

Einstieg in die Gerätekonfiguration

Mit Hilfe des Plugins FlyWithLua

Carsten Lynker

15. März 2015





Inhaltsverzeichnis

1	Eine Warnung	4
2	Installation der Plugins	5
2.1	XSquawkBox	5
2.2	DataRefEditor	5
2.3	FlyWithLua	6
2.4	Darstellungsmodelle	6
2.5	FlyWithLua vorbereiten	7
3	Flugzeuge und Szenarien	8
3.1	Flugzeuge	8
3.2	Szenerie	8
3.3	Download Hinweise	9
3.4	Meinungen zum Aufbau der Welt	9
4	Die XSquawkBox	11
4.1	Einrichtung	11
4.2	Tipps zur Nutzung	14
5	Geräte konfigurieren	16
5.1	Achsen	16
5.2	Tasten	17
6	FlyWithLua	20
6.1	Das Script-Konzept	20
6.2	Scripte verwenden	21
6.3	Eine Mehrfachkonfiguration erschaffen	21
6.4	Unser erstes Programm	23
6.5	Variablen	23
6.6	Bedingte Programmierung	25
6.7	Code direkt ausführen	26
6.8	Zusammenfassung	27
7	DataRefs	28
7.1	Variablen im Simulator	28
7.2	Einzelwerte und Arrays	28
7.3	Der DataRefEditor	28
7.4	Startwerte für DataRefs	30



8 Die Beispielscripte	31
8.1 anti rollover brake.lua	31
8.2 automatic set qnh.lua	31
8.3 automatic_cowl_flaps.lua	32
8.4 autopilot infoline.lua	32
8.5 call ATC on airport.lua	32
8.6 display clist.lua	33
8.7 gpu.lua	33
8.8 heading speed altitude instrument.lua	34
8.9 HeliTrim.lua	34
8.10 Instrumententest.lua	34
8.11 Joystick_Sensitivity.lua	34
8.12 keystroke_sniffer.lua	35
8.13 lean_helper.lua	35
8.14 mixture_spline.lua	35
8.15 OpenDoors.lua	36
8.16 program GPX files into FMC.lua	36
8.17 QNH_helper.lua	36
8.18 reload scenery.lua	36
8.19 show last metar and next airport.lua	37
8.20 SimpleTrim.lua	37
8.21 transponder_helper.lua	37
8.22 VFR autopilot helper.lua	37
8.23 VFR weather.lua	38
8.24 XSB_helper.lua	38
9 Hilfe bei Problemen	39

This manual is made to fit the need of VATSIM Germany, giving a brief help to get new pilots connected.

It's under the same licence as FlyWithLua, so if you want it in your language, feel free to translate it. If you have made a translation, please send a copy to me, and I can include it into the official FlyWithLua ZIP file.

If you need the \LaTeX source code of it, please ask and I will send it to you via e-mail.



1 Eine Warnung

Bevor man FlyWithLua verwendet, sollte man sich bewusst sein, dass man FlyWithLua nicht umsonst bekommt! FlyWithLua ist ein quelloffenes und freies Plugin in dem Sinne, dass man es nicht mit Geld kaufen muss (man darf dem Autor aber spenden, wenn man das möchte).

Nein, man »bezahlt« mit einem Teil seiner Lebenszeit, denn FlyWithLua erfordert eine gewisse Einarbeitung, um es sinnvoll einsetzen zu können.

Dieses PDF versucht, diese Einarbeitung für Nicht-Programmierer so einfach und kurz zu gestalten wie möglich. Weil speziell Einsteiger angesprochen werden, kann aber der Einsatz von FlyWithLua auch nicht ohne eine gewisse Detailtiefe vermittelt werden.

Wer dieses PDF sorgsam durcharbeitet, der kann FlyWithLua danach einsetzen und von den Beispielscripten profitieren. Ein Programmierkurs ist dieses PDF nicht. Soll näher in die Programmierung mit Lua eingestiegen werden, so führt kein Weg am original Handbuch vorbei. Wie in der Luftfahrt üblich, liegt es in englischer Sprache vor.



2 Installation der Plugins

2.1 XSquawkBox

Zunächst muss die XSquawkBox installiert werden, damit wir uns mit dem VATSIM Netz verbinden können und um eine direkte Eingabezeile für FlyWithLua zu haben.

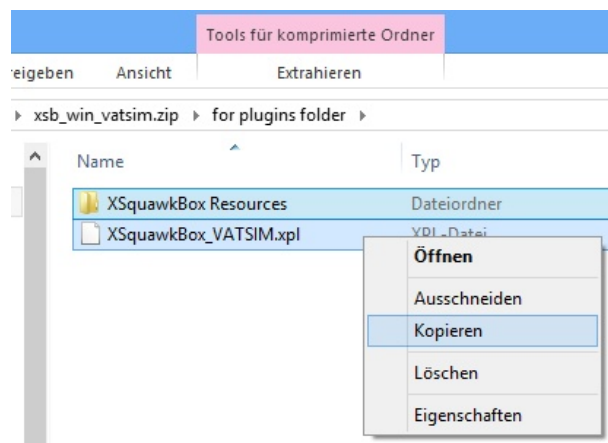
Wir laden die neueste Version von dieser Seite herunter:

<http://www.xsquawkbox.net/xsb/download/>

Das Paket liegt in Form eines ZIP Archivs vor. Wir müssen nun, um das Paket zu installieren, es in das Unterverzeichnis für die Plugins entpacken:

Ort wo X-Plane liegt/X-Plane 10/Resources/plugins/

Wir öffnen einfach das ZIP Archiv durch einen Doppelklick, wechseln in den Unterordner und drücken Strg-A zum Auswählen beider Elemente und dann Strg-C um sie zu kopieren. Alternativ kann es mit der Maus geschehen, wie hier unter Windows 8 zu sehen:



Dann wechseln wir in das Unterverzeichnis für die Plugins und drücken Strg-V, um beide Elemente dort zu platzieren. Auf anderen Betriebssystemen muss man entsprechend vorgehen.

2.2 DataRefEditor

Ein weiteres Plugin, dass wir für die Einrichtung von X-Plane nutzen werden, ist der DataRef-Editor. Dieses Plugin kann man hier laden:

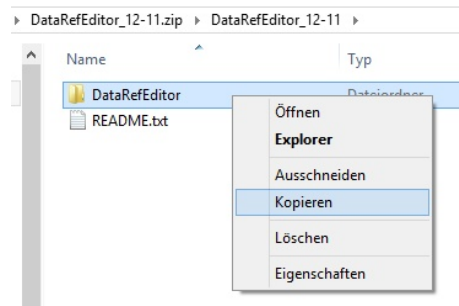
<http://www.xsquawkbox.net/xpsdk/mediawiki/DataRefEditor>

Wie zuvor bei der XSquawkBox kopieren wir den Unterordner DataRefEditor in das Verzeichnis für Plugins.



2.3 FlyWithLua

2 INSTALLATION DER PLUGINS



2.3 FlyWithLua

Zum Schluss fehlt uns noch das Plugin FlyWithLua. Wir laden es hier:

<http://forums.x-plane.org/index.php?app=downloads&showfile=17468>

Das ZIP enthält einen Unterordner FlyWithLua, den wir in das Pluginverzeichnis entpacken. Jetzt sollte unser Pluginverzeichnis so aussehen:

Name	Änderungsdatum	Typ	Größe
DataRefEditor	04.02.2013 12:12	Dateiordner	
FlyWithLua	27.11.2012 19:46	Dateiordner	
PluginAdmin	23.01.2013 13:31	Dateiordner	
XPLM.framework	23.01.2013 13:31	Dateiordner	
XPWidgets.framework	23.01.2013 13:31	Dateiordner	
XSquawkBox Resources	19.02.2012 18:05	Dateiordner	
Commands.txt	19.12.2011 17:14	TXT-Datei	66 KB
DataRefs.txt	10.12.2012 12:14	TXT-Datei	359 KB
XPLM.dll	31.12.2012 09:24	Anwendungserwe...	173 KB
XPLM.shlb	17.11.2012 14:14	SHLB-Datei	55 KB
XPLM.so	23.01.2013 13:31	SO-Datei	532 KB
XPLM_64.dll	31.12.2012 09:24	Anwendungserwe...	218 KB
XPLM_64.so	23.01.2013 13:31	SO-Datei	596 KB
XPWidgets.dll	26.11.2012 22:35	Anwendungserwe...	95 KB
XPWidgets.shlb	17.11.2012 14:14	SHLB-Datei	87 KB
XPWidgets.so	17.11.2012 14:14	SO-Datei	160 KB
XPWidgets_64.dll	17.11.2012 14:14	Anwendungserwe...	114 KB
XPWidgets_64.so	17.11.2012 14:14	SO-Datei	179 KB
XSquawkBox_VATSIM.xpl	12.10.2011 15:39	XPL-Datei	940 KB

2.4 Darstellungsmodelle

Abschließend sind wir so wagemutig und starten X-Plane. Der Start dauert nun etwas länger, weil das Plugin XSquawkBox während des Starts Informationen zur Darstellung anderer Flieger sammelt. Diese Informationen liegen in dem Unterverzeichnis

Ort wo X-Plane liegt/X-Plane 10/Resources/plugins/XSquawkBox Resources/CSL/



Dort hinein kopieren wir nach eigenem Bedürfnis weitere CSL Pakete, die man hier laden kann:

<http://forums.x-plane.org/index.php?app=downloads&showcat=12>

Je mehr CSL Pakete wir installieren, desto eher hat die XSquawkBox die Chance andere Flieger mit einem korrekten Modell darzustellen, desto länger dauert aber auch der Start von X-Plane. Man sollte also nicht alles installieren was man im Netz findet, sondern nur die Modelle, denen man auch in VATSIM begegnet.

Haben wir neue CSL Pakete installiert, so müssen wir X-Plane neu starten. Die XSquawkBox überwacht den CSL Unterordner nicht, sondern wertet ihn nur beim Start einmalig aus.

2.5 FlyWithLua vorbereiten

Nach dem ersten Start mit unseren drei neuen Plugins begrüßt uns FlyWithLua mit einem relativ nervigen Text, wir sollen gefälligst das Handbuch lesen. Wird dieser Text angezeigt, dann ist FlyWithLua korrekt gestartet und wir haben die Installation beinahe geschafft. Wir beenden X-Plane einfach, installieren je nach Bedarf ein paar CSL Pakete für die XSquawkBox und löschen dann folgende Dateien:

```
/X-Plane 10/Resources/plugins/FlyWithLua/Scripts/please read the manual.lua  
/X-Plane 10/Resources/plugins/FlyWithLua/Scripts/stupid bubble test.lua
```

FlyWithLua wird alle Dateien ausführen, die sich im Scripts Ordner befinden und die auf .lua oder .fwl enden. Dateien mit anderen Endungen wie .txt werden nicht automatisch gestartet, auch wenn sich darin ausführbarer Lua Code befindet. Durch das Löschen der beiden Dateien haben wir den nervigen Text und die störenden Sprechblasen entfernt, sie wurden durch Lua Code in den beiden Dateien erzeugt.



3 Flugzeuge und Szenerien

Auch für Flugzeuge und Szenerie gilt zur Installation das gleiche Prinzip. Man muss die Pakete, die man sich im Internet heruntergeladen hat, in das dafür vorgesehene Verzeichnis entpacken.

3.1 Flugzeuge

Die Flugzeuge entpackt man in den Ordner

Ort wo X-Plane liegt/X-Plane 10/Aircraft/

oder einen beliebigen Unterordner davon. Es empfiehlt sich, einen Unterordner `Downloads` anzulegen, und die Flieger, die man aus dem Netz geladen hat, dort hinein zu »installieren«.

Jedes installierte Flugzeug besteht aus einem eigenen Ordner, indem sich mindestens eine Datei mit der Endung `.acf` befindet.

Man kann beliebig viele Flugzeuge installieren, ohne die Performance von X-Plane zu beeinträchtigen. Denn X-Plane lädt immer nur ein bestimmtes Flugzeug, erstellt aber keine Datenbank aller Flugzeuge. Es ist daher auch möglich, ein neues Flugzeug zu installieren (in einen passenden Unterordner zu kopieren), und direkt zu verwenden, ohne X-Plane neu starten zu müssen.

3.2 Szenerie

Mit der Szenerie ist es fast wie mit den Flugzeugen, sie sind Ordner, die an den dafür vorgesehenen Platz kopiert werden:

Ort wo X-Plane liegt/X-Plane 10/Custom Scenery/

Allerdings lassen sie sich nicht so einfach im laufenden Betrieb installieren. Hat man eine neue Szenerie installiert, muss man X-Plane einmal neu starten, und dann wieder beenden. Anschließend editiert man folgende Datei, die X-Plane nach jeder neu hinzugefügten Szenerie neu anlegt:

Ort wo X-Plane liegt/X-Plane 10/Custom Scenery/scenery_packs.ini

Innerhalb dieser Datei wird die Hierarchie der Szenerien festgelegt. Man kann sich die Hierarchie vorstellen wie das Übereinanderlegen von Folien. An den Stellen, wo eine oben liegende Folie transparent ist, kann man die darunter liegenden Folien sehen. Das entspricht der Position der zugehörigen Zeile in der Datei. Szenerie in einer früheren, quasi oberen, Zeile kann die später folgenden Zeilen überdecken. An Stellen wo diese Szenerie es erlaubt werden darunter liegende Szenerien sichtbar.



Deshalb bringen wir 3D Objekte nach oben und Bodentextur nach unten. Man kann die Szenereihefolge so lange anpassen, bis alles passt. Wer X-Plane dabei nicht ständig neu starten will, der kann das Lua Script `reload scenery.lua` verwenden.

3.3 Download Hinweise

Szenerie findet man im Internet unter anderem hier:

<http://zonephoto.x-plane.fr/>

<http://simheaven.de/>

Manche Szenerie erfordert die Installation der Bibliothek OpenSceneryX, die man hier laden kann:

<http://www.opensceneryx.com/>

Wer sich eine aktuelle 3D Szenerie selbst aus Daten von OpenStreetMap erzeugen möchte, der findet hier ein hilfreiches Programm:

<http://osm2xp.com/>

3.4 Meinungen zum Aufbau der Welt

Austin Meyer, der Erfinder und Kopf hinter X-Plane, setzt mit X-Plane 10 voll auf die Erzeugung einer 3D Welt, wie sie sein könnte, anhand von Daten aus dem OpenStreetMap Projekt.

Fototapeten auf den Boden zu zeichnen hält Austin Meyer dabei für den falschen Weg. Daher bleibt es spannend zu beobachten, wie X-Plane künftig die Welt darstellen wird.

Meine persönliche Meinung ist, dass Fototapeten ein falscher Weg sind zur Darstellung in X-Plane 10. Denn die 3D Leistung von X-Plane 10 ist enorm. X-Plane 10 baut die Welt aus neutralen Bodentexturen auf und setzt darauf ein Straßennetz, dessen Daten aus dem OpenStreetMap Projekt stammen. Um diese Straßen herum werden Gebäude platziert, wie sie in der Realität stehen *könnten*. Denn nicht überall sind die Gebäude ausreichend in OSM als Datensatz zu finden. Dies stellt einen relativ guten Kompromiss zwischen Exaktheit und löchrigen Daten dar.

In einigen Gebieten liegen die Daten jedoch in hoher Dichte vor, so dass man aus den Daten von OSM ein Gebiet vernünftig darstellen kann, ohne zusätzlich Gebäude zu erfinden. Allerdings sind dazwischen auch oft Gebiete ohne Daten, wie das folgende Beispiel der Stadt Paderborn zeigt.



Um diese Stellen mit Objekten zu befüllen sollte man das »Autogen« von X-Plane nutzen. Hässlich wird es aber, wenn man 3D Darstellung mit einer Fototapete mischt. Denn dann wachsen Gebäude aus auf den Boden gewalzten Gebäuden heraus.

4 Die XSquawkBox

4.1 Einrichtung

Die Einrichtung der XSquawkBox vollzieht sich in mehreren Schritten. Zunächst klicken wir auf das Menü Plugins, dann auf XSquawkBox und schließlich auf Setup Audio... und erhalten einen Dialog zur Einrichtung der Hardware für die Kommunikation mit VATSIM Lotsen und anderen Fliegern.



Wir müssen hier eine Hardware wählen, damit die XSquawkBox weiß womit sie arbeiten soll. Die Einstellungen sind schnell erledigt, auf den Button Setup Mic klicken wir nicht.

Hinter dem Button Setup Mic verbirgt sich ein Einstelldialog, der versucht automatisch den Pegel des Mikrofons zu ermitteln. Die Erfahrung zeigt, dass dies oft daneben geht und dann nichts mehr übertragen wird.

Konnten wir dennoch die Finger nicht davon lassen und haben uns die Audioquelle in der Eingangspegelschwelle verstellt, dann können wir X-Plane beenden und folgende Datei löschen, wodurch XSquawkBox zurück gesetzt wird:

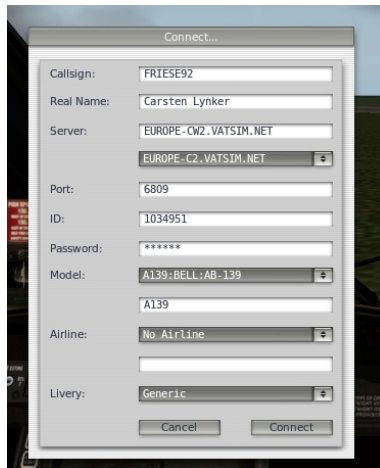
```
.../X-Plane 10/Resources/plugins/XSquawkBox Resources/XSquawkBox Prefs
```



4.1 Einrichtung

4 DIE XSQUAWKBOX

Jetzt verbinden wir uns mit dem VATSIM Netz, indem wir Plugins, dann XSquawkBox und Connect... anklicken. Wir erhalten diesen Dialog:



Unter Callsign geben wir hier unser Flugzeugkennzeichen ein, also z. B. DHXYZ für einen Helikopter oder DEABC für eine Cessna 172SP. Das Callsign kann während einer bestehenden Verbindung nicht verändert werden, auch nicht das Muster, dass wir fliegen. Haben wir uns also als DHHHH mit einer Robin 22 Kaffeemühle eingeloggt, wechseln dann aber mit dem entsprechenden Menü von X-Plane zu einer Boeing 747, so sehen uns alle anderen Teilnehmer im VATSIM Netz weiterhin als Helikopter. Mit hoher Geschwindigkeit in einigen Kilometern Höhe wird das dann schnell unglaublich.

Ändern wir also unser Flugmuster, so müssen wir uns abmelden und dann neu wieder anmelden.

Die Eintragung unter Real Name sollte klar sein und der eigene Vor- und Zuname ist Pflicht¹. Bei Server tragen wir nichts ein, sondern wählen den Server über den Button darunter aus.

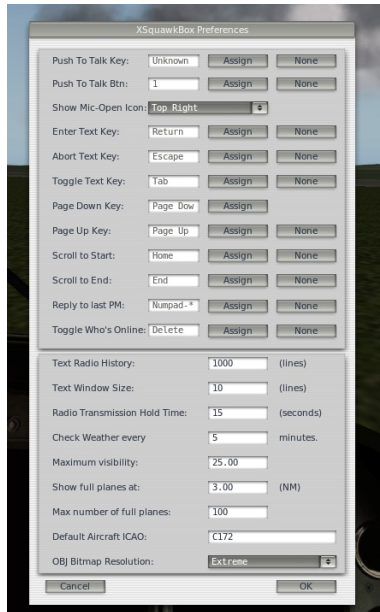
Der Port sollte 6809 lauten, ID und Passwort entsprechen denen, die wir von VATSIM bei der Anmeldung genannt bekommen haben. Hier muss die VATSIM-ID eingetragen werden und auch das VATSIM-Kennwort, nicht die Anmeldedaten für die VATSIM-Germany.

Jetzt löschen wir alles aus den Zeilen darunter (Model, Airline und Livery). Unter Model geben wir nun die ICAO Kennung unseres Flugmusters ein, z. B. C172 für die kleine Cessna. Wer sich nicht sicher ist, welche Kennung sein Muster hat, findet im Netz viele Quellen dazu. Als Beispiel:

http://www.flugzeuginfo.net/table_accodes_dt.php

Als letztes schauen wir uns noch das Untermenü Preferences... an.

¹ Artikel A4 der [Code of Conduct](#) schreibt echte Namen vor.



Zum Sprechen am VATSIM-Funk benötigen wir mindestens die Zuweisung einer Sprech taste oder eines Sprechknopfes (am Joystick). Wir klicken dazu auf **Assign** und drücken anschließend die gewünschte Taste oder den gewünschten Knopf.

Weiter wichtig sind **Enter Text Key** und **Toggle Text Key**. Hier sind **Enter** und **Tab** voreingestellt, was wir so belassen können.

Auch die weiteren Einstellungen können bei der Voreinstellung bleiben. Lediglich die beiden letzten Punkte sollten wir unserem Wunsch nach anpassen.

Der Punkt **Default Aircraft ICAO** wird immer dann wichtig, wenn die XSquawkBox einen anderen Flieger nicht in ihrer CSL Paketdatenbank findet. Dann wird statt nach der eigentlichen ICAO nach dem hier eingetragenen Code gesucht und das Flugzeug entsprechend dargestellt.

Stehen wir also an einem Flughafen und neben uns parkt ein anderer VATSIM Teilnehmer mit einem Boeing Dreamliner, dessen ICAO Code **B788** nicht in der CSL Datenbank gefunden werden kann, dann wird dieser Flieger durch eine Cessna 172 angezeigt, wenn man wie hier gezeigt einen Eintrag **C172** gemacht hat. Wer also lieber »dicke Pötte« fliegt, der wird hier vermutlich lieber **A320** eintragen.

Die Auswahl **OBJ Bitmap Resolution** gibt an, wie groß eine Bitmap sein darf, mit der ein Flieger dargestellt wird. Höhere Werte sind schöner anzusehen, schlucken aber mehr »Frames«. Einen geeigneten Wert kann man nur durch Ausprobieren finden.

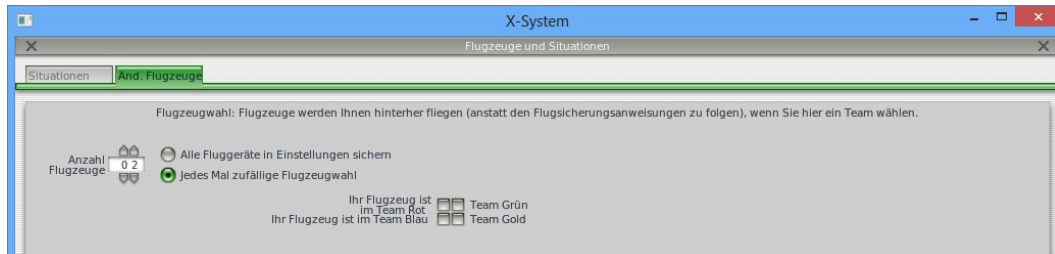
Zum Abschluss der Konfiguration der XSquawkBox klicken wir noch in das **X-Plane Menü** Flugzeuge, dann auf **Flugzeuge** und **Situationen** und darin auf den Reiter **And. Flugzeuge**.



4.2 Tipps zur Nutzung

4 DIE XSQUAWKBOX

Unter Anzahl der Flugzeuge stellen wir einen Wert von mindestens 2 ein (oder mehr, je nach Leistung des PC)².



XSquawkBox benötigt diesen Wert von 2 oder höher, um andere Flieger darstellen zu können. Allerdings bezieht sich der Wert nur auf Flugzeuge, die über X-Plane als komplettes Flugmuster geladen werden (so wie X-Plane selbst auch andere Flieger simuliert). Der Normalfall ist aber, dass die anderen Teilnehmer aus VATSIM über Objekte dargestellt werden, die XSquawkBox aus den CSL Paketen generiert.

Die Anzahl der durch CSL generierten Objekte ist nicht durch diesen Eintrag begrenzt, sondern durch den Eintrag *Max number of full planes* in den Einstellungen der XSquawkBox. Man braucht also keine Angst haben, beim Eintrag von 2 später nur im Duett durch VATSIM fliegen zu können.

Sind alle diese Einstellungen vorgenommen, dann kann X-Plane jetzt ohne Fehlermeldungen gestartet werden und dem ersten Flug in VATSIM steht nichts mehr im Weg.

4.2 Tipps zur Nutzung

Stellen wir uns auf den Regionalflughafen Paderborn/Lippstadt EDLP und loggen wir uns über XSquawkBox in das VATSIM Netz ein. Jetzt drücken wir die Taste *Entf*, oder was auch immer wir in den Einstellungen von XSquawkBox unter *Toggle Who's Online* eingestellt haben. Durch erneutes Drücken verschwindet die Liste wieder. Ist man gerade erst frisch online, dann sollte man etwas warten, bis die Liste sich gefüllt hat.

Wir erhalten eine Übersicht aller Radiostationen, die wir in unserer Funkreichweite haben. Dort ist zufällig der Ground Controller zu sehen. Wir möchten nun mit diesem Controller Kontakt aufnehmen. Dazu tippen wir folgendes auf dem Ziffernblock:

Enter //121,92 Enter

XSquawkBox öffnet bei ersten Drücken der Entertaste eine Eingabezeile und macht aus dem Komma einen Punkt. Mit erneutem Druck der Entertaste wird die Zeile wieder geschlossen und ausgewertet.

²Bei X-Plane 10 scheint der Wert entgegen der Anleitung der XSquawkBox nicht mehr relevant zu sein. Man kann bei schwacher PC-Leistung auch einen Wert von 1 eintragen. Sieht man dann andere Flieger, wenn man online ist, ist alles in Ordnung. Eine Fehlermeldung der XSquawkBox muss dann ignoriert werden.



Wir haben mit den beiden Slash einen Kurzbefehl begonnen um die Frequenz für COM1 festzulegen. Wenn wir auf unser Radio schauen, werden wir es sehen. Nun wollen wir die ATIS abhören und tippen:

Enter `///125,72` *Enter*

Mit drei statt zwei Slash stellen wir COM2 ein. Auf COM2 können wir hören, aber nicht sprechen. Haben wir uns an der ATIS sattgehört, wechseln wir COM2 auf UNICOM:

Enter `///122,80` *Enter*

Beginnen wir die Eingabe nicht mit Zeichen, die für einen Kurzbefehl stehen, so wird die Eingabe als Textmitteilung auf der COM1 Frequenz gesendet. Geben wir also ein:

Enter `Hallo Leute, ich fliege gleich nach Dortmund!` *Enter*

Mit dem Abschicken hören wir ein »Klack« Geräusch und bekommen den Text für ein paar Sekunden von der XSquawkBox angezeigt. Der Controller und alle anderen Teilnehmer in Funkreichweite wissen nun, dass wir blutige Anfänger sind und geben entsprechend Acht.

Neben uns parkt unser Mentor. Wollen wir ihm eine private Textnachricht senden, die die anderen Teilnehmer auf der Frequenz nicht sehen sollen, dann beginnen wir die Zeile mit `.msg` gefolgt von einem Leerzeichen, dem Callsign des Teilnehmers, den wir erreichen wollen, und dem eigentlichen Text, getrennt durch ein weiteres Leerzeichen. Also etwa so:

Enter `.msg FRIESE201 War das gerade falsch?` *Enter*

Auf dem Weg nach Dortmund stellen wir später fest, dass der Platz gar nicht mit einem Lotsen besetzt ist. Dennoch möchten wir QNH und aktive Piste wissen. Wir holen uns dazu das Metar:

Enter `.metar EDLW` *Enter*

Aus dem Metar können wir das QNH direkt ablesen, die aktive Piste wählen wir uns so, dass wir mit Gegen- statt Rückenwind landen.

Wäre ein Lotse online, so könnten wir uns das ATIS auch als Text anzeigen lassen:

Enter `.atis EDLW_TWR` *Enter*

Hierbei muss immer das vollständige »Callsign« des Lotsen angegeben werden. Wird die entsprechende Frequenz eingedreht, also der Kontakt zu einem Lotsen erstmalig aufgebaut, so zeigt XSquawkBox diese ATIS automatisch an.



Seite 16 von 39



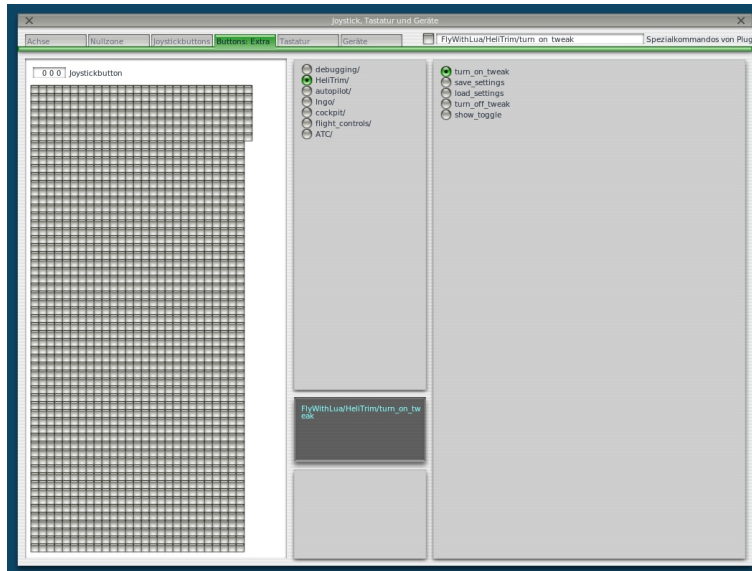
Was uns direkt zum zweiten Reiter führt. Denn nun zentrieren wir alle Achsen und klicken auf den großen Button. Jetzt sind die Achsen fertig eingestellt. In diesem Beispiel sind die Achsen so gewählt, dass man einen Helikopter steuern kann.

Oben auf diesem zweiten Reiter finden wir Einstellungen zur künstlichen Stabilität und zur Linearität der Achsen. Für die Steuerung eines Helikopters wird eigentlich eine vollständig lineare Steuerung empfohlen, die Stabilitätsgrade sind Geschmackssache. Oft empfehlen die Hersteller von AddOn Fliegern in ihrem Handbuch bestimmte Einstellungen für Stabilität und Linearität. In einem solchen Fall sollte man mit diesen Empfehlungen beginnen und die Werte dann nach ein paar Flügen wenn nötig anpassen.

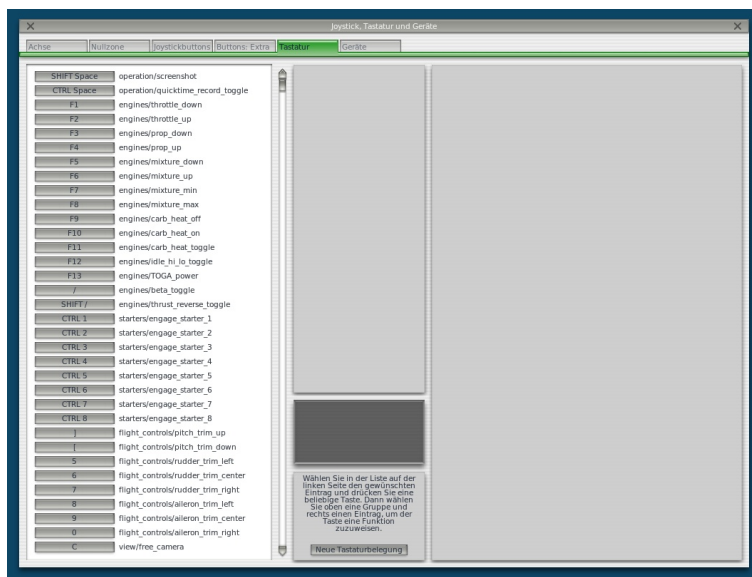
5.2 Tasten

Tasten auf der Tastatur oder auf einem Joystick lassen sich jeweils genau einem Kommando zuweisen. Ein Kommando besteht aus drei Teilen, in denen jeweils bestimmte Handlungen definiert sind. Ein Teil beschreibt was passieren soll, wenn die Taste gedrückt wird. Dieser Teil wird genau einmal ausgeführt. Ein weiterer Teil beschreibt, was passieren soll, wenn die Taste gehalten wird und schließlich beschreibt der dritte Teil, was beim Loslassen der Taste geschehen soll.

Zum Zuweisen der Kommandos zu den einzelnen Tasten auf Joysticks verwenden wir am Besten den Reiter `Buttons Extra`.



Für die Tastatur nutzen wir den Reiter Tastatur.



Die Kommandos haben alle einen eindeutigen Namen, der von der Form an einen Dateinamen mit Pfad erinnert. So bewirkt das Kommando `sim/view/chase` beim Drücken einen Wechsel in die Außenansicht. Es handelt sich um ein von X-Plane selbst bereitgestelltes Kommando, weil sein Name mit `sim/` beginnt. Beim Halten und Loslassen verursacht dieses Kommando keine Aktionen.

Um nun ein Kommando zuzuordnen drücken wir den entsprechenden Knopf und die Nummer des Knopfes erscheint oben links im Fenster. Jetzt wählen wir das Kommando mit der Maus aus, oder wir klicken oben rechts in das Eingabefeld, in dem der Name des aktuell zugewiesen Kommandos zu lesen ist, bzw. auf den Knubbel davor. Dort können wir dann, wie aus einem



Dateimenü, ein Kommando auswählen, auch wenn es mit der Maus gerade nicht zugänglich ist.

Das hier im Bild gezeigte Kommando `FlyWithLua/HeliTrim/turn_on_tweak` kann in X-Plane noch nicht gewählt werden, wenn wir bis hierher strikt dieser Anleitung gefolgt sind. Es handelt sich um ein vom Benutzer geschaffenes Kommando, zu dem wir später kommen werden.

Wir schließen X-plane an dieser Stelle.



6 FlyWithLua

Wird X-Plane gestartet, so werden die zuletzt zugeordneten Achsen und Tasten wiederhergestellt. Das ist nur dann praktisch, solange wir Flugzeuge gleichen Typs verwenden. Soll nun aber, nachdem wir beim letzten Mal einen Helikopter geflogen sind, nun eine Cessna 172 geflogen werden, so müssen wir nach dem Laden des Flugzeugmodells erneut in das Menü Joystick, Tastatur und Geräte hinein und entsprechende Anpassungen vornehmen.

Hier setzt FlyWithLua an und vereinfacht uns die Arbeit. Doch schauen wir uns zunächst an, wie FlyWithLua überhaupt arbeitet.

6.1 Das Script-Konzept

Kern von FlyWithLua sind die Scripte. Dies sind in **Lua** (einer Programmiersprache) geschriebene kleine Programme. Das Plugin FlyWithLua durchsucht nun bei jedem Start, dem Laden eines neuen Fliegers oder dem Wechsel des Ortes den Ordner

Ort wo X-Plane liegt/X-Plane 10/Resources/plugins/FlyWithLua/Scripts/

nach Dateien, die auf `.lua` oder `.fwl` enden. Alle dabei gefundenen Scripte werden an den Lua Compiler LuaJIT 2.0 übergeben, von diesem in Maschinencode übersetzt und ausgeführt. Weil die Übersetzung in Maschinencode erst beim Laden erfolgt, lassen sich die Dateien mit einem gewöhnlichen Texteditor lesen und bearbeiten. Um dies zu überprüfen, öffnen wir die Datei `QNH_helper.lua` und erhalten lesbaren Code (auch wenn wir ihn vielleicht noch nicht verstehen). Wir finden diese Datei im Unterordner `Scripts (disabled)`.

```
1 DataRef( "QNH_Pilot", "sim/cockpit2/gauges/actuators/barometer_setting_in_hg_pilot", "
   writable" )
2 last_QNH_Pilot = QNH_Pilot
3 QNH_display_time = os.clock()
4
5 function QNH_helper()
6     -- if the QNH setting has changed, we want to display the info for 3 sec
7     if QNH_Pilot ~= last_QNH_Pilot then
8         QNH_display_time = os.clock() + 3.0
9     end
10    -- we place the info above the mouse pointer
11    if os.clock() < QNH_display_time then
12        draw_string(MOUSE_X - 110, MOUSE_Y + 20, "a QNH of " .. math.floor( (
            QNH_Pilot * 100) + 0.5) / 100 .. "in/hg is " .. math.floor(
            QNH_Pilot * 33.8637526 + 0.5) .. "hPa in Europe")
13        last_QNH_Pilot = QNH_Pilot
14    end
15 end
16
17 do_every_draw( "QNH_helper()" )
```

Sollten sich jetzt die großen Fragezeichen auf der Stirn einfunden, so ist das nicht weiter schlimm. Wir müssen kein Programmierer sein, um FlyWithLua einsetzen zu können.



6.2 Skripte verwenden

Wir können Skripte einfach verwenden, indem wir sie in das Skriptverzeichnis hinein kopieren. Dies sollten wir nun mit den folgenden Dateien machen, die sich alle im Unterverzeichnis `Scripts (disabled)` befinden. Es handelt sich dabei um Skripte, die Bestandteil des FlyWithLua Plugins sind.

```
automatic set qnh.lua  
call ATC on airport.lua32  
gpu.lua  
QNH_helper.lua  
VFR weather.lua
```

Nachdem wir die Skriptdateien in das Verzeichnis `Scripts` kopiert haben, klicken wir auf den Menüpunkt `Plugins, FlyWithLua und Reload all Lua script files`. Jetzt werden die neu hinzugefügten Skripte geladen, übersetzt und ausgeführt. **Ein Neustart von X-Plane ist nicht erforderlich!**

Alle fünf Skripte haben nun ihre Arbeit begonnen und wir sehen die Auswirkungen zum Teil in neuen Menüs unterhalb von `Plugins` und dann `FlyWithLua Macros` oder `FlyWithLua ATC`. So existiert etwa der Menüpunkt `Ground Power Unit` unterhalb von `FlyWithLua Macros`. Mit diesem Menüpunkt können wir eine GPU anklemmen, auch wenn das Flugmuster dies gar nicht vorsieht. Eine blinkende Anzeige informiert uns über eine angekoppelte GPU. Bevor wir losrollen, sollten wir diese wieder abklemmen.

Man kann sich FlyWithLua also als eine Art von Plugin Sammlung vorstellen, die ähnlich funktioniert wie die in C geschriebenen »großen« Plugins. Nur dass man die Skripte lesen und editieren kann, ganz im Gegensatz zu den `IchBinEinPlugin.xpl` Dateien, die den direkten Maschinencode für ein Plugin enthalten.

6.3 Eine Mehrfachkonfiguration erschaffen

Kommen wir zurück auf unser Problem mit dem ständigen Umkonfigurieren zwischen Flugzeugen und Helikoptern. Wir starten nun X-Plane (soweit noch nicht geschehen) und richten uns eine Cessna C172 ein. Danach beenden wir X-Plane und starten den Simulator erneut.

Nachdem X-Plane neu gestartet wurde, sehen wir uns das Verzeichnis

Ort wo X-Plane liegt/X-Plane 10/Resources/plugins/FlyWithLua/

näher an. Darin befindet sich eine Datei `initial_assignments.txt`, die wir mit einem Texteditor³ öffnen.

³Ein guter Editor unter Windows ist [Notepad++](#).



6.3 Eine Mehrfachkonfiguration erschaffen

6 FLYWITHLUA

Sie sieht in etwa so aus:

```
1  -----
2  ---      FlyWithLua: The initial assignments are stored in this file.      ---
3  ---      8< -----
4
5  clear_all_axis_assignments()
6  set_axis_assignment( 0, "roll", "normal" )
7  set_axis_assignment( 1, "pitch", "normal" )
8  set_axis_assignment( 3, "yaw", "normal" )
9  set_axis_assignment( 10, "mixture", "reverse" )
10 set_axis_assignment( 11, "prop", "reverse" )
11 set_axis_assignment( 12, "throttle", "normal" )
12
13 clear_all_button_assignments()
14 set_button_assignment( (0*40) + 2, "sim/view/chase" )
15 set_button_assignment( (0*40) + 3, "sim/view/chase" )
16 set_button_assignment( (0*40) + 4, "sim/general/backward" )
17 set_button_assignment( (0*40) + 5, "sim/general/forward" )
18 set_button_assignment( (0*40) + 6, "FlyWithLua/autopilot/activate_autopilot" )
19 set_button_assignment( (0*40) + 7, "FlyWithLua/autopilot/set_autopilot_off" )
20 set_button_assignment( (0*40) + 8, "sim/view/flashlight_red" )
21 set_button_assignment( (0*40) + 9, "sim/view/flashlight_wht" )
22 set_button_assignment( (0*40) + 10, "FlyWithLua/debugging/reload_scripts" )
23 set_button_assignment( (0*40) + 13, "FlyWithLua/Ingo/SetQNH" )
24 set_button_assignment( (0*40) + 16, "sim/general/up" )
25 set_button_assignment( (0*40) + 18, "sim/view/night_vision" )
26 set_button_assignment( (0*40) + 20, "sim/general/down" )
27 set_button_assignment( (0*40) + 22, "FlyWithLua/cockpit/toggle_HASI" )
28 set_button_assignment( (4*40) + 0, "sim/flight_controls/flaps_up" )
29 set_button_assignment( (4*40) + 1, "sim/flight_controls/flaps_down" )
30 set_button_assignment( (4*40) + 2, "sim/flight_controls/speed_brakes_up_all" )
31 set_button_assignment( (4*40) + 3, "sim/flight_controls/speed_brakes_down_all" )
32 set_button_assignment( (8*40) + 0, "sim/lights/nav_lights_toggle" )
33 set_button_assignment( (8*40) + 1, "sim/lights/beacon_lights_toggle" )
34 set_button_assignment( (8*40) + 2, "sim/lights/landing_lights_toggle" )
35 set_button_assignment( (8*40) + 3, "sim/lights/strobe_lights_toggle" )
36 set_button_assignment( (8*40) + 6, "sim/operation/dev_console" )
37 set_button_assignment( (8*40) + 7, "sim/operation/toggle_full_screen" )
38
39 --- setting nullzone, sensitivity and augment
40 set( "sim/joystick/joystick_pitch_nullzone", 0.000 )
41 set( "sim/joystick/joystick_roll_nullzone", 0.000 )
42 set( "sim/joystick/joystick_heading_nullzone", 0.000 )
43 set( "sim/joystick/joystick_pitch_sensitivity", 0.500 )
44 set( "sim/joystick/joystick_roll_sensitivity", 0.500 )
45 set( "sim/joystick/joystick_heading_sensitivity", 0.500 )
46 set( "sim/joystick/joystick_pitch_augment", 0.300 )
47 set( "sim/joystick/joystick_roll_augment", 0.300 )
48 set( "sim/joystick/joystick_heading_augment", 0.300 )
49
50 ----- >8 -----
```

Im Texteditor sagen wir nun speichern unter... und legen diese Datei unter dem Namen zzz_Meine_Konfiguration.lua im Verzeichnis Scripts ab.



6.4 Unser erstes Programm

Laden wir jetzt alle Scripte neu, so wird dieses gerade von uns erschaffene Script vom Lua-JIT Compiler übersetzt. Der Compiler erwartet in jeder Zeile entweder nichts, oder einen Lua Befehl.

Wir nennen uns also ab jetzt stolz Programmierer, denn FlyWithLua verarbeitet ein von uns generiertes Script. Na gut, allzu viel haben wir ja noch nicht daran selbst gemacht. Sehen wir uns das Script mal aus Sicht des Compilers an.

Als erstes stößt er auf eine Zeile, die mit zwei aufeinander folgenden Minus beginnt. Findet der Compiler zwei Minus hintereinander, so hört er in dieser Zeile auf und widmet sich der nächsten Zeile. Durch Einsatz der »Doppelminus« Zeichen können wir also Text einfügen, den der Compiler ignoriert. Programmierer sprechen dabei von Kommentaren, die man ausgiebig in Scripten verwenden sollte, denn sie machen das Script selbst nicht langsamer (Kommentare gelangen nicht in den Maschinencode), aber deutlich verständlicher.

Der erste echte Befehl steht in Zeile 5 und lautet `clear_all_axis_assignments()`. Alle Befehle bestehen aus einem Namen, der nicht mit einer Ziffer beginnen und keine Leerzeichen enthalten darf. Direkt nach dem Namen kommen runde Klammern. An diesen Klammern erkennt LuaJIT, dass es sich um einen Befehl handelt.

Innerhalb der Klammern können sich ein oder mehrere Argumente befinden, die durch Komma getrennt werden. In Zeile 6 hat der Befehl `set_axis_assignment()` drei Argumente 0, "roll" und "normal".

6.5 Variablen

Neben Befehlen kennt die Sprache Lua aber auch noch weitere Elemente, nämlich Variablen. Diese Elemente sind wie kleine Schachteln, in die wir etwas hineinlegen können, dass dann später wieder hervor geholt werden kann. Das »Hineinlegen« geschieht durch eine Zuweisung, die wir Lua durch den Namen der Variable, gefolgt von einem Gleichheitszeichen und dem Inhalt, den wir hineinlegen wollen, mitteilen.

Wir fügen vor der Zeile 13 folgendes ein (drei neue Zeilen):

```
Thrustmaster = 0
Quadrant = 160
XBox = 320
```

Dann verwenden wir die Suchen&Ersetzen Funktion des Editors und ersetzen `(0*40)` durch `Thrustmaster`, `(4*40)` durch `Quadrant` und `(8*40)` durch `XBox`.

Natürlich passt das jetzt nicht auf das eigene Script, aber das Verfahren sollte klar sein und kann nun auf die eigenen Geräte angepasst angewendet werden. Das Script kann noch von unnötigen Kommentaren befreit werden und sieht nun ungefähr so aus:



```
1  — define the joystick offsets (for button assignments)
2  Thrustmaster = 0
3  Quadrant = 160
4  XBox = 320
5
6  — do the initial things
7  clear_all_axis_assignments()
8  set_axis_assignment( 0, "roll",      "normal" )
9  set_axis_assignment( 1, "pitch",     "normal" )
10 set_axis_assignment( 3, "yaw",       "normal" )
11 set_axis_assignment( 10, "mixture",  "reverse" )
12 set_axis_assignment( 11, "prop",     "reverse" )
13 set_axis_assignment( 12, "throttle", "normal" )
14
15 clear_all_button_assignments()
16 set_button_assignment( Thrustmaster + 0, "FlyWithLua/autopilot/set_autopilot_off" )
17 set_button_assignment( Thrustmaster + 2, "sim/view/chase" )
18 set_button_assignment( Thrustmaster + 3, "sim/view/chase" )
19 set_button_assignment( Thrustmaster + 4, "sim/general/zoom_in_fast" )
20 set_button_assignment( Thrustmaster + 5, "sim/general/zoom_out_fast" )
21 set_button_assignment( Thrustmaster + 6, "FlyWithLua/autopilot/activate_autopilot" )
22 set_button_assignment( Thrustmaster + 7, "FlyWithLua/autopilot/set_autopilot_off" )
23 set_button_assignment( Thrustmaster + 8, "sim/view/flashlight_red" )
24 set_button_assignment( Thrustmaster + 9, "sim/view/flashlight_wht" )
25 set_button_assignment( Thrustmaster + 10, "FlyWithLua/debugging/reload_scripts" )
26 set_button_assignment( Thrustmaster + 13, "FlyWithLua/Ingo/SetQNH" )
27 set_button_assignment( Thrustmaster + 16, "sim/general/up" )
28 set_button_assignment( Thrustmaster + 18, "sim/view/night_vision_toggle" )
29 set_button_assignment( Thrustmaster + 20, "sim/general/down" )
30 set_button_assignment( Thrustmaster + 22, "FlyWithLua/cockpit/toggle_HASI" )
31 set_button_assignment( Quadrant + 0, "sim/flight_controls/flaps_up" )
32 set_button_assignment( Quadrant + 1, "sim/flight_controls/flaps_down" )
33 set_button_assignment( Quadrant + 2, "sim/flight_controls/speed_brakes_up_all" )
34 set_button_assignment( Quadrant + 3, "sim/flight_controls/speed_brakes_down_all" )
35 set_button_assignment( XBox + 0, "sim/lights/nav_lights_toggle" )
36 set_button_assignment( XBox + 1, "sim/lights/beacon_lights_toggle" )
37 set_button_assignment( XBox + 3, "sim/lights/strobe_lights_toggle" )
38 set_button_assignment( XBox + 2, "sim/lights/landing_lights_toggle" )
39 set_button_assignment( XBox + 6, "sim/operation/dev_console" )
40 set_button_assignment( XBox + 7, "sim/operation/toggle_full_screen" )
41
42 set( "sim/joystick/joystick_pitch_nullzone", 0.0 )
43 set( "sim/joystick/joystick_roll_nullzone", 0.0 )
44 set( "sim/joystick/joystick_heading_nullzone", 0.0 )
45 set( "sim/joystick/joystick_pitch_augment", 0.2 )
46 set( "sim/joystick/joystick_roll_augment", 0.2 )
47 set( "sim/joystick/joystick_heading_augment", 0.2 )
48 set( "sim/joystick/joystick_pitch_sensitivity", 0.2 )
49 set( "sim/joystick/joystick_roll_sensitivity", 0.2 )
50 set( "sim/joystick/joystick_heading_sensitivity", 0.2 )
```

Warum wir diesen Aufwand betreiben? Wird ein neuer Joystick angesteckt, so können sich die Nummern der Tasten an den Joysticks verschieben. Wir müssen in so einem Fall jedoch nur noch die Zeilen 2 bis 4 anpassen, und schon stimmt die Konfiguration wieder.



6.6 Bedingte Programmierung

Wir können Befehle auch an eine Bedingung knüpfen. So wollen wir, wenn wir einen Helikopter fliegen, bestimmte Einstellungen automatisch verändern. Wir ergänzen unser Script um folgende Zeilen:

```
52 — all helicopter will get a unique setting as default
53 function set_helicopter_assignments()
54     set_axis_assignment(12, "collective", "normal")
55     set( "sim/joystick/joystick_pitch_nullzone",      0.0 )
56     set( "sim/joystick/joystick_roll_nullzone",       0.0 )
57     set( "sim/joystick/joystick_heading_nullzone",    0.0 )
58     set( "sim/joystick/joystick_pitch_augment",       0.0 )
59     set( "sim/joystick/joystick_roll_augment",        0.0 )
60     set( "sim/joystick/joystick_heading_augment",     0.0 )
61     set( "sim/joystick/joystick_pitch_sensitivity",   0.0 )
62     set( "sim/joystick/joystick_roll_sensitivity",    0.0 )
63     set( "sim/joystick/joystick_heading_sensitivity", 0.0 )
64     set( "sim/flightmodel/controls/parkbrake", 0.0 )
65 end
66
67 — MD902 Explorer (EXPL)
68 if PLANE_ICAO == "EXPL" then
69     set_helicopter_assignments()
70 end
71
72 — Huges 300 (Hu30)
73 if PLANE_ICAO == "Hu30" then
74     set_helicopter_assignments()
75     set( "sim/graphics/view/field_of_view_deg", 85 )
76 end
77
78 — Default Robinson R22 (r22)
79 if PLANE_ICAO == "r22" then
80     set_helicopter_assignments()
81     set( "sim/graphics/view/field_of_view_deg", 85 )
82 end
83
84 — Robinson R44 (RH44)
85 if PLANE_ICAO == "RH44" then
86     set_helicopter_assignments()
87     set( "sim/graphics/view/field_of_view_deg", 85 )
88 end
89
90 — Bell 206 Dreamfoil
91 if PLANE_ICAO == "B06" then
92     set_helicopter_assignments()
93     set( "sim/graphics/view/field_of_view_deg", 80 )
94 end
```

Zunächst wird in Zeile 53 bis 65 ein neuer Befehl `set_helicopter_assignments()` erzeugt. Dies machen wir durch das Schlüsselwort `function`, gefolgt von dem zu schaffenden Befehl mit seinen Argumenten in runden Klammern und dem Code, der ausgeführt werden soll. Die Definition endet mit den Schlüsselwort `end` in Zeile 65.

Die Zeilen 54 bis 64 werden so zum Inhalt des neuen Befehls, jedoch dabei nicht sofort ausgeführt. Mit Zeile 68 kommt Code, den der Compiler wieder direkt verarbeitet.



Zeile 68 beginnt mit dem Schlüsselwort `if`, dem eine logische Bedingung folgen muss, danach das Schlüsselwort `then`, gefolgt von dem Code, der ausgeführt werden soll, wenn die Bedingung zutrifft, und abgeschlossen schließlich durch das Schlüsselwort `end`.

Die Bedingung lautet `PLANE_ICAO == "EXPL"`. Durch dieses »Doppelgleich« ist die Bedingung immer dann zutreffend, wenn beide Seiten identisch sind, also hier der Inhalt der Variable `PLANE_ICAO` den Wert `"EXPL"` enthält, eine Zeichenkette.

Trifft dies zu, so wird der zuvor erschaffene Befehl `set_helicopter_assignments()` ausgeführt.

6.7 Code direkt ausführen

Befinden wir uns in X-Plane und drücken die Taste `Enter`, so gelangen wir in die Eingabezeile der XSquawkBox. Hier haben wir bereits die Frequenz von COM1 verstellt. Anstatt der beiden Slash, zum Verstellen der Frequenz, starten wir die Zeile mit einem `>`, dem »Größer« Zeichen. Danach können wir Lua Code eingeben und durch `Enter` direkt ausführen lassen. Versuchen wir es:

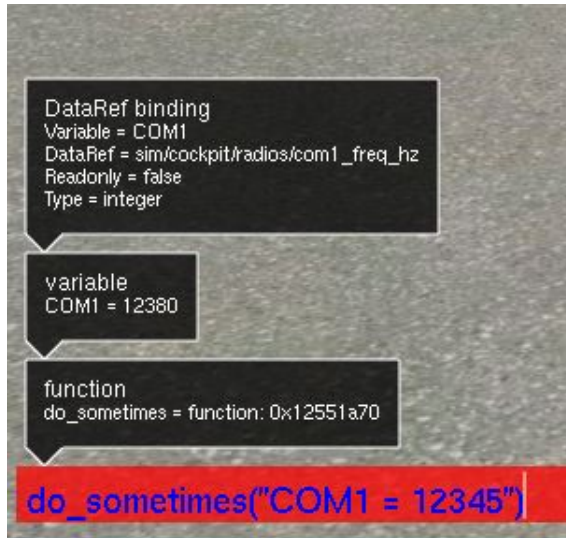
Enter `>print(PLANE_ICAO)` *Enter*

Sollten wir gerade einen MD902 fliegen, so wird uns nun als »Antwort« der Text `EXPL` ausgegeben. Wir können auf diese Weise die Variable `PLANE_ICAO` (und auch jede andere Variable) auslesen.

Der Nachteil ist jedoch, dass dies nur mit der Version 1.02 der XSquawkBox funktioniert. Leider ist die aktuelle Version zwar 64-bit tauglich (oder sagen wir mal sie stürzt nur selten ab), aber die Funktionen zur Kommunikation mit anderen Plugins wurden scheinbar übersehen. Wir können also weder das METAR auslesen, noch den Flugplan verändern, usw. Man kann so weiter mit der alten Version im 32-bit Modus fliegen, oder man nimmt die neue Version und verzichtet auf das Zusammenspiel zwischen FlyWithLua und XSquawkBox.

Hat man die XSquawkBox nicht installiert oder verwendet die aktuelle, fehlerbehaftete Version, so gibt es eine Alternative zum Ausführen von Code. FlyWithLua bietet ein Kommando `FlyWithLua/debugging/enter_code` an, das man einer Taste oder einem Joystickknopf zuordnen kann.

Dieses Kommando hat gegenüber der XSquawkBox Eingabezeile sogar einen Vorteil. Während der Eingabe wird diese geprüft und es werden Tipps in Form von Sprechblasen angezeigt.



6.8 Zusammenfassung

Mit diesem Wissen lassen die Joysticks und Tastaturen bereits musterabhängig automatisch einrichten, was eine Menge Arbeit erspart, wenn man mit stark unterschiedlichen Konfigurationen fliegen will.

Manche Flieger in X-Plane haben jedoch die Unsitte, keinen ICAO Code mitzuliefern. Dann kann man den Umweg über die Variable `PLANE_TAILNUMBER` nehmen. In dieser Variable findet man die Zulassungskennung des Flugzeugs, also das, was beim Auto das Nummernschild ist.

FlyWithLua ist ein so mächtiges Werkzeug, dass sich noch weitaus mehr in der automatischen Konfiguration umsetzen lässt. Mit diesem Handbuch sollen aber vor allem Nicht-Programmierer eine Chance bekommen, sich mit FlyWithLua das Fliegen zu vereinfachen.

Wer mehr mit Lua unternehmen möchte, der sollte sich das offizielle Manual genau durchlesen. Dort werden weitere Techniken erklärt, wie man X-Plane mit Lua programmieren kann.



7 DataRefs

7.1 Variablen im Simulator

Der Simulator X-Plane muss sich bestimmte veränderliche Werte (Variablen) merken. Dies geschieht an ganz bestimmten Stellen im Speicher. Die Stelle im Speicher wird durch ein sogenanntes Offset definiert, einem Zeiger in Form einer Zahl. Wer den Flugsimulator von Microsoft kennt, weiß vielleicht durch das Addon FSUIPC von diesen Offsets.

Zum Glück ist man bei der Programmierschnittstelle von X-Plane einen etwas einfacher zu merkenden Weg gegangen, man hat den Offsets Namen gegeben. Hinter den Namen sind die Referenzen zu den Speicherplätzen der veränderlichen Werte abgelegt. Man spricht deshalb von DataRefs.

Ein solches DataRef ist z. B. dieses: `sim/cockpit/electrical/landing_lights_on`

Eine Auflistung aller DataRefs findet man hier:

<http://www.xsquawkbox.net/xpsdk/docs/DataRefs.html>

7.2 Einzelwerte und Arrays

Die Zeiger können auf einen einzigen Wert zeigen. So findet man an der Speicheradresse des oben genannten DataRefs entweder die Zahl 1 oder die Zahl 0, je nachdem ob das Landelicht an oder aus geschaltet ist.

Die Außentemperatur am Flugzeug ist auch ein Einzelwert, die Zylinderkopftemperatur jedoch nicht. Das DataRef `sim/flightmodel/engine/ENG_CHT_c` zeigt auf eine Folge von acht Werten, die die Temperaturen von bis zu acht Zylinderköpfen speichern. Eine solche Folge von Werten nennt man ein Array.

Jeder Wert innerhalb eines Arrays hat einen Index, der als ganze Zahl, beginnend mit 0, die Position innerhalb der Reihe angibt. So finden wir bei `sim/flightmodel/engine/ENG_CHT_c` am Index 2 die Zylinderkopftemperatur des dritten Zylinderkopfes (0 = erster Wert, 1 = zweiter Wert, 2 = dritter Wert, usw.).

7.3 Der DataRefEditor

Um die Werte hinter den DataRefs anzuzeigen und zu verändern, nutzen wir das Plugin DataRef-Editor. Wir klicken auf `Plugins`, dann auf `Data Ref Editor` und schließlich auf `Show Datarefs`. Es erscheint ein Fenster wie dieses:



Dieses Fenster hat einen dunkleren, breiten Rand. An diesem Rand können wir durch Ziehen mit der Maus die Fenstergröße verändern. Durch Klick auf den Button `Close` wird das Fenster wieder geschlossen.

Sehr wichtig ist die Eingabezeile unten links, wo wir Text eingeben können. Der DataRefEditor filtert dann alle DataRefs, dessen Namenanfang mit unserer Eingabe überein stimmt. Durch Drücken der Taste `Tab` vervollständigt der DataRefEditor die Eingabe, ähnlich wie die Eingabezeile eines Betriebssystems (DOS-Box, Shell).

In der Auflistung sehen wir die Namen der DataRefs und dahinter ein Gleichheitszeichen und einen Wert. Befindet sich hinter dem Gleichheitszeichen noch eine eckige Klammer, dann handelt es sich um ein Array. Hier im Beispiel besagt die Folge `= [0/24]` : dass wir den ersten Wert eines Arrays mit insgesamt 24 Werten dargestellt bekommen.

Klicken wir nun auf den Wert, ändert sich die Anzeige zu einem Eingabefeld, und wir können den Wert direkt verändern, indem wir einfach einen neuen Wert eingeben. Probieren wir es aus und speichern unter

```
sim/cockpit2/switches/custom_slider_on
```

die Werte der ersten Elemente des Arrays auf 0 oder 1 und beobachten wir in der Außenansicht was passiert.

Wir wechseln zu den weiteren Elementen (Index Werten), indem wir die beiden Schieber vor und hinter dem Button `Close` nutzen. Der hintere ist für Einzelschritte, der vordere für größere Schritte (100'er).

Tatsächlich öffnen und schließen sich Türen. Dennoch scheint X-Plane mit 24 Elementen mehr Informationsspeicher anzubieten, als die meisten Flugzeuge tatsächlich nutzen. Die Auswirkungen sind also nicht immer gleich, sondern unterscheiden sich (manchmal) abhängig vom geladenen Flugzeug.



7.4 Startwerte für DataRefs

Auch Lua kann auf die DataRefs zugreifen. Dazu ist nicht einmal der DataRefEditor nötig. Öffnen wir nochmals unsere Konfigurationsdatei `zzz_Meine_Konfiguration.lua` und fügen wir am Ende folgendes an:

```
96 set( "sim/graphics/view/field_of_view_deg", 35 )
```

Danach laden wir alle Scripte neu.

Wie vermutlich leicht zu erkennen ist, haben wir den Blickwinkel auf 35° verändert, mit dem X-Plane die Welt darstellt. Wollen wir nun noch die ersten beiden Türen öffnen, so fügen wir weiter an:

```
97 set_array( "sim/cockpit2/switches/custom_slider_on", 0, 1 )
98 set_array( "sim/cockpit2/switches/custom_slider_on", 1, 1 )
```

Unser Flieger begrüßt uns nun beim Start immer mit offenen Türen, sodass wir leichter einsteigen können.

Das einzige Problem jedoch ist, dass wir nicht während eines Fluges die Scripte neu laden sollten, sonst öffnen sich die Türen unbeabsichtigt. Da wir immer Kalt&Dunkel starten, fügen wir eine bedingte Programmierung ein:

```
97 if get("sim/cockpit/electrical/battery_on") == 0 then
98     set_array( "sim/cockpit2/switches/custom_slider_on", 0, 1 )
99     set_array( "sim/cockpit2/switches/custom_slider_on", 1, 1 )
100 end
```

Mit `get()` lassen sich also auch DataRefs auslesen. Wollen wir ein Element eines Arrays auslesen, dann nutzen wir

`get("Name des DataRef", Index)`

Mit diesem Wissen lassen sich nun auch Dinge automatisch konfigurieren, die X-Plane eigentlich so nicht vorgesehen hat (wie ein Start mit offenen Türen). Praktischer als automatische Türöffnungen sind aber sicher die Konfiguration unterschiedlicher Blickwinkel, in Abhängigkeit vom geladenen Flieger.

X-Plane bietet noch tausende anderer DataRefs, die sich sinnvoll mit FlyWithLua manipulieren lassen. Finde sie heraus und steigere dein Erlebnis mit X-Plane!



8 Die Beispielscripte

FlyWithLua bringt eine Reihe von Scripten mit, die sich alle im Verzeichnis `Scripts (disabled)` befinden. Durch einfaches Kopieren der Dateien in das `Scripts` Verzeichnis kann man diese Scripte quasi »installieren«.

Hier sollen nun die Scripte kurz vorgestellt werden, die sich auch ohne Programmierkenntnisse nutzen lassen.

8.1 anti rollover brake.lua

Dieses Script erzeugt das Kommando `FlyWithLua/flight_controls/ARB`, das einem Knopf oder einer Taste zugeordnet werden kann. Wird dieser Knopf dann gedrückt, so beginnt das Plugin optimal zu bremsen.

Dazu werden, falls gesetzt, die Klappen eingefahren, damit mehr Druck auf die Reifen übertragen wird. Ist die Bremswirkung allerdings so groß, dass der Flieger nach vorn zu überschlagen droht, wird die Bremskraft sofort reduziert. Es handelt sich also um ein Anti-Überschlag-Bremssystem, welches die Landung mit einem kleinen Spornradflugzeug erheblich vereinfacht.

Außerdem wird nicht nur symmetrisch gebremst. Die beiden Vorderräder erhalten eine unsymmetrische Bremskraftverteilung, um die Lenkbewegungen zu unterstützen. Die Lenkunterstützung wird auch dann eingesetzt, wenn der Flieger rollt.

Zu allem Überfluss werden die Bremskräfte auch noch visuell durch farbige Balken angezeigt. Die gelben Balken zeigen die nicht symmetrischen Bremskräfte, der rote Balken die symmetrische Bremskraft.

8.2 automatic set qnh.lua

Man bekommt das neue Kommando `FlyWithLua/Ingo/SetQNH`, welches die Höhenmesser auf den aktuellen QNH Druck einstellt. Ehemalige Microsoft Simulator Piloten werden sich dieses Kommando auf die Taste B legen wollen, auf der X-Plane normalerweise die Bremsen bedient.

Wer es komplett narrensicher haben möchte, der kann im Untermenü `FlyWithLua Macros` den Punkt `automatic set QNH` anhaken. Dann zeigen die Höhenmesser immer die Höhe gemäß QNH an. Will man zurück zu Flugflächen, dann kann man den Haken während des Fluges jederzeit wieder entfernen.



8.3 automatic_cowl_flaps.lua

Dieses Script steuert die Kühklappen nach der Öltemperatur. Es steuert immer die ersten beiden Motoren, wer ein Flugzeug mit mehr als zwei Motoren automatisieren möchte, muss das Script entsprechend erweitern. Wer die Kühklappen lieber selbst regulieren möchte, der darf das Script nicht verwenden, da es sich nicht deaktivieren lässt.

8.4 autopilot infoline.lua

In der Außenansicht wird ein Text angezeigt, der die Einstellungen des Autopiloten wiedergibt. Wer selbstgebaute Schalter, Taster und Drehregler zur Steuerung des Autopiloten einsetzt, aber keine Anzeige in Form einer Hardware nutzt, kann mit diesem Script den Autopiloten in der Außensicht im Auge behalten.

8.5 call ATC on airport.lua

Dieses Script erzeugt Menüeinträge im ATC Menü von FlyWithLua und bietet zu jedem Menüpunkt auch ein Kommando, um es auf eine Taste zu legen. Die Menüeinträge sind:

```
call delivery controller on VATSIM
call ground controller on VATSIM
call tower controller on VATSIM
call approach controller on VATSIM
listen to ATIS on COM2
listen to arrival ATIS on COM2
call arrival tower controller on VATSIM
call arrival approach controller on VATSIM
```

Passend dazu gibt es diese Kommandos:

```
FlyWithLua/ATC/contact_DEL
FlyWithLua/ATC/contact_GND
FlyWithLua/ATC/contact_TWR
FlyWithLua/ATC/contact_APP
FlyWithLua/ATC/listen_ATIS
FlyWithLua/ATC/listen_arrival_ATIS
FlyWithLua/ATC/contact_arrival_TWR
FlyWithLua/ATC/contact_arrival_APP
```

Diese Kommandos stellen lediglich die Frequenz auf COM1 oder COM2 ein, mehr nicht. Dafür aber mit einer kleinen Automatik. So ermittelt FlyWithLua/ATC/contact_DEL zunächst den Flughafen, der uns gerade am nächsten ist. Nehmen wir an es wäre EDLP. Jetzt schaut das Script nach, ob ein Controller online ist, der EDLP und _DEL im Stationsnamen hat. Findet das Script einen solchen Lotsen, dann wird auf die Frequenz des gefundenen Lotsen gerastet. Ist



kein Delivery Controller online, so wird nach dem nächst höheren Rang gesucht, also nach einem Ground, Tower oder Approach Controller. Findet sich niemand, so schaltet das Script auf UNICOM 122,80 MHz.

Damit man im Anflug auf einen Flughafen bereits rechtzeitig Kontakt zum Approach oder Tower Controller aufnehmen kann (der nächste Flughafen ist oft ein kleines Flugfeld auf der Anflugroute), gibt es dazu eigene Kommandos, bei denen das Script den einzustellenden Flughafen aus dem Flugplan der XSquawkBox entnimmt.

Solange man nicht eines der Kommandos oder Menüpunkte selbst auswählt, beeinflusst das Script den Simulator nicht. Es kann also bedenkenlos installiert werden.

8.6 display clist.lua

Dieses Script sucht im Aircraft Verzeichnis nach der Datei `clist.txt` und wertet diese aus. Wird keine Datei dieser Art gefunden, so gibt das Script einen Hinweis aus und verhält sich ansonsten passiv.

Wird jedoch eine Datei `clist.txt` gefunden, so zeigt das Script die darin enthaltenen Checklisten an, sobald man mit der Maus an den rechten Rand des Bildschirms fährt. Berührt man den Rand, so werden die Titel der einzelnen Checklisten angezeigt. Fährt man an die passende Position, so wird der zugehörige Titel hell angezeigt. Bewegt man die Maus wieder vom Rand weg, so wird, falls ein Titel ausgewählt ist, die entsprechende Checkliste dauerhaft angezeigt.

Ein Klicken mit der Maus ist nicht notwendig. Man wird das aktuelle Fenster mit einer Checkliste wieder los, indem man den linken Rand kurz dort berührt, wo man keinen Titel auswählt.

Klingt mal wieder viel komplizierter als es ist — also einfach ausprobieren!

Passende Dateien `clist.txt` findet man im Internet zuhauf. Eine mögliche Quelle wäre:

<http://forums.x-plane.org/index.php?app=downloads>

Die Datei `clist.txt` muss sich im »Checklister« Format befinden. Alternativ oder ergänzend kann eine Datei `FWL_checklist.txt` im Aircraft Verzeichnis und/oder `global checklist.txt` im Script Verzeichnis angelegt werden. Diese beiden Dateien nutzen ein vereinfachtes Format, das in der mitgelieferten Datei `global checklist.txt` erklärt wird.

Letztere eignet sich hervorragend als globaler Spickzettel, in der Notizen hinterlegt werden können, die alle Flugmuster betreffen.

8.7 gpu.lua

Man bekommt den Menüeintrag `Ground Power Unit`, über den man die GPU einschalten kann, auch wenn der Flieger dies eigentlich gar nicht vorsieht. Ein blinkender Text informiert über eine angeklemmte GPU, damit man nicht mit einem am Flieger baumelnden Generator abhebt.



8.8 heading speed altitude instrument.lua

Ist dieses Script installiert, so wird in der Außenansicht ein Instrument angezeigt, das HASI. Das Kommando `FlyWithLua/cockpit/toggle_HASI` ermöglicht es, das Instrument unabhängig von der Anzeigeart (Innen- oder Außensicht) ein oder aus zu schalten.

Weil HASI (unter vielen anderen Informationen) ein Winddreieck anzeigt, wird es von VFR Piloten geschätzt, da man an Plätzen ohne Controller schnell sehen kann, aus welcher Richtung man die Piste anfliegen sollte.

Die letzte Zeile des Scripts sieht so aus:

```
214 do_every_draw('if (xp_view_is_external > 0) or alway_show_HASI then show_hasi(  
    SCREEN_WIDTH-330, SCREEN_HIGHT-340, 150) end')
```

Durch Veränderung der Zahl 150 kann man den Radius des Instruments in Pixeln verändern. Die beiden Werte davor bestimmen die Position auf dem Bildschirm. X-Plane beschreibt die Pixelposition mit einer X und Y Koordinate. Der Ursprung des Koordinatensystems liegt unten links. Oben rechts befindet sich das Pixel (SCREEN_WIDTH, SCREEN_HIGHT). Festgelegt wird der untere linke Punkt des HASI, der als Vorgabe 330 Pixel vom rechten und 340 Pixel vom oberen Rand entfernt liegt.

8.9 HeliTrim.lua

Zu diesem Script existiert ein eigenes Handbuch. Es beugt Krämpfen im Handgelenk vor, wenn man einen Helikopter mit einem »Würgestick« fliegt. Ein sehr nützliches Script für Piloten ohne Fußpedale.

8.10 Instrumententest.lua

Dieses Script zeigt in der Außenansicht ein Instrument an, auf dem Ladedruck und Drehzahl eines Kolbenmotors dargestellt werden. Es dient der Programmierausbildung, weniger dem praktischen Nutzen. Damit es überhaupt arbeitet, muss man in seiner Konfiguration folgendes einfügen:

```
show_MPPS_instrument = true
```

8.11 Joystick_Sensitivity.lua

Hat man dieses Script installiert, so wird die Linearität und Stabilität der Joystickachsen in Abhängigkeit der Geschwindigkeit (IAS) automatisch justiert. Das Flugverhalten fühlt sich so »geschmeidiger« an. Rollend am Boden bekommt man große Steuerausschläge durch kleine Auslenkungen. Hat man seine Reisegeschwindigkeit erreicht, werden die Steuerbewegungen am Joystick nur noch gefühlvoll übertragen.



Wer sowohl X-Plane als auch den MSFS kennt, wird dieses Verhalten als »microsoftiger« empfinden. An einem Joystick mit Rückstellfeder erreicht man mit der gleichen Kraft immer den gleichen Ausschlag. Bei einem Flieger mit mechanisch gekoppelten Steuerflächen erreicht man mit der gleichen Kraft jedoch abhängig von der Windgeschwindigkeit, die eine Gegenkraft zur Steuerbewegung erzeugt, unterschiedliche Ausschläge der Steuerflächen.

Dieses Script wird man lieben oder hassen, probiere es aus und finde deinen Weg.

8.12 **keystroke_sniffer.lua**

Es erzeugt den Menüpunkt `Show keystroke numbers`. Damit lassen sich Tastencodes auslesen. Dieses Script ist nur für Programmierer interessant, die mit Lua Tasten direkt abfragen möchten. Weil es nichts verändert, solange man nicht auf den Menüpunkt klickt, kann es bedenkenlos installiert werden.

8.13 **lean_helper.lua**

Immer wenn man die Gemischverstellung betätigt, wird für eine kurze Zeit die Abgastemperatur der ersten beiden Motoren angezeigt. Eine Maximalanzeige zeigt den Höchstwert seit dem Anzeigen an, ähnlich wie ein Peak-Level an einem Audio-Verstärker. Kleinere Striche zeigen die Temperatur relativ zum Höchstwert und erleichtern so das Leanen.

Das Script wirkt nur bei Fliegern, die in den letzten Zeilen des Scripts definiert sind. Man muss es daher um die Flieger ergänzen (`if` Argument erweitern), bei denen man die Lean Hilfe sehen möchte.

8.14 **mixture_spline.lua**

Bei den im Script festgelegten Flugmustern erhält die Achse für die Gemischverstellung ein nichtlineares Verhalten. Ist das Gemisch in der Aus-Stellung und wird der Mischerhebel am Joystick/Quadrant auf die Hälfte seines Weges geschoben, so ist der Gemischhebel für X-Plane bereits bei 75%. Die letzte Hälfte des Weges regelt nun das letzte Viertel des Gemischreglers in X-Plane, was ein exakteres Einstellen ermöglicht.

Das klingt viel komplizierter als es ist, man sollte es einfach ausprobieren. Möchte man weitere Flieger mit diesem Verhalten ausstatten, so muss man die ersten Zeilen des Scripts entsprechend anpassen.



8.15 OpenDoors.lua

Erzeugt den Menüpunkt `Open all doors`, der alle Türen gleichzeitig öffnet oder schließt. Außerdem startet man mit geöffneter Pilotentür, wenn man den Simulator Kalt&Dunkel hochfährt.

Eine harmlose Spielerei mit begrenztem Nutzen.

8.16 program GPX files into FMC.lua

Ein Highlight unter den Scripten, dass man ohne zu zögern installieren kann, denn es verändert am Simulator nichts.

Man muss allerdings folgenden Ordner erzeugen:

Ort wo X-Plane liegt/X-Plane 10/Resources/plugins/FlyWithLua/Scripts/GPX

Jetzt kann man GPX Dateien in diesem Verzeichnis ablegen und das Script erzeugt für jede Datei einen Eintrag im ATC Menü von FlyWithLua. Wählt man dann den Menüeintrag zu einer GPX Datei aus, so werden alle Wegpunkte aus der GPX Datei in den FMC übertragen.

Das erlaubt es uns, mit der Webseite FL95.de einen VFR Flug zu planen. Wir gehen dabei auf den linken Reiter `Laden Speichern Drucken` und klicken auf `GPS-Export`. Wichtig ist, dass GPX als Ausgabeformat ausgewählt ist (der Standard bei FL95).

GPX ist ein weit verbreitetes Format. Auch das App [VFRnav](#) auf einem Smartphone oder Tablet speichert Flugplanungen in diesem Format. So kann man auf der Arbeit in der Mittagspause schon am Handy planen, wo man abends entlang fliegen möchte.

Viele kleinere Flieger zeigen die Wegpunkte zu einem »roten Faden« verbunden im GPS Instrument an, auch wenn gar kein FMC im Flieger verbaut ist.

8.17 QNH_helper.lua

Zeigt den eingestellten Druck am Höhenmesser in metrischen und imperialen Einheiten an, wenn er verändert wird. Eine sehr nützliche Hilfe, auf die man nicht verzichten sollte.

Außer der automatischen Anzeige, die nach wenigen Sekunden wieder verschwindet, verändert das Script nichts.

8.18 reload scenery.lua

Lädt die Szenerie neu, ohne den Flug zurück zu setzen. Praktisch für Szenerieentwickler. Ausgelöst wird das Neuladen über einen Menüpunkt `Reload the scenery`.



8.19 show last metar and next airport.lua

Macht genau das, was man vermutet, durch eine Textzeile, die dauerhaft oben links eingeblendet wird. Angezeigt werden neben dem METAR und dem nächstgelegenen Flughafen auch der ICAO Code des Flugzeugs und die aktuelle Framerate.

8.20 SimpleTrim.lua

Am Beginn des Scriptes muss eine Joystick-Knopfnummer eingetragen werden. Drückt und hält man dann diesen Knopf, so werden die Achsen abgekoppelt, in Zentralstellung gebracht, und die bisher zum Steuern genutzten Achsen verändern nun die Trimmung. Dadurch kann man das Flugzeug trimmen, als ob man es fliegen würde. Ist eine Trimmung gefunden, so lässt man den Knopf (und danach den Joystick) wieder los.

Auch das klingt wieder schlimmer als es ist. Wichtig ist nur, dass man den Knopf durch verändern des Scripts festlegen muss, nicht durch Zuordnung eines Kommandos.

8.21 transponder_helper.lua

Über den Menüpunkt *Automatically set Transponder* kann man wählen, ob das Script über den Status des Transponders wachen soll. Hebt man ab, ohne den Transponder von Standby auf Modus-C zu stellen, so erledigt das Script dies automatisch. Bei der Landung wird der Transponder entsprechend wieder zurück gestellt.

Für Faulpelze und Schussel ein hilfreiches Script, will man Stress mit den Controllern vermeiden.

8.22 VFR autopilot helper.lua

Es erzeugt folgende beiden Kommandos:

```
FlyWithLua/autopilot/activate_autopilot  
FlyWithLua/autopilot/set_autopilot_off
```

Legt man beide auf passende Tasten, so kann man damit den Autopiloten ein- und ausschalten. Jedoch anders als mit den Kommandos von X-Plane. Wird der Autopilot mit dem neuen Kommando gestartet, so schaltet das Script den Autopiloten in den Modus *Höhe und Richtung halten*, wobei die aktuelle Höhe und Richtung eingestellt werden.

Ist der Autopilot aktiv, so werden die Steuerachsen am Joystick abgekoppelt und in Neutralstellung gebracht. Mit der Pitch-Achse kann man nun die Höhe im Autopiloten verändern, mit der Roll-Achse den Kurs. Man erhält eine Art notdürftiges FlyByWire System (poor man's FBW).



Das Verhalten lässt sich über den Menüpunkt `use VFR autopilot tweak` ein- oder ausschalten. Es ist bei Flugzeugen über 5,7 t zulässigem Gesamtgewicht ausgeschaltet, bei leichteren Modellen eingeschaltet.

In X-Plane ist es so, dass der Autopilot auch dann funktioniert, wenn das Muster gar keinen Autopiloten verbaut hat. Dann sollte man sich den Autopiloten Zustand mit dem Script `autopilot infoline.lua` anzeigen lassen.

8.23 VFR weather.lua

Das Königsscript unter den Scripten für VFR Piloten. Es manipuliert das Online-Wetter von VATSIM so, dass (wenn man es will) immer VMC herrscht. Allerdings wird das Wetter nur so weit abgemildert, dass man gemäß Definition VFR fliegen darf. Damit unterscheidet es sich maßgeblich vom manuellen Einstellen von CAVOK, da zum Beispiel die Windrichtung beibehalten wird. Lediglich die Windstärke wird auf 25 bis 40 Knoten reduziert (je nach Höhe).

Man muss sich also nie wieder zwischen Schleudergang oder Langeweile entscheiden, sondern bekommt ein ausgewogen dosiertes Stück Wetter serviert, bei dem auch die Höhenangaben wieder einen Sinn ergeben, denn der Luftdruck wird nicht verändert.

Über den Menüpunkt `force online VFR` entscheidet man, ob man die Hilfe dieses Scripts in Anspruch nehmen will. Sitzt man in einem »Pott« der Alpha, Bravo oder Charlie Klasse, ist die Wettermanipulation ausgeschaltet, da man diese Flieger meist IFR fliegen wird.

Über den Menüpunkt `set Visual Meteorological Conditions` kann man die Anpassung auch einmalig auslösen, selbst wenn das Script inaktiv sein sollte.

8.24 XSB_helper.lua

Man bekommt einen Menüpunkt `create Traffic Pattern flightplan`, der den Flugplan der XSquawkBox vorausfüllt und danach anzeigt.

Es kann passieren, dass man den Menüpunkt anklickt, der Flugplan aber trotzdem nicht richtig ausgefüllt ist (als hätte XSquawkBox das Ausfüllen nicht mitbekommen und zeigt noch alte Werte). Dann muss man den Flugplan schließen ohne ihn nach VATSIM abzusenden und erneut öffnen. Danach sollten im zweiten Anlauf alle Einträge so verändert sein, dass man zu einer Platzrunde starten kann.

Die Angaben zum verfügbaren Sprit sind manchmal so abenteuerlich geschätzt, dass man diesen Eintrag immer kontrollieren sollte. X-Plane selbst schätzt übrigens auch nicht viel besser als FlyWithLua.



9 Hilfe bei Problemen

Sollten sich Probleme ergeben, so stehen die Entwickler von FlyWithLua zwar zur Verfügung, und leisten Unterstützung in deutscher oder englischer Sprache, jedoch ist FlyWithLua ein »Hobbyprojekt«, mit dem die Entwickler kein Geld verdienen. Daher erwarte keine Antwort auf eine Mail (die Adresse findet man am Ende des offiziellen Handbuchs) innerhalb weniger Stunden. Es kann auch mal Tage dauern, bis eine Antwort gegeben wird, oder eine Frage wird gar nicht beantwortet (weil die Mail vielleicht im Spam Filter verhungert ist).

Wer sich mit einem Problem an die Entwickler wendet, der muss immer die Datei `log.txt` und `FlyWithLua_Debug.txt` als Anhang mitsenden, sonst wird die Anfrage direkt ohne Rückmeldung ignoriert. Beide Dateien befinden sich im Hauptverzeichnis von X-Plane.

Eine sehr gute Idee ist es auch, zunächst im Unterforum für X-Plane der VATSIM Germany zu fragen.

<http://board.vacc-sag.org/42/>

Das FlyWithLua-Team wünscht viel Freude beim Fliegen mit X-Plane!

Carsten Lynker

Hauptentwickler FlyWithLua