () {
   printf "
   help       Shows this usage list.
   version    Shows the current software version.

   input      Makes the tablet input and the
              teletype-style consoles that use a
              frame-buffer (FBTTYs) match the
              graphical display's orientation.

   output     Orients the graphical display and
              FBTTYs to match the tablet input.

   increment  Corrects tablet input and FBTTYs to
              match the graphical display, then
              turns them all 90 degrees clockwise.

   [o]        Sets all frame-buffer consoles,
              graphical display and tablet input
              to the [o] orientation.

   x          Affects only the graphical display
              and tablet input.

   Options for [o] include 0, 1, 2, or 3
   but also terms such as left or ccw.
   For instance:  turntab x right\n"
 }
 }
```

This macro is invoked in definition 11.

I am yet to encounter unambiguous language around the subject of text interfaces for operating systems that holds up after, say, 1960-something. Even “graphical display” is horribly imprecise. I’m sorry for any confusion.

6.2 Cite origins

Cite origins[22] ≡

```
{software () {
   printf '
 This is turntab Version[2].
 <URL for turntab[5]>
 Dates[3]Credits[4]
 Copyright status[6]
 Acknowledgement of Country[1]'
 }
 }
```

This macro is invoked in definition 11.

7 Finish

Finish[23] ≡

```
{# Humphrey, we're leaving!

 exit $?}
```

This macro is invoked in definition 10.

Tada, the end... Actually, there is endless room for improvement. Thanks for reading!

Catch you round like
all the way you've come
after four `turntab increment`s,

Kermie F. Breydon
Boon Wurrung Country, 7 October 2020