Using the Checklist Script

display clist.lua

Carsten Lynker

March 16, 2013

Contents

1 Basic Usage

1.1 Why this script?

There are many files `clist.txt` in public, who are working with the famous plugin Checklister, made for X-Plane 8.

All you have to do is to copy these files into the aircraft directory where the main aircraft file `*.acf` is located. And you will have to install Checklister.

But Checklister won't work in 64-bit. So I decided to write a script displaying the `clist.txt` files.

All you have to do to get checklists in 64-bit is to copy the script `display clist.lua` into the `Scripts` directory of FlyWithLua. After that, just move your mouse pointer to the left side of your screen (or window) to display a menu of all pages inside the `clist.txt` file. While the mouse pointer is on the left side, move up/down to choose the checklist page you want to see and move away from the left side.

When the mouse pointer leaves the left side of the screen and a page is selected, then this page of the checklist will be displayed. To hide the page, just go back to the left side and find an up/down position, where no page is selected. Then move away.

It's as simple as that!

1.2 Don't want it in 32-bit?

You really like the original Checklister and don't want to see the script working in 32-bit? Just rename the script from `display clist.lua` to `display clist.lua64` and FlyWithLua will only execute it in 64-bit.

2 Advanced Usage

2.1 Write easy formatted checklists

If you can't find a `clist.txt` for your plane, you can write a checklist in an easy way. Instead of creating a file `clist.txt` create a file `FWL_checklist.txt` and copy it into the main aircraft directory (just like a `clist.txt` file).

A file `*_checklist.txt` must look like this:

```
Transponder Codes
----------------
VFR in North American airspace      1200
VFR standard squawk code            7000
VFR circuit traffic code in the UK  7010
Radio Failure                       7600
Emergency (ICAO, worldwide)         7700

ITIPAR Initial Call
------------------
Identification                   D-ESAG
Type of Aircraft                 Cessna 172
Intention                        VFR to Hamburg
Position                         position 5 minutes to Sierra 1
Altitude                         altitude 1500 feet
Request                          for landing
```

Use an ASCII text editor and start with the title of the first page.

After the title you can add a line containing at least two - (minus) characters. This is an optional feature, if you leave it away, just continue with the first row of the page.

Then write the rows of the page. If you want to format them, use spaces or tab. A monospace font in your editor can help, but keep in mind to save it as an ASCII text file without any font or style information.

To write down the next page, separate them by an empty line. Make sure that this line is really empty and does not contain any space or tab character.

That's it. A very simple checklist format. This format is good to copy&paste checklists from PDFs.

2.2 The Checklists subfolder

There has to be a subfolder inside `Scripts` folder named `Checklists`. Inside this subfolder you can store as many checklist files as you like. FlyWithLua will load them, if they fit to these names (loading them in these order):

`TAILNUMBER_`*tailnumber*`_clist.txt`
`TAILNUMBER_`*tailnumber*`_checklist.txt`
`ICAO_`*ICAO code*`_clist.txt`
`ICAO_`*ICAO code*`_checklist.txt`
`global_checklist.txt`

If the script finds more then one checklist that matches, it will load all of them and stick them together.

A file matching to the ICAO code will only be loaded if no other file was found before (matching to the tailnumber or be placed inside the aircraft folder).

2.3 Lua code inside a checklist

You can use Lua code inside a checklist. To do this, you will have to write the code surrounded by a letter `$` before and after the code. The code must be finished in the same line.

Here is an example of a checklist page:

```
Speeds
------
Vso            $get("sim/aircraft/view/acf_Vso")$
Vs             $get("sim/aircraft/view/acf_Vs")$
Vfe            $get("sim/aircraft/view/acf_Vfe")$
Vno            $get("sim/aircraft/view/acf_Vno")$
Vne            $get("sim/aircraft/view/acf_Vne")$
```

The code will be replaced by it's returning value (must be a number or a string), while the script is loading the checklist file. If you want it more dynamic, you can use `%%` instead of `$`. Then the code will be evaluated every single frame. Be careful because this can slow down the simulator, if you use slow code.

If you want to execute Lua code, but do not want it to be displayed, then you can start a line with > (greater than). Look at this example:

```
>require("radio")

Plane Data
----------
ICAO code                      "$PLANE_ICAO$"
Tailnumber                     "$PLANE_TAILNUMBER$"
Your transponder is set to     %%SQUAWK%%
Your transponder mode is       %%if (TRANSPONDER_MODE > 1) then return "ACTIVE" else return "STANDBY" end%%
```

In the first line of this example, the module `radio` is loaded. It provides the variables `SQUAWK` and `TRANSPONDER_MODE`.

As the code is masked by `%%`, it will be evaluated every frame. So if you change the transponder settings during flight, the checklist page will always show the right values.

Another nice thing is, that you can use `return` to create a more complex Lua code. In the checklist page you will see the word ACTIVE, if the value of the variable `TRANSPONDER_MODE` is greater than 1, else you will see the word STANDBY.

If you want to be more precise, you can use this code:

```
>require("radio")
>CCTMODE = {"OFF", "STANDBY", "ACTIVE", "TEST"}

Plane Data
----------
ICAO code                      "$PLANE_ICAO$"
Tailnumber                     "$PLANE_TAILNUMBER$"
Your transponder is set to     %%SQUAWK%%
Your transponder mode is       %%CCTMODE[TRANSPONDER_MODE+1]%%
```

3 Change design

You can change the design of the displayed windows in the `display clist.lua` script.

Search for this part of the script:

```
--------8<----------------------------------------------------------------------------------------
-- You can edit these 5 parameters to customize the output of this script.
local rowhight = 21                     -- the line spacing in screen pixel
local framewidth = 5                    -- the space between text and frame in screen pixel
local y_offset = 25                     -- the distance of the window from the upper screen limit
local show_on_right_side = false        -- set this to true to display the pages on the right side
local transparent_percent = 0.65        -- the darkness of the windows background
---------------------------------------------------------------------------->8--------------------
```

The most important value would be the variable `show_on_right_side`. Change it's value to `true` and the script will display the menu and the pages on the right side of the screen.

The value of `transparent_percent` must be between 0.0 and 1.0, where 1.0 will result in a complete black window, and 0.0 will force the script to display only the text without the background.