

# Flight Assistant User Guide

Beta v1.5

Developer: Degas

Exclusive Release for <https://www.gta5-mods.com/>

Download page: <https://www.gta5-mods.com/scripts/flight-assistant>

## Index

<b>1.</b>	<b>Introduction.....</b>	<b>3</b>
<b>2.</b>	<b>Aircraft States.....</b>	<b>3</b>
<b>3.</b>	<b>Aircraft Mode.....</b>	<b>4</b>
<b>4.</b>	<b>Sounds.....</b>	<b>5</b>
<b>5.</b>	<b>Interface.....</b>	<b>7</b>
<b>6.</b>	<b>Parameters.....</b>	<b>8</b>
6.1.	config.xml.....	8
6.1.1.	Keys.....	8
6.1.2.	Function Keys.....	9
6.1.3.	Refresh Rate.....	9
6.1.4.	Allowers.....	9
6.1.5.	Common.....	10
6.1.6.	Rotation.....	10
6.2.	display.xml.....	11
6.3.	model.xml.....	12
6.4.	airport.xml.....	14
6.5.	keyboard.xml.....	16
<b>7.</b>	<b>Collision System.....</b>	<b>16</b>
<b>8.</b>	<b>Landing System.....</b>	<b>16</b>
<b>9.</b>	<b>Commands.....</b>	<b>17</b>
9.1.	Command Argument Type.....	17
9.2.	Regular Commands.....	17
<b>10.</b>	<b>Setting XML Parameters.....</b>	<b>22</b>
<b>11.</b>	<b>Memory and Action.....</b>	<b>23</b>
11.1.	Actions.....	24
<b>12.</b>	<b>Hints &amp; Tips &amp; Headache Avoidance.....</b>	<b>25</b>

## 1. Introduction

Welcome to *Flight Assistant* script manual. This manual intends to give an overview of how the script works, what are the meanings of all the parameters and a glimpse of what can be achieved. Keep in mind that this script is intended for planes only. Also, the use of *VTOL* is not supported.

The aim of this script is to make piloting freighter and/or passenger a tiny bit more immersive, where you can have a great range of features according to aircraft's manufacturer and even personalised company announcements. This doesn't mean military fighters are excluded, they were just not the focus.

As you may already be aware, there is a long way to go on reading this file, but do not let yourself be overwhelmed with it, take a deep breath and go steps by steps. This manual was built following a progressive approach, from understanding the basic concepts to making your own chain of commands.

It is highly recommended to read a chapter and check it out in-game to see how it works instead of reading it all in one go. By the end of this manual, you should be able to grasp the key features this script has to offer.

The *FlightAssistant.dll* and *FlightAssistant.xml* file must be placed inside *script* folder on your game main folder. Also, all sub-folders must be placed in there.

### Requirements:

ScriptHookVDotNet3.dll v 3.1.0 and required files and distribution.

Available at (<https://github.com/crosire/scripthookvdotnet/releases>)

### Developed and tested on:

Windows 10 Pro 64-bit.

Processor intel i5 3.60GHz Coffee Lake.

8GB Ram Single-Channel 1604MHz.

GPU 2017MB NVIDIA GeForce GT 710.

SSD Storage.

Tested on: GTAV Build 877.

FPS with this script:  $\pm 25$  fps

FPS without this script:  $\pm 25$  fps

## 2. Aircraft States

The aircraft can have one out of six states at a time, those states define specific behaviours and activate or suppress different set of alarms. Each state can lead to another if requirements are met. Table 1 describes the basic transitions which can be induced by the player or by parameters such as speed, height, distance and others.

States	Characteristic
<i>Taxiing</i>	Speed is restricted by Taxiing Speed. Can change to <i>Takeoff</i> via <TOGA> hotkey.
<i>Takeoff</i>	Speed is not restricted. Changes to <i>Taxiing</i> when accelerated backwards. Changes to <i>Taxiing</i> via <TOGA> hotkey when Speed $\leq$ Taxiing Speed while on ground. Changes to <i>Aborting Takeoff</i> via <TOGA> hotkey when Speed > Taxiing Speed. Changes to <i>Aborting Takeoff</i> when losing speed while charging the runway. <sup>1</sup> Changes to <i>Aborting Takeoff</i> if a vehicle blocks the way ahead (auto mode only).

	Does <b>not</b> change status to <i>Aborting Takeoff</i> if V1 call was made.
<i>Aborting Takeoff</i>	Speed is not restricted. Changes to <i>Taxiing</i> when aircrafts halt. Changes to <i>Takeoff</i> if player keeps thrust and goes airborne.
<i>Flying</i>	Speed is not restricted. Changes to <i>Landing</i> via <TOGA> hotkey. Changes to <i>Landing</i> if touches the ground with landing gear deployed.
<i>Landing</i>	Speed is not restricted. Changes to <i>Taxiing</i> if Speed $\leq$ Taxiing Speed. Changes to <i>Going Around</i> via <TOGA> hotkey when Height $\leq$ Landing Height. Changes to <i>Flying</i> via <TOGA> hotkey when Height > Landing Height. Changes to <i>Going Around</i> when positive thrust after touchdown. Changes to <i>Going Around</i> if hover for too long above the runway. Changes to <i>Going Around</i> if runway is occupied (auto mode only).
<i>Going Around</i>	Speed is not restricted Changes to <i>Flying</i> if Height > Landing Height. Changes to <i>Taxiing</i> if Speed $\leq$ Taxiing Speed while on ground.

Table 1 – State transitions.

<sup>1</sup>If another vehicle is identified behind the aircraft, it **will not** reject take off, in that case the aircraft will try going airborne as soon as possible, forcing the V1 callout. It will force V1 callout if Speed is too high to stop.

### 3. Aircraft Mode

There are three distinct ways to interact with the aircraft: manual, semi and auto. Each of them has their pros and cons, so in the end of the day is all about your style.

The manual mode is the closest to piloting the aircraft without the script, but things can not be easy, right? When trying to move backwards the player must manually activate the reverse in order to perform the powerback. In this mode the player can ignore all messages and warnings as the control unit will not impose any correction to the movement.

Semi-automatic mode (or semi mode for short) makes piloting more comfortable, it does not remove responsibilities from the player but easy up on the tasks at hand, like deploying reverses when needed, retracting landing gear after take off or go-around.

At last, the auto mode extends the perks of semi mode and take control of the aircraft to bring it to safety, avoiding collision with the ground or with other aircrafts (or at least try to). It also automatically applies brakes and reverse power. For some players the auto mode might become inconvenient, as for example, during a fast descent to catch the glide slope the Sink Rate Avoidance Manoeuvre may put the entire approach in check.

Characteristic	Manual	Semi	Auto
Play Alarms	YES	YES	YES
Sets Lights/High Beams	NO	YES	YES
Deploy Landing Gear	NO	NO	YES
Retract Landing Gear	NO	YES	YES
Deploy Reverse	NO	YES	YES
Easy Powerback	NO	YES	YES

Auto-Throttle, Trim pitch/Roll	NO	YES	YES
Climb / Turn command	NO	NO	YES
Landing/Aborted Take off Brake	NO	NO	YES
Landing Adjust	NO	NO	YES
Take off Adjust	NO	NO	YES
Unable Take off if hazardous	NO	NO	YES
Collision Avoidance	NO	NO	YES
Allow Landing Gear During Flight	YES	YES	NO
Allow Reverse During Flight	YES	YES	NO
Allow All Engines Off During Flight	YES	NO	NO
Corrects Engine Unbalance	NO	NO	YES
Saves Fuel When Taxiing	NO	NO	YES

Table 2 – Aircraft Mode Capabilities.

#### 4. Sounds

Sounds are optional and work fine with mp3 and wav files, other extensions were not tested. Each model can have their own set of sounds. It is worth to mention that all sounds of a set must have the same extension. In case the player does not want a specific alert (i.e., bank angle warning), then all it takes is simply remove the file from the folder or not name it (can simply rewrite with an underscore in front of the name).

The files have no restriction regarding length, but it must be taken into account that tracks are not pre-loaded, therefore they are loaded and executed every time when played, which may cause a slight delay between visual alert and the associated sound. It is also noteworthy that all files must be written in lowercase letters and without space or any kind of special character, this rule also applies to folder names.

The sounds are managed on a separated thread, if the player pops up game's main menu, the alerts that were playing in that moment will remain playing until the track is over.

Sound	Loop Code	Function
10, 20, 30, ..., 2500	height	Height callout when landing, it uses <Height – Runway Height> as reference or <Height – Surface Height> if no runway was selected.
bank_angle	bank	Alerts when  roll  > 30, it plays every 5° above the limit.
beep_1	beep 1	Alerts when fuel is below minimum level, plays once.
beep_2	beep 2	Notify state changes that may require attention or when stabiliser is turned off.
beep_3	beep 3	Alerts when all engines are off while not in Taxiing state.
beep_4	beep 4	Alerts when reverse is open during flight.
clear_of_conflict	clear	Notify when collision avoidance manoeuvre ended.
climb	climb	Notify when aircraft is more than 1.5° below the glide slope, but still far from runway (>1000m).
climb_now	climb now	Notify when aircraft is 1.5° below the glide slope and close to runway (<1000m).

descent	descent	Notify when aircraft is 10° above the glide slope, but still far from runway (>1000m).
descent_now	descent now	Notify when aircraft is 3° above the glide slope, and close to runway (<1000m).
extra1	extra1	Any sound the player wants, can be associated with a hotkey
extra2	extra2	Any sound the player wants, can be associated with a hotkey
extra3	extra3	Any sound the player wants, can be associated with a hotkey
fire_alarm	fire	Continuous when an engine is on fire.
flap	flap	Sound played when flaps/landing gear are deployed or retracted.
gear_alarm	landing gear	Continuous sound to alert the landing gear is deployed while on Flying state.
glide_slope	glide	Notify that a runway was set for landing.
minimum	minimum	Notify that minimum decision height for landing was reached.
minimum_approaching	apprminimum	Notify that the aircraft is 100 height unit above minimum.
monitor_v_speed	v speed	Continuous alert indicating high descend with Height > 1500m.
retard	retard	Notify that you are close to the ground and should cease any throttle up (plays regardless of no throttle up).
sink_dont	don't sink	Notify your take off Vspeed is low.
sink_rate	sink rate	Continuous alert indicating high descend with Height < 1500m.
stall_alarm	stall	Continuous alert that your Angle of Attack is high enough to stall.
terrain	terrain	Continuous alert indicating imminent collision with the ground.
terrain_caution	terrain caution	Continuous alert indicating your height is between 100 e 50m.
too_low_gear	low gear	Continuous alert indicating your height is below 50m and no landing gear is deployed while landing.
too_low_terrain	low terrain	Continuous alert indicating your height is below 50m.
traffic_traffic	traffic	Continuous alert indicating that the aircraft is in route of collision with another vehicle, or that the runway you selected is occupied while you are also on it.
V1	v1	Notify that V1 mark during take off was reached, therefore it is no longer safe to abort the procedure.

Table 3 – Available Sounds.

The extra 1, extra 2 and extra 3 sound files work differently from the rest of the set as a command can be issued to change the file name to any other file in a subfolder from base folder.

## 5. Interface

Information is grouped and displayed in sets of windows that can be activated/deactivated on the run. After a while, an experienced player could easily land the aircraft just by looking at those windows/displays.



Figure 1 – Interface sample.

- 1- Extended System: Display secondary information regarding aircraft condition, being fuel level and aircraft integrity perhaps the most important values in this group.
- 2- System: Display relevant information for navigation such as aircraft's rotations (pitch, roll), height (relative to sea level) and both ground speed and vertical speed.
- 3- ECAM: Displays messages regarding aircraft condition and behaviour, as example, when an aircraft has a stall, ECAM will advise the pilot to lower the nose (or fix roll first in case of banked angle) and increase thrust, if the name ECAM on top of the box becomes red, it means there are more than five messages to be displayed. The colour scheme reflects the category of messages: blue is informative, green is procedure information, orange warnings, red critical messages and white for control mode change.
- 4- Weather Report: A summary of current weather, the field weather does not display the name of current weather, but the group it belongs, for example: Rain, Storm and Clearing belong to "Rain" group, Foggy, Halloween and such as "Fog", and so on. The field Wind informs the angle of the wind in relation to the aircraft.
- 5- Visual Alerts: Visual alerts that pops up, simulating warning lights in the cockpit.
- 6- Applied Force: Shows the input control and intensity that the aircraft is applying (used mainly for debug when developing/adjusting auto control).
- 7- Approach Helper: Helps increase situational awareness, the information displayed changes regarding being airborne or not, as well which stage of landing or take off the aircraft is. *Parallel Angle* and *Center Align* are the angle of the aircraft towards the runway and distance to the centre of the runway respectively. When landing, *Glide Slope* displays your angle in relation to runway's glide slope line.

- 8- Engine Number: Displays the status of the engines: green when is up and running, red when is off, yellow when is destroyed yet on, purple when destroyed and off. They can have a gauge that displays the rotation of the engine.

There are other sets of windows, activated by key bind, that displays model configuration, successfully loaded sounds, key binds and such (check chapter 6.1.1 and look for *displayConfig*).

## 6. Parameters

This section covers all parameters from *XML* files you can find on base folder. Base folder must be set on the *FlightAssistant.xml* file that is kept together with the .dll file. Both files must be kept at scripts folder that is inside your game folder. Base folder information is essential to find remaining files applied in this mod.

The *FlightAssistant.xml* file also brings the option to load Simaoelis JDeezNutz's London City airport when the script loads. London City airport can be downloaded at <https://www.gta5-mods.com/maps/london-city-airport-beta>.

### 6.1. config.xml

This section covers configuration regarding key binds, display of particles and windows, selection of keyboard, general aircraft parameters, and engine rotation behaviour.

#### 6.1.1. Keys

Although one of the goals was to use the lowest amount of key binds as possible, a few were still applied in order to make things more accessible.

Key	Description
scriptDisable	Turns script on/off.
commandInput	Opens text box to insert command.
disableSound	Enable/disable alert sounds.
TOGA	Changes aircraft's state.
controlChange	Changes aircraft operation mode.
deployReverse	Deploy/retract reverse.
lockReverse	Locks reverse on current state.
displayConfig	Opens windows displaying current configuration, use left/right arrow keys to browse through the windows.
searchILS	Enable/Disable valid ILS in range.
resetILS	Reset ILS search.
engineCheck	Allows testing aircraft's engine rotation
rotationUp	Increase rotation if engineCheck is active.
rotationDown	Decrease rotation if engineCheck is active.
debug	Enable/disable debug visualization. When active, it's possible to see Glide Slope lines, PAPI lights, collision scan range and transmission sphere even when on ground.
playExtra1	Plays the sound file extra1, if available.
playExtra2	Plays the sound file extra2, if available.
playExtra3	Plays the sound file extra3, if available.

Table 4 – Key binds.



### 6.1.2. Function Keys

Function keys are keys which commands and memories can be assign to execute when pressed. Each function key (named fKey, with capital k) has two fields: *shortcut* and *value*.

Shortcut defines the key combination to activate the associated command, while value defines the command or memory to execute.

*I.e: <fKey shortcut="Ctrl + I" value="engine off; /seatbelt">*

In the example above, when the user press *control* key and *i* simultaneously, the aircraft will try to shut down the engines and execute the seatbelt memory (plays the seatbelt sound).

### 6.1.3. Refresh Rate

There are routines frequently being executed that are costly to the computer or that do not require to run at higher frequency. Avoiding unnecessary calculation helps preserving the FPS. Therefore, the *refreshRate* node contains a set of routines that can have their execution period adjusted. Keep in mind that setting periods too high may compromise features that relies on data generated from those routines.

Item	Unit	Description
statusUpdate	ms	Rate in which aircraft updates all data regarding state, state change, fuel management, runway validation, command interpretation/execution.
collisionScan	ms	Rate in which selected runway or nearest runway is scanned looking if it's in use, also checks for collision against the ground or another aircraft. Depending on scan groundScan and airScan, this routine can become quite costly.
engineSpin	ms	Rate in which engine rotation value is updated.
runwayAnimation	ms	Rate in which arrows and parallel lines move. The cost is to calculate the movement is low.

Table 5 – Refresh rate fields.

### 6.1.4. Allowers

Allowers refer mostly, but not exclusively, to enabling or disabling display features.

Item	Description
script	Allows script run on game's start.
movement	Allows reverse control, movement/speed management. When disabled, several features are compromised.
referenceline	Enable/disable reference line being visible.
ils	Allows glide slope and PAPI lights features.
ecam	Allows ECAM display.
system	Allows System display.
extendedSystem	Allows Extended System display.
weather	Allows Weather Report display.
sounds	Allows alert sounds.
approachHelper	Allows Approach Helper display.
appliedPower	Allows Applied Power display.
alertBox	Allows warning box display.
collisionChecker	Allows scans for collision and runway occupation.
engineNumber	Allows display the engine status and associated gauge.
particles	Allows fuel dump and reverse particles.

Table 6 – Allowers.

### 6.1.5. Common

Item	Description
keyboardStandard	Selects the keyboard profile to use from <i>keyboard.xml</i>
taxiSpeedMulti	Positive float value that sets maximum taxi speed for all aircrafts following the formula: <i>Max Taxi Speed = 30kts * taxiSpeedMulti</i>
minimumsCoef	Positive float coefficient that sets decision height (Minimums) callout when landing for all aircrafts: <i>Minimums = 250*minimumsCoef</i> <i>ApproachingMinimums = Minimums + 100</i> P.S.: 250 is based on player's chosen height unit (feet or metre). If feet, then 250ft*minimumsCoef; if metre was chosen then 250m*minimumsCoef

Table 7 – Miscellaneous.

### 6.1.6. Rotation

This section describes the base rotation and increment ratio. Idle fields refer to base/minimum rotation without throttle up, being the range between 0 and 100. Throttle down do not affect rotation, unless in situation the reverse is deployed or movement feature is turned off. All those fields must be positive value.

Item	Description
idleTaxi	Base rotation while taxiing
idleTaxiForwardLimit	Maximum rotation while taxing without reverse deployed.
idleTaxiReverseLimit	Maximum rotation while taxing with reverse deployed.
idleAir	Base rotation while airborne.
idleFlare	Base rotation while flaring prior to landing.
idleTakeoff	Base rotation during take off.
idleGearDown	Base rotation while airborne with gear down. The idea is to simulate the feeling of throttle up to compensate gear drag without applying real thrust.
autoTakeoffRotationCharge	The rotation threshold that must be met before auto mode apply thrust for take off.
incrementTaxi	Rate which rotation will increase/decrease while taxiing without reverse deployed.
incrementTaxiReverse	Rate which rotation will increase/decrease while taxiing with reverse deployed.
incrementTakeoff	Rate which rotation will increase/decrease when taking off, while rotation is below autoTakeoffRotationCharge.
incrementTakeoffHigh	Rate which rotation will increase/decrease when taking off, while rotation is equal or higher than autoTakeoffRotationCharge.
incrementEngineStartGround	Rate which rotation will increase when <i>homing</i> <sup>1</sup> during engine start up while on ground.
incrementEngineStartAir	Rate which rotation will increase when <i>homing</i> <sup>1</sup> during engine start up while in air.
incrementGeneral	Rate which engine will increase/decrease in any other situation.

Table 8 – Rotation management.

<sup>1</sup>When all engines are turned off, the next time they are turned on the rotation must reach at least 50% so the aircraft can move properly, that is part of game's mechanism, but since idle rotation on vanilla is set to 50%, such behaviour is mostly not noticed.

## 6.2. display.xml

This file sets the position of display windows and display parameters as text scale and measure units.

Item	Description
textScale	Scales the font size, hopefully it can be used to set up the texts nicely according to screen resolution.
alertBox	Sets the position offset of the Visual Alerts (item 5 on interface) on the screen. Layout determines if the boxes will be displayed in horizontal (side by side) or vertical (stack like). Style determines the way it will be displayed: In case of horizontal layout, style can be Left-to-Right, Middle/Center or Right-to-Left, vertical on the other hand can have Top-Down, Middle/Center or Bottom-Up. P.S.: just the first letter is enough to identify layout (H/V) or style (L,M,C,R,T, B).
approachHelper	Sets the position offset of Approach Helper (item 7 on interface) on the screen.
ecam	Sets the position offset of ECAM (item 3 on interface) on the screen.
system	Sets the position offset of System (item 2 on interface) on the screen.
extendedSytem	Sets the position offset of Extended System (item 1 on interface) on the screen.
weatherReport	Sets the position offset of Weather Report (item 4 on interface) on the screen.
appliedPower	Sets the position offset of Applied Power (item 6 on interface) on the screen.
engineNumber	Sets the position offset of Engine Number (item 8 on interface) on the screen.
height	The measure unit to be used, can be metre or feet. Just the initial letter is enough to identify; i.e F or M
speed	The measure unit to be used, can be knots or kilometres/h, the latter is not sure to be applied on associated displays.
distance1	The measure unit for long horizontal distance, can be nautical miles (nm), or kilometres (km).
distance2	The measure unit for short horizontal distance, can be metres (m), feet (ft) or statute mile (sm).
temperature	Measure unit for engine temperature, can be Celsius (c) or Fahrenheit (F). Engine temperature affects nothing in this script.
source	the <b>folder</b> where the gauge and needle images are located. Set the path after the base folder path. For example, if base folder is scripts/abcd, and your gauge and needle are in scripts/abcd/mypics, then source field should be mypics. The script will look for the files named as Gauge.png and Needle.png on selected folder.
size	Sets the scale of both the gauge and needle.
numberAdjust	sets an offset to the engine number.
upperLimit	sets in which angle the gauge ends.
needleAdjust	sets the needle starting angle.

Table 9 – Display fields.

### 6.3. model.xml

This section provides explanation regarding aircraft's parameters.

Item	Description
plane	Contains the display name of the model, also helps navigation on the xml file as spawn name does not make clear what model it is about.
spawnName	Name used to spawn the aircraft by trainer. Without this field it will not be able to match the model when you enter the aircraft.
hash	Identifier generated using spawnName, it is compared with the aircraft the player is piloting. This field is filled automatically and whenever the xml file is reloaded the value is overwritten.
soundFolder	Path to alert sounds <b>folder</b> . Base folder + sound folder = complete path, e.g. if base folder is scripts/abcd, and your sounds are in scripts/abcd/mysound, then soundFolder field should be mysound. You can have different sound folders for different models, for example, you can have an Airbus sound folder for all airbus models and Boeing folder for Boeing models and so on.
retractableGear	When <i>false</i> will suppress alarms and notifications regarding landing gear due the fact this model does not retract it, when <i>true</i> will play corresponding alarms and notifications if flying with deployed landing gear.
reverse	When the model has reverse animation on the engine, it is associated with a car door (front right, front left, rear right, rear left, trunk, hood) or bomb bay. This field specifies which door the script should use, write <i>none</i> if it does not have a reverse animation.
reversePower	Vector representing the direction and intensity of the reverse power, it should be strong enough to make it fall from the sky if full reverse power is applied during flight. The main field is Y which consists of a negative value, but depending on model's centre of gravity and origin, it may pitch up or down when reversing on ground, therefore it is recommended to adjust Z value to inhibit the unwanted pitch. Pitching down/up when reversing airborne is expected.
weatherCoef	uses a fraction of reverse power when braking on landing or rejected take off under bad weathers (rain and snow), value 1 means 100% of reverse power, 1.5 is 150%. Negative values can be applied in order to simulate loss of braking efficiency due weather. Since reversePower may have strong intensity, it is advised using small values like 0.2. Negative values may apply force higher than fiction deceleration, resulting in increasing speed. Use the command "abort test" with auto mode to check how the braking is behaving on rejected takeoffs.
brakeCoef	Same as weatherCoef, but this one is applied on dry weathers.
powerBackCoef	Another coefficient using reversePower, this one is applied when you have reverse open and tries to move forward, resulting in a movement backwards.
computerControl	Specifies initial mode when entering the aircraft, can be <i>manual</i> , <i>semi</i> or <i>auto</i> .
groundPower	Specifies the power the auto mode will apply while on ground for correction during take off and such. Values x, y and z are float and range from 0 (no power) to 1 (max power). X, Y and Z represents Pitch, Roll and Yaw respectively.

airPower	Same as ground power but applied when aircraft is airborne. An airborne aircraft requires less power to rotate its axis, therefore airpower values tend to be lower than groundPower (due to friction and other physics involved).
transmitter	Transmitter is the 3D point used as reference regarding glide slope calculations, collision scans and other calculations. This field is an offset in relation to aircraft origin. It is recommended setting the transmitter to be under the wings. X moves to left/right of the aircraft, Y moves forward/backward and Z moves up/down. All values are float.
groundScan	Sets how many area spheres of wing span radius will be created to check for collision while on ground. More spheres, lower performance. The circles are created following the aircraft's movement direction. The value must be a positive integer.
airScan	Same as groundScan but used when airborne. The scan when airborne consists of groundScan+airScan, therefore an airScan of 20 and groundScan of 3 results a scan of 23 when airborne. If value is too high it may check unrendered area, which consists in waste of resource.
invertedDoor	For most of the models, closed door means closed reverse, although a model was found where it was the opposite (closed door meant open reverse and vice-versa). Use <i>true</i> for inverted logic, <i>false</i> for natural logic.
invertedLight	Same principle as inversedDoor but regarding lights and high beams.
sinkRateCoef	Float positive coefficient applied on Sink Rate calculation, higher coefficient means higher tolerance, in other words, the aircraft can descent faster without triggering any Sink Rate alarm. The formula to trigger Sink Rate is: <i>(bool)Sink Rate = V Speed (in metres) &lt; -20 * sinkRateCoef and Aircraft is Airborne.</i>
stallCoef	Float positive coefficient applied on Stall calculation; higher coefficient means higher tolerance. The formula to trigger Stall is: <i>(bool)Stall = Angle of Attack (AoA) &gt; 15° * stallCoef.</i>
v1Coef	Float positive coefficient applied to determine V1 callout, it is not a precise or reliable formula but is good enough. The formula to trigger V1 is: <i>(bool)V1 = (Aircraft Speed / Runway remaining length) * v1Coef &gt; 1 and State == Takeoff.</i>
takeoffHeightCoef	Float positive coefficient applied to set the height in which states will change from Takeoff to Flying or Going Around to Flying. The formula to trigger state change is: <i>(bool)Change State = Height above Surface (metres) &gt; 100*takeoffHeightCoef.</i>
landingHeightCoef	Similar to takeoffHeightCoef, this positive float coefficient is applied on state change from Landing to Flying or Landing to Going Around. The formula to trigger state change is: <i>(bool)Change State = Height above Surface (metres) &gt; 100*landingHeightCoef + 20.</i> There are more conditions to such state change, but it does not involve landingHeightCoef.
refuelSpeedCoef	Coefficient related to the speed in which refuel happens. Refuel Speed follow the formula: <i>Refuel Speed = 10*refuelSpeedCoef.</i>

refuelRatioCoef	Coefficient related to the amount of fuel inserted when <i>Refuel Tick Count</i> reaches Refuel Speed. <i>If (Refuel Tick Count++ &gt; Refuel Speed) then</i> <i>Fuel Level = Fuel Level + 0.01*refuelRatioCoef.</i> <i>Refuel Tick Count = 0.</i>
fuelConsumeCoef	This fields set the coefficient consumption of fuel, following the formula: <i>(float) Consumption = 0.005*fuelConsumeCoef*(Engines On + 2.5*Broken Engines On)</i>
minimumFuel	Percentage defining the fuel level to trigger fuel alarm.
throttleUp	Percentage of throttle up applied to match descent rate if aircraft is descending more than it should.
throttleDown	Percentage of throttle down applied to match descent rate if aircraft is descending less than it should.
pitch	Pitch to be kept when glide slope is between -0.3° and 2°
classification	Defines the size of aircraft between <i>small</i> , <i>medium</i> , <i>large</i> , <i>huge</i> and <i>not defined</i> , simply writing the initial letter is enough to identify. The omission of this field makes the script calculate classification according to model's width and length. The classification influences the runway it can use for Takeoff or Landing.
category	Defines the category of the aircraft, which can be passenger/pax, freight or military. Writing the initial is enough. The aircraft's category influences the runway it can use for Takeoff or Landing.
engineBone	Defines the aircraft bone that shows spinning in the engine (usually <i>misc_a</i> , <i>misc_b</i> , ...). This information is used to identify if specific engine is running or not.

Table 10 – Aircraft Configuration.

When creating a new model, all fields not present will inherit the value from the *default* model. If by accident a field from *default* model is missing, the value applied is a hard coded one.

It is also worth noting that *config.xml* fields as well fields from *display.xml* can be incorporated into models, fields not inserted will follow the value of their corresponding xml file.

```
<plane value="Airbus A319">
  <spawnName value="a319" />
  <hash>-1865365006</hash>
  <engineBone value="misc_a, misc_b" />
  <computerControl value="auto" />
  <airPower x="0.400000" y="0.400000" z="0.300000" />
  <display>
    <measure>
      <temperature value="fahrenheit" />
    </measure>
  </display>
</plane>
```

Figure 2 – Example of incorporation of display.xml fields.

This way, it is possible to set specific layouts according to the aircraft without disrupting general layout or the layout of another aircraft and have personalized engine rotation base values.

#### 6.4. airport.xml

This section provides explanation regarding airport parameters.

Item	Description
------	-------------

airport	Name of the airport.
position	The 3D location that represents the airport, it is applied to determine if the aircraft is close enough to get a gate when <i>Request Gate</i> command is issued.
load	Defines if the airport should be loaded or not, so instead of erasing unwanted airports the user can simply set <i>false</i> and the script will ignore such airport.
maintenance	Sets the 3D location of the airport to be used as hangar/maintenance area. Besides <i>x</i> , <i>y</i> and <i>z</i> fields, there's also the <i>radius</i> field that determines the area of detection.
icao	Airport code, must be unique among all other airports.
ilsRangeCoef	the range coefficient of the ILS signal, the range (in metres) is $1000 * ilsRangeCoef$ .
category	The category of the airport; can be Military or Civil. Military: Military and Freight aircrafts can use the airport Civil: Passenger/Pax, Freight and Military aircrafts can use the airport.

Table 11 – Airport parameters.

Each airport can have a set of runways, which can be set through fields below.

Item	Description
Runway	Short runway identification.
start	Sets the 3D location that marks the beginning of the runway.
end	Sets the 3D location that marks the end of the runway.
width	Sets the width of the runway, it is applied on verification of whether the runway is occupied or not.
slopeAngle	Defines the glide slope angle in relation to the runway.
touchDownCoef	Defines the distance (in metres) from the touchdown zone in relation to the start of the runways and sets from where the ILS signal is projected.
papiOffSet	Sets the position offset of PAPI lights.
classification	Sets up to which aircraft can use the runway; e.g. Medium classification means that small and medium aircrafts can use.
operation	Sets the attribution of the runway with one of the three values: Takeoff: Can be used for take off only. Landing: Can be used for landing only. Both: Can be used for take off and landing.

Table 12 – Runway fields.

The *airport.xml* has another set of fields regarding boarding gate configuration.

Item	Type	Description
alias	string	Name of the gate to be displayed at notification.
position	vector	Position of the gate on map.
approach	vector	Location prior to position. <b>Currently not in use.</b>
radius	float	Area of scan to determine if spot is in use by another aircraft.
classification	string	Defines up to which size of aircraft can use (small, medium...)

Table 13 – Gate properties.

## 6.5. keyboard.xml

This file is a massive key mapper, as script's key event only gave corresponding key code according to US keyboard standard, which is different of my keyboard standard, and no matter what it was done it never gave localized keys. Solution? Make things the hard way. The key map works exclusively when typing on FMS, which is explained on chapter 7.

## 7. Collision System

While flying, the aircraft can trigger the collision system in two occasions, collision against the ground or another aircraft. Only planes are set to be detected, so watch out for nasty helicopters.

Be it against the ground or not, the aircraft will simply pitch up and add thrust, once it no longer detects the threat of collision it starts to maintain a pitch of 5 degrees. Be careful, while pitching back to 5 degrees you may find yourself entering on another collision against the ground.

In case of a collision with an aircraft, a red blip in a shape of a plane will be set on the other aircraft. This blip will be removed when the distance between your aircraft and the other is long enough. While taxiing on ground, no action is taken, the alarm will be playing when another aircraft is inside the scan area and either you or the other aircraft is moving, if both are not moving then the alarm ceases.

## 8. Landing System

The new landing system available since Beta 1.4 works similar to *Hold* command. The parameters, however, are different. The pitch to be applied during landing comes from model's profile: the pitch field nested on *<autoLand>* node. It is highly recommended the use of values equal or greater than zero, also keep in mind that reaching or not the required pitch relies on airPower X value and aircraft's handling itself. Instead of spending long time searching the perfect power value, increase/decrease the pitch value to compensate the error margin. I.e: For some models it is set to keep 1.5° of pitch, because in reality it'll be steadily keeping 1° which is the desired value.

Throttle down field is the percentage of deacceleration applied in order to increase the descent rate, throttle up, however, is the percentage of thrust applied when there's a need to decrease the descent rate. For most models the throttle up value can be zero, as the lack of deacceleration itself already helps reducing the descent rate, but if model's handling has low lift power, thrusts may be required.

The Landing System only works on auto-mode and when the aircraft is somewhat aligned with glide slope ( $|\text{center line}| < 150\text{m}$  and  $|\text{parallel angle}| < 45^\circ$ ). If the aircraft is more than  $2^\circ$  above glide slope, a pitch down and deacceleration will be applied; if below  $-0.5^\circ$  of glide slope then Landing System won't take any action. When between those values, the aircraft will try keeping the pitch and reach an ideal descent rate towards the touchdown zone. The descent rate is calculated taking into consideration ground speed, distance to touchdown, aircraft's height and runway's height. If the aircraft enters pre-stall condition, the system comes to a halt, this also happens if there's no valid runway in range.

In addition to pitch and descent control, Landing System includes roll correction, which tries keeping roll at  $0^\circ$  and yaw correction that tries to correct parallel angle towards  $0^\circ$ , be aware that it does not correct centre alignment, that is still up to the player to correct.

## 9. Commands

In order to effectively reduce the amount of key binds, commands can be issued from *Flight Management System – FMS*. FMS allows the player to type command lines that execute functions.



### 9.1. Command argument type

Some commands can require arguments to be valid, in other words, values as an input that are essential to accomplish the task. For example, if you want the aircraft to climb to 3000ft, then the command *height* can be issued, but that alone doesn't imply the height itself, thus the command needs the intended height as argument, making the full command *height 3000*. The types of arguments are the following: *none*, *boolean*, *affirmative*, *negative*, *integer*, *float*, *point*, *vector*, *string*.

<i>Argument Type</i>	<i>Description</i>	<i>Example</i>
none	No argument is needed.	<i>cancel</i>
boolean	Binary value: <i>true</i> or <i>false</i>	ecam false
affirmative	Similar to boolean <i>True</i> but set to a more natural wording: on, start, true, 1.	engine on engine start
negative	Similar to boolean <i>False</i> but set to a more natural wording: off, end, false, 0.	engine off
integer	A number with no fraction: ..., -2, -1, 0, 1, 2, ...	height 3000
float	A fractioned number: 1.0032, -56.9, 2, -90, 90.0000001; The notation applied uses dot instead of comma.	trim 10.5
point	Two integer values, separated with a comma.	display.ecam = 100, 600
vector	Three float values, separated with commas.	model.airPower = 0.87, 1, 0.4
string	A sequence of characters.	model.displayname = 12k.Cjz

Table 14 – Argument type.

### 9.2. Regular commands

Regular commands are those which do not rely on external files. **Green** coloured commands apply to auto mode only, **orange** applies to semi and auto, **red** applies to all control mode. Commands of this section are not case sensitive.

<b>Command</b>	<b>Arguments</b>	<b>Description</b>	<b>Example</b>
<b>height</b>	integer	Sets a height to automatically climb/descent. The command follows player's height unit and works with absolute height, so if aircraft is at 2000ft and player desires to climb up to 3000ft, the player must input 3000 instead of the difference between current height and desire one.	height 2500
<b>turn</b>	integer	Aircraft turns to the specified angle (in degrees), positive angle turns clock wise, negative counter clock-wise. If angle is higher than 180 or bellow -180, the angle is simplified. E.g. 190 $\equiv$ -170	turn 90

<b>pitch/trim</b>	float	Sets a pitch angle to be maintained. It is disabled when aircraft is on hazardous situation like stall, sink rate and such, by switching to manual mode or by <i>cancel</i> command. There is an error margin on the kept angle.	trim 10 pitch 10  P.S.: both are equivalent.
<b>roll</b>	float	Sets a roll angle to be maintained. It is disabled when aircraft is on hazardous situation like stall, sink rate and such, by switching to manual mode or by <i>cancel</i> command. There is an error margin on the kept angle.	roll 25
<b>yaw</b>	float	Different to roll and trim, yaw command sets a constant power on rudder to be kept, expressed as percentage between -100 and 100. It is disabled when aircraft is on hazardous situation like stall, sink rate and such, by switching to manual mode or by <i>cancel</i> command.	yaw 30
<b>throttle</b>	float	Like yaw, sets power to be constantly applied, but in this case as throttle (up from 0 to 100, or down from 0 to -100).	throttle 75 throttle -100
<b>hold</b>	float, float	Sets a climb/descent rate and a pitch respectively. If rate is positive and pitch is negative, the command is refused as one cannot climb while pitching down. Hold command has priority towards previous commands (trim, yaw etc...)	hold -300, 1.5 hold -1000, -2 hold 300.7, 5
<b>cancel</b>	none	Interrupts trim, yaw, roll, throttle, climb/descent and turning commands.	cancel
<b>stab</b>	none	Interrupts pitch/trim, roll, yaw and throttle commands.	stab
<b>takeoff</b>	string	Manually sets a runway to take off, ignoring classification, operation and category. Argument consists of airport ICAO and runway name separated by a dot, or just runway name if it's unique.	takeoff Isia.12R takeoff 12R
<b>land</b>	string	Same as take off command, but to land.	land Isia.12R land 12R
<b>refuel</b>	none	Inverts current refuel status: if it's refuelling then it stops, if it	refuel refuel on

		was not refuelling then it tries to refuel up to 100%.	refuel start refuel end
	affirmative	Tries to refuel the aircraft up to 100% if conditions are met.	
	negative	Stops refuelling the aircraft.	
<b>Refuel up to</b>	float	Refuels the aircraft up to inserted value, which must be between 0 and 100. If value is lower than current fuel level, than fuel will be drained.	refuel up to 75.5
<b>dump</b>	none	Inverts current fuel dumping status, if it's dumping then it ceases, otherwise tries to dump down to minimum fuel level.	dump dump start dump on dump end
	affirmative	Tries to dump fuel down to minimum fuel level.	
	negative	Ceases to dump fuel.	
<b>dump down to</b>	float	Tries to dump fuel down to informed valued or minimum fuel level, whichever reaches first.	dump down to 27.09
<b>airport</b>	none	Notifies all registered airports and the number of valid runways.	airport
<b>icao</b>	string	Value is the ICAO of an airport, in return it is notified of all valid runways, their headings and length.	icao lsia
<b>toga</b>	none	Has same effect as pressing the TOGA key bind, changing aircraft's state according to circumstances.	toga
<b>mode</b>	string	Changes aircraft mode to manual, semi or auto.	mode auto mode semi
<b>engine</b>	none	Tries to invert current engine status. Those which are off are set on, those which are on are set off. Auto mode may block this action.	engine engine start engine off engine 3 engine 2 start engine 2 on engine 4 off engine 4 false
	affirmative	Tries to set all engines on.	
	negative	Tries to shut down all engines. Auto mode may block this action.	
	Integer	Tries to invert the status of corresponding engine. Auto mode may block this action.	
	Integer, affirmative	Tries to turn on the corresponding engine.	
	integer, negative	Tries to turn off corresponding engine. Auto mode may block this action.	
<b>map</b>	none	Invert navigation map style. If map is small sets to big, and vice-versa.	map map on map off

	affirmative	Sets large navigation map.	
	negative	Sets small navigation map.	
<b>radar</b>	none	Invert map visibility. If map is visible then it hides, otherwise displays it.	Radar radar on radar off
	affirmative	Sets map to visible.	
	negative	Sets map to invisible.	
<b>wings distance</b>	none	Notifies the distance between wingtips	wing distance
<b>transmitter</b>	none	Notify world position of transmitter.	transmitter
<b>transmitter offset</b>	none	Notify offset position of transmitter.	transmitter offset
<b>night vision</b>	none	Toggles night vision on or off. Can use <i>night</i> for short.	nightvision night
<b>play</b>	string	Argument can be extra 1, extra 2 or extra 3 sound files. This command plays selected file once.	play extra 1
<b>loop</b>	string	Plays argument sound in a loop if file is available, argument is any of the Loop Code from sound table. Loops one sound at a time. To break the loop, use argument "off".	loop height loop stall loop off
<b>set extra</b>	int, string	Sets a new sound file to extra1, extra2 or extra3, depending on the int value. The string contains the path to the sound file from base folder.	Set extra 1 abcd/file.mp3
<b>ils</b>	none	Inverts ILS status. The deactivation of ILS disables Helper display, Approach Reference Line, PAPI Lights and sounds associated with those.	Ils ils on ils off
	affirmative	Activates ILS. The activation of ILS affects Helper display, Approach Reference Line, PAPI Lights and sounds associated with those. Helper display may remain off if command for such was issued.	
	negative	Disable ILS instruments. The deactivation of ILS disables Helper display, Approach Reference Line, PAPI Lights and sounds associated with those.	
<b>particle</b>	none	Invert/Enable/Disable particles feature.	particle particle on particle off
	affirmative		
	negative		
<b>alert</b>	none	Invert/Enable/Disable alert box feature.	alert alert on
	affirmative		

	negative		alert off
<b>ecam</b>	none	Invert/Enable/Disable ECAM feature.	ecam
	affirmative		ecam on
	negative		ecam off
<b>helper</b>	none	Invert/Enable/Disable approach helper feature.	helper
	affirmative		helper on
	negative		helper off
<b>weather</b>	none	Invert/Enable/Disable weather report feature.	weather
	affirmative		weather on
	negative		weather off
<b>system</b>	none	Invert/Enable/Disable system display feature.	system
	affirmative		system on
	negative		system off
<b>ext system</b>	none	Invert/Enable/Disable extended system display feature.	ext system
	affirmative		ext system on
	negative		ext system off
<b>sound</b>	none	Invert/Enable/Disable sound feature.	sound
	affirmative		sound on
	negative		sound off
<b>reference</b>	none	Invert/Enable/Disable reference line feature.	reference
	affirmative		reference on
	negative		reference off
<b>fade</b>	none	Invert/Enable/Disable fade state of all text fields.	fade
	affirmative		fade on
	negative		fade end
<b>gauge</b>	none	Invert/Enable/Disable display of gauge.	gauge
	affirmative		gauge on
	negative		gauge off
<b>deice</b>	none	Invert/Enable/Disable ice effect feature. <b>This feature is currently unavailable.</b>	deice
	affirmative		deice on
	negative		deice off
<b>abort test</b>	none	Invert/Enable/Disable take off abort test feature.	abort test
	affirmative		abort test on
	negative		abort test end
<b>debug</b>	none	Invert/Enable/Disable debug mode.	debug
	affirmative		debug start
	negative		debug off
<b>company</b>	string	Changes aircraft's company.	company easyJet
<b>request gate</b>	none	Requests a new boarding gate.	request gate
<b>repair</b>	none	Repairs the aircraft to complete health.	repair
<b>keep last</b>	none	Invert/Enable/Disable FMS auto fill with last command when popping up.	keep last
	affirmative		keep last on
	negative		keep last off
<b>lock</b>	none	Invert/Enable/Disable aircraft movements when typing on FMS.	lock
	affirmative		lock on
	negative		lock off

Table 15 – Command list.

It is important to know that the commands from Table 14 do not change any parameter from XML files. To change those via FMS there is another set of commands explained on next section.

Additionally, several commands can be issued at once, separating one from another with semicolon (“;”): *engine off; refuel; play extra 2*

## 10. Setting XML Parameters

To change parameters from XML files, the user can manually change the file or use FMS to change some of them. After modifying the files, the user can issue the *reload* command to make the script read once again all XML files.

To manually create an aircraft model, airport/runway/gate and such, it is highly recommended to copy an existing one and change the parameters, this way the probability of forgetting to close a node tag or misspell the tag name is reduced. Remember, XML fields are case sensitive.

XML files can be edited via FMS, with exception to *airport.xml*, *keyboard.xml* and *memory.xml*. Which once the setup is done you rarely will change it.

Each XML file has a prefix to use:

- model.xml: plane.
- display.xml: display.
- config.xml: config.

After typing the prefix, the next step is writing the field. When using FMS, you don't need to worry about upper or lower letters of the fields, neither the sub nodes. For example, if you want to change the *brakeCoef* of current aircraft:

- ~~plane.engine.brakecoef~~
- plane.brakecoef

Once you have typed prefix and field, you must insert equals symbol = before inserting the value. The value format varies according the field. For example, the field *plane.reversepower* is a x, y, z vector, so the way you insert the data is different from *plane.spawnname*, which is a string.

- String: = text
- Int: = number
- Float: = number
- Bool = value
- Point (int fields x and y): = number, number
- Vector (float fields x, y and z): = number, number, number

Examples:

- plane.spawnname = b748f
- display.ecam = 100, 350
- plane.reversepower = 0, -10.45, 2
- plane.stallCoef = 3.1
- plane.groundscan = 3
- config.ils = false

As for now, the only XML field that doesn't follow the field name is plane display name, which resides on `<plane value="name">`. To change that field the command is *plane.display = Boeing 747-800 Freighter*.

Regarding aircrafts, an additional command can be used:

*plane.create(<spawn\_name>)*

This command creates a new node for a model if it doesn't exist. This new plane model creates only the bare minimum fields required, therefore the fields not set inherit values from *default* model.

Since display.xml and config.xml fields can be incorporated to model.xml, to change those value you must insert the corresponding prefix:

- ~~plane.ecam = 250, 100~~
- plane.display.ecam = 250, 100

If the command is correctly issued, the data will be saved and the script will automatically reload all XML files.

## 11. Memory and Action

To ease the amount of typing, *memory.xml* file enables the user to define commands and chain commands and invoke them via FMS, memories can be inserted on memory.xml under <registry> and <memories> node.

```
1  <?xml version="1.0"?>
2  <registry>
3    <memories>
4      <memory alias="visual" value="reference line off; helper off" />
5      <memory alias="instrument" value="reference line on; helper on" />
6      <memory alias="init" value="lock on"/>
7      <memory alias="seteasy" value="plane.company = EasyJet"/>
8      <memory alias="easy" value="companyEasyJet"/>
9      <memory alias="company" value="company@" />
10     <memory alias="keeper" value="keep last @" />
11     <memory alias="extra" value="setextra@" />
12     <memory alias="pa" value="play extra 1" />
13     <memory alias="seatbelt" value="play extra 2" />
14     <memory alias="gate" value="request gate"/>
15     <memory alias="rest" value="engine off; refuel; /seatbelt"/>
16     <memory alias="wakeup" value="engine on; refuel off"/>
17     <memory alias="route" value="/t#@; /l#@"/>
18     <memory alias="t" value="take off @" />
19     <memory alias="l" value="land @" />
20     <memory alias="m1" value="/m2"/>
21     <memory alias="m2" value="/m3"/>
22     <memory alias="m3" value="/m1"/>
23   </memories>
24 > <behaviour> ...
102 </behaviour>
103 </registry>
```

Figure 3 – Memory Example.

A memory has two fields, *alias* and *value*. *Alias* is the name you'll use to invoke on FMS, keep in mind that to invoke a memory, you must use "/" in front of the alias when inserting the command. *Value* is the command to be issued, this command could be any valid command from section 7 or 8 or even another memory.

If the command requires a parameter, e.g. *refuel up to <number>*, such argument number must be passed using "#" followed by desired value, and the memory value must have "@" symbol identify where to insert it. Therefore, if you desire a more flexible memory to refuel the aircraft up to a value, the memory will look like this:

```
<memory alias="mem1" value="refuel up to @" />
```

When invoking, you'll type:

```
/mem1 #95
```

In this situation, if the aircraft is on taxiing state and not moving, it will refuel up to 95%. If you want a more static memory to, let's say, always refuel up to 80%, you could create a memory like:

```
<memory alias="mem2" value="refuel up to 80" />
```

Which takes no argument when invoking `/mem2`.

Looking at line 17 on Figure 3, we can find a more elaborated example, the *route* memory. As it has two "@" symbols, it means it requires two arguments.

```
<memory alias="route" value="/t#@; /l#@"/>
```

Let's say we invoke such memory as the following:

```
/route #12R #21
```

The script will replace all "@" in the respective order and automatically process the following command:

```
/t#12R; /l#21
```

As those are chained commands, it will break into two pieces (in this case) and process according to the respective order: `/t#12R` and `/l#21`. While processing `/t#12R`, it will recursively call the *t* memory using `12R` as argument, which leads to the following command:

```
/t#12R => take off @ => take off 12R
```

As *take off* is a final command (not a memory), the process will regress the recursions solving all memories invokes on the way back to root command. So far, the command was translated as follow:

```
/route #12R #21 => take off 12R; /l#21
```

First part is solved, now the process will apply same method for `/l#21`, which will lead to the final command:

```
take off 12R; land 21
```

As memories can invoke other memories, including themselves, loops can be created, leading to infinite process as displayed on line 20 with `/m1` memory. Therefore, each time a non-final command memory is called on translation, it is registered on a list of invoked memories. Once a memory tries to invoke another one that is already on the list, the entire chain of command will be disposed and a notification alerting the loop will be displayed.

### 10.1. Actions

As memories were already a powerful feature, *Action* is the cherry on top. During the transition from one state to another or upon reaching a state (let's call it *Event*), actions can be triggered, in other words, an automated command issuer. For example, you can turn engines off when reaching the boarding gate, set files to extra1, 2 or 3 and play them according to flight stage and company name, and many other useful applications. This action fields can be found at `<behaviour>` node at *memory.xml*, and are inside already established events. The *Events* are as follow:

- **afterload:** execute actions after *XML* files are read, in other words, when aircraft model is switched. When the command *reload* is issued, it does not trigger this event.
- **reachedgate:** execute actions when aircraft completely stops inside the area of designated gate.
- **leftgate:** execute actions when aircraft leaves the area of designated gate.
- **taxi/takeoff/landing/flying/goaround/takeoffaborted:** execute actions when aircraft enters the respective state.



- **<previous state>To<current state>**: execute actions when a specific transition of states occurs. Keep in mind that the commands are executed at <current state>, right before <current state> actions. For example:

```

<flying>
  <action value="map off">
</flying>
<takeoffToFlying>
  <action value="map on">
</takeoffToFlying>

```

After successful take off, the chain of events will be *map on* followed of *map off*. Transitions considered “impossible” such as *taxi -> goaround* are not executed, in fact, all *feasible* transitions are already included on the XML file, all you have to do is to include or remove <action /> tags to your liking.

Another interesting feature is that actions can be defined to be executed by specific airline company.

```

24 <behaviour>
25 >   <afterload> ...
27   </afterload>
28 >   <taxi> ...
30   </taxi>
31 >   <takeoff> ...
33   </takeoff>
34 >   <flying> ...
36   </flying>
37 >   <landing> ...
41   </landing>
42 >   <goaround> ...
44   </goaround>
45 >   <takeoffaborted> ...
47   </takeoffaborted>
48 >   <taxiToTakeoff> ...
50   </taxiToTakeoff>
51   <takeoffToTaxi>
52   </takeoffToTaxi>
53   <takeoffToFlying>
54     <action value="/seatbelt" />
55     <action value="/extra#3#Sounds/Easyjet/after_takeoff.mp3; play extra3" company="EasyJet"/>
56     <action value="/extra#3#Sounds/NoCompany/after_takeoff.mp3; play extra3" company="none"/>
57   </takeoffToFlying>
58   <takeoffToTakeoffaborted>
59   </takeoffToTakeoffaborted>
60 >   <landingToTaxi> ...
64   </landingToTaxi>
65   <landingToFlying>
66   </landingToFlying>

```

Figure 4 – Events and Actions.

Not defining a company field sets such action to be executed by all companies. Figure 4 displays three actions at line 54, 55 and 56 respectively. The first one is executed by all companies, the second one only by EasyJet and the last one only those who have no associated company.

Caution must be taken regarding loop between Events, as there's no verification for that. An example of such loop is:

```

<taxi>
  <action value="toga" />
</taxi>
<taxiToTakeoff>
  <action value="toga" />
</taxiToTakeoff>

```

In the above situation, an *Event* loop is created between taxi and taxiToTakeoff as one will be switching to the other and vice-versa.

## 11. Tips & Hints & Headaches Avoidance

- Do not use space or special characters for folder or files (“-”, “\_” are ok, I’ll let you have them 😊).
- Try using memories instead of direct command, they are more dynamic and you can set an alias that is easier for you to remember and faster to type.
- The amount of information might be overwhelming, so take it slow, play around with parameters, leave a backup of original *xml* files somewhere safe in case it gets screwed up (yes, it happened during development... more than once...).
- Before adjusting a model’s parameters, make sure the *handling.meta* is properly set to your taste. This script is **not** intended to compensate or correct handling poorly set.
- You can deactivate display windows
- When adjusting position of display windows, use the command “keep last”, so you won’t need to type all of the command over again.
- “While I was playing some weird notifications appeared”. Leave a comment describing what such notification displayed, and what was the aircraft doing that showed that. It might be a “debug notification” left behind.
- Unbalanced engine effect changes from model to model, being centre of mass one of the key parameters that defines effect’s behaviour. Keep in mind this feature is still on development.
- Depending on the handling or airpower setup, Height and Turn commands may have a difficult time stabilizing to cease the operation. Until a better way to conclude the operation is applied, use this feature with caution.
- If you open the reverse while flying, when you close it there might be a tiny gap to close completely due to game mechanism. Most of the time if you simply decelerate may solve this inconvenience.
- In order to set a great variety of PA announcements through extra1, 2 and 3 files, look at the examples on *memory.xml* behaviour’s node.

```
<action value="/extra#3#Sounds/Easyjet/boarding.mp3; play extra3" company="EasyJet"/>
```

- Start playing with Semi-Auto control, once you are comfortable with the aircraft behaviour, then proceed to try the Auto mode. Going straight to Auto mode without being used to commands and control may make you fight against the controls the script impose while trying to keep the flight envelope.
- This is a work in progress, do not fear giving your honest feedback.
- This script was tested on a low FPS machine configuration, for those who run at much higher FPS (basically everybody) there -might- be a drop of frame rate.
- While developing, I kept trying to optimize every bit of code (Beta v1 came after Alpha v20), after all it is more than 13.500 lines of code. So be clear when reporting a bug, I highly recommend pasting the error displayed at ScriptHookVDotNet.log
- If the FPS drop is unbearable, leave a comment at the download page.
- Suggestions for new features or fixes are always welcome.
- This is a one-man script, be patient for fixes and further releases.
- No blips set by this script should be left behind when deactivating or aborting the script. If a blip is still showing, leave a comment at the download page.