

# **THRUSTMASTER®**

## USER MANUAL





# CONTENTS

|                                                                                 |           |
|---------------------------------------------------------------------------------|-----------|
| <b>1. Introduction .....</b>                                                    | <b>2</b>  |
| <b>2. System requirements.....</b>                                              | <b>2</b>  |
| <b>3. “Fly NOW !” quick launch.....</b>                                         | <b>3</b>  |
| <b>4. The Dashboard .....</b>                                                   | <b>5</b>  |
| Print configuration .....                                                       | 5         |
| The controllers .....                                                           | 8         |
| The buttons .....                                                               | 8         |
| The axes .....                                                                  | 8         |
| The keyboard .....                                                              | 9         |
| <b>5. Operating principles .....</b>                                            | <b>10</b> |
| <b>6. Creating a Basic configuration .....</b>                                  | <b>11</b> |
| Select Controller(s) to configure.....                                          | 11        |
| Configure axes mapping .....                                                    | 12        |
| Configure axes response .....                                                   | 14        |
| <b>7. Graphical User Interface.....</b>                                         | <b>20</b> |
| The “Select Control” window .....                                               | 20        |
| Selecting a button .....                                                        | 21        |
| The “Assign Parameters” window .....                                            | 22        |
| Assigning several functions to the same button .....                            | 24        |
| Creating a chain.....                                                           | 25        |
| Creating a sequence.....                                                        | 25        |
| LED Management .....                                                            | 25        |
| <b>8. Programming an axis.....</b>                                              | <b>28</b> |
| Axmap1: generates events that depend on the axis movement direction .....       | 28        |
| Axmap2: generates an event when the axis reaches a predefined value range ..... | 29        |
| The “Select Event” window .....                                                 | 30        |
| Launching the configuration .....                                               | 31        |
| <b>9. “Basic” exercise.....</b>                                                 | <b>34</b> |
| <b>10. Creating an Advanced configuration .....</b>                             | <b>36</b> |
| Layers .....                                                                    | 36        |
| Press and Release.....                                                          | 39        |
| <b>11. A few words about the T.A.R.G.E.T text programming language.....</b>     | <b>40</b> |
| Consumer warranty information .....                                             | 42        |
| Additional warranty provisions .....                                            | 42        |
| Liability .....                                                                 | 42        |



## Foreword

First of all, welcome to this manual and congratulations on your purchase! We hope that this game controller will provide you with many hours of enjoyment and will meet your every need. To help you get the most out of your controller, we would ask you to read this manual and keep it within easy reach the first few times you use T.A.R.G.E.T. This document has been written to answer questions from all types of users, whether experienced or complete beginners.

## 1. Introduction

---

T.A.R.G.E.T (Thrustmaster Advanced pRogramming Graphical EdiTor) is a software suite which lets you test, configure and program Thrustmaster controllers. All of the features can be accessed from the software's main page. This page is called the "Dashboard".

All Thrustmaster controllers are Plug and Play; installing and using this application is therefore optional. However, if you want to use profiles, change the control sensitivity or just get the maximum out of your purchase, T.A.R.G.E.T. will be your loyal ally.

Thrustmaster products such as the HOTAS COUGAR are famous for their exceptional programming. The downside of this power was the use of a programming language in a non-user friendly text format. In order to make this power accessible to as many people as possible, Thrustmaster has developed an application which can satisfy both beginner and expert users: T.A.R.G.E.T.

By default, a DirectX controller only generates information related to its nature. With T.A.R.G.E.T you can add keyboard and mouse functions. This will allow you to be more effective, to use buttons and axes as you like and use the keyboard as little as possible, as this hinders you from becoming fully immersed in the game. In certain cases, you can even correct software ergonomics or functionality faults.

These items of equipment are often called programmable controllers; however, you must not be intimidated by these terms as in practice, "programming" a joystick simply means associating an input (a button or axis) with an output (a function, keyboard key emulation, etc.).

With T.A.R.G.E.T, you can create complex programs using only your mouse, according to the Drag and Drop principle.

There are several levels of programming complexity to allow you to start using the application:

- **Basic** will be ideal for starting out, or for creating a simple profile quickly.
- **Advanced** lets you use the advanced functions offered by a top-of-the-range customization application.
- **Script** lets you use the full power and flexibility of T.A.R.G.E.T's script-based system.

T.A.R.G.E.T perfectly summarizes Thrustmaster's experience in programmable peripherals. In addition to creating a full graphic interface, the traditional Script language has been fully rewritten to provide more flexibility, new functions and almost unlimited possibilities. You must have a good level of knowledge of input peripherals to get the most out of the application's capabilities.

## 2. System requirements

---

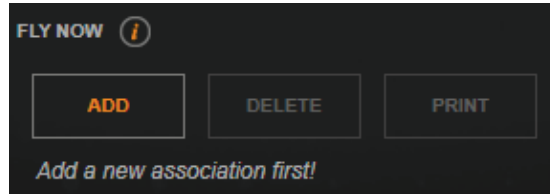
### PC

- Desktop/laptop PC with Intel Pentium III/Athlon 1GHz processor or higher
- 512MB RAM
- Operating system: Microsoft Windows® Vista®/7/8/10 (32 & 64-bit)
- 16-bit, 1280 x 960 video resolution
- 100MB free hard disk space



### 3. “Fly NOW !” quick launch

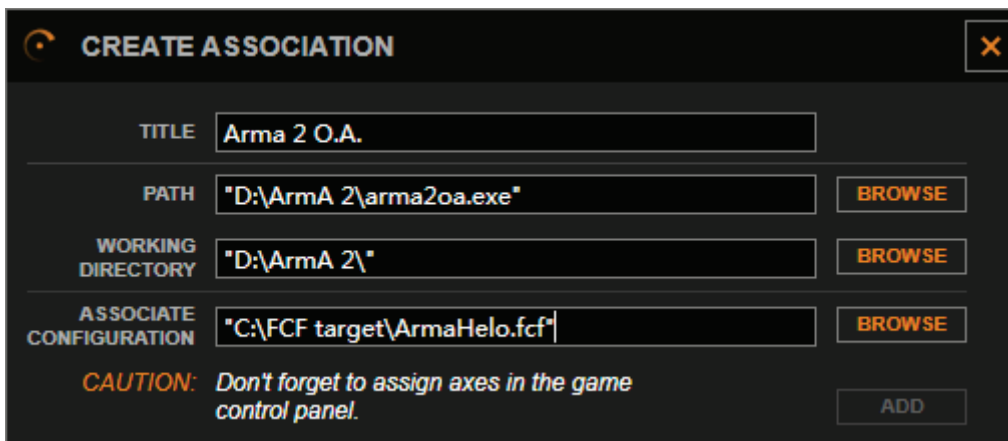
Double-click the T.A.R.G.E.T GUI (Graphical User Interface) icon on your desktop, or launch the application via the Windows **Start** menu. “Fly NOW!” is a little tool to launch a configuration and the associated gaming software quickly.



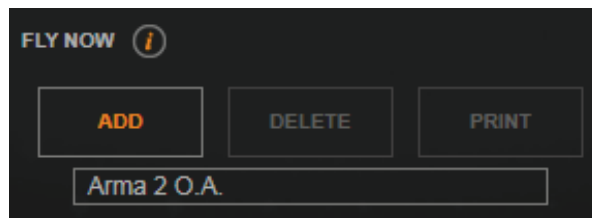
To use it, the first thing to do is to teach "Fly NOW!" which simulator you want to use: simply click the **Add** button. A new window opens.

- In the **Title** Field, name the simulator.
- In the **Path** field, click the **Browse** button and indicate the path to the application's executable file.

**Warning:** certain programs require specific commands to be launched; this is the case for the DCS series. In this case, the best thing to do is to right-click the shortcut to the simulator on your desktop. Select **Properties** and copy the contents from the target field. Then all you need to do is copy and paste this to the **Path** field in the T.A.R.G.E.T window.

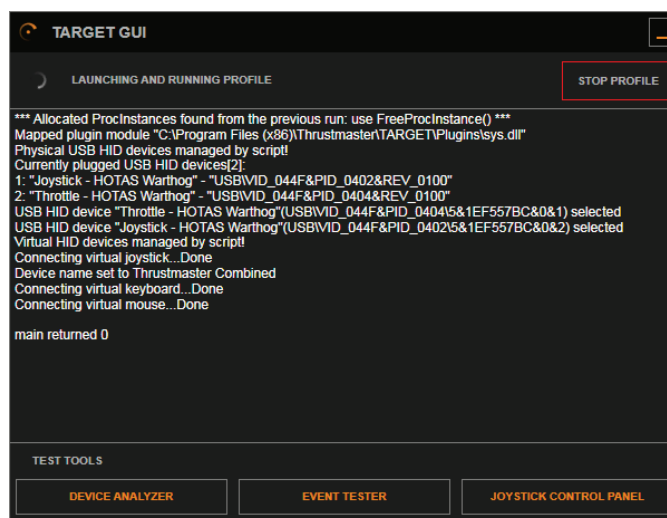


- In the **Associate Configuration** field, click the **Browse** button and select the profile that corresponds to the selected simulator. The T.A.R.G.E.T profiles are .fcf or .tmc type files.
- Finally, click the **Add** button. This is how you add new simulators to the “Fly NOW!” quick launch.





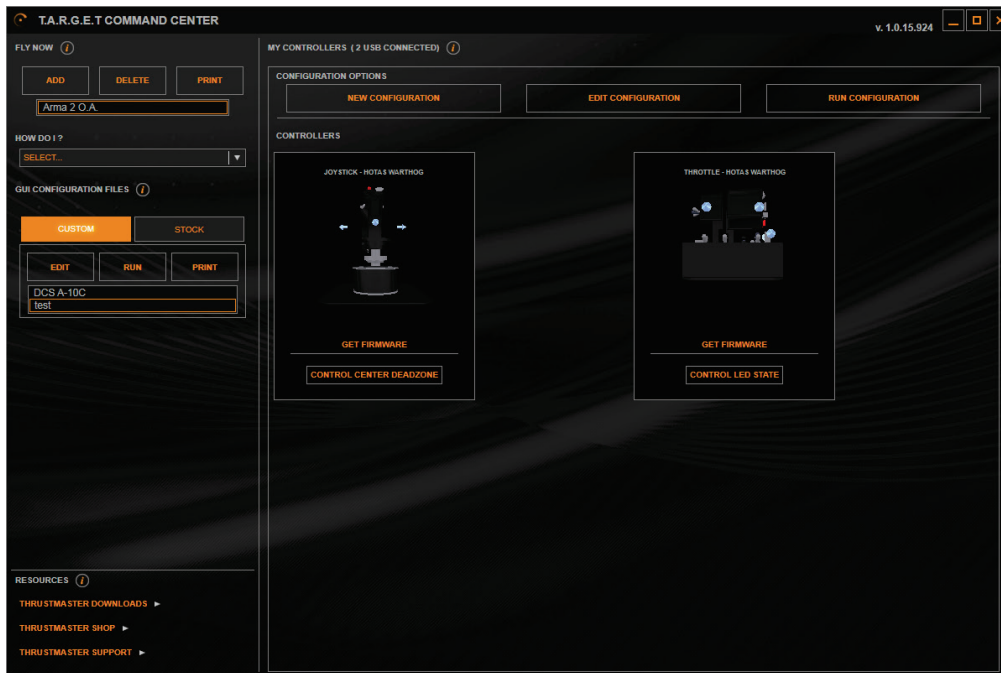
To launch the simulator and its associated configuration automatically, simply click on the green arrow that appears when you place the cursor over the gaming software name. The following window opens: the configuration is being launched. Once this procedure is complete, the simulator will be started.



The first thing to do once you have accessed the simulator is to configure the axes in the simulator's **Options** menu. You will only need to do this once. When you exit the simulator, you will need to shut down the T.A.R.G.E.T configuration by clicking the **Stop** button in the top right of the **T.A.R.G.E.T Wizard Running Script** window.



## 4. The Dashboard



The Dashboard is the main T.A.R.G.E.T. page.

This page is divided into several zones. The left-hand column is intended for quick launch:

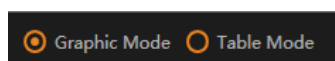
- **Fly NOW!** lets you quickly launch a simulator and its associated configuration.
- **How Do I?** lets you display help.
- **My Configuration Files** lets you launch, edit or print a configuration quickly. Double-click on a configuration to launch the editor. Right-click to open an action menu. These are stored under 2 tabs: **Custom** for personal creations; and **Stock** for the files provided with T.A.R.G.E.T.
- **Resources** contains links to Web support.

The right-hand section is dedicated to editing and managing Thrustmaster USB peripherals.

- The **New Configuration** button lets you create a new configuration.
- The **Load Configuration** button lets you edit a configuration.
- The **Run Configuration** button lets you execute a configuration and thereby activate programming for your controllers.
- The lower windows correspond to the controllers that you currently have connected. Certain windows may contain a button if the peripheral suggests operating options.

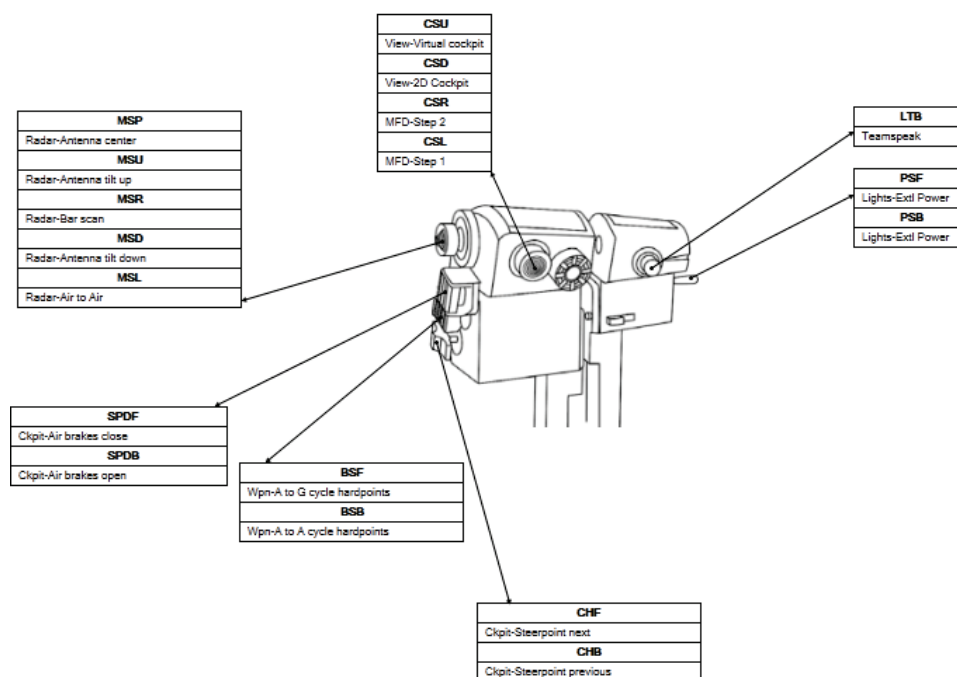
### Print configuration

The **Print** button in the **GUI Configuration Files** area generates an image that displays the functions mapped on buttons. The way in which button functions are displayed is generated automatically: that is why sometimes the text box positions don't seem to be optimized. The **Print** function can generate 2 kinds of images. You can switch between them by selecting the type in the lower left-hand corner:





**Graphic mode** is easy to read but limited for very complex files. If the configuration file is complex, some programmed functions may not be displayed. For CHAIN or SEQUENCE (see later), only the first function is displayed.



**Table Mode** is more basic (you have to know the names of your controller buttons) but displays a wider range of data (highly recommended for advanced configuration files).

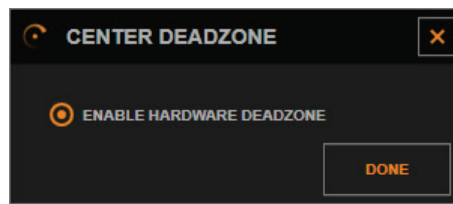
| Thrustmaster Hotas Warthog Throttle |       |       |                                                   | FalconAF |
|-------------------------------------|-------|-------|---------------------------------------------------|----------|
| Control Name                        | Shift | Layer | Press                                             | Release  |
| MSP                                 |       |       | Radar-Antenna center                              |          |
| MSU                                 |       |       | Radar-Antenna tilt up                             |          |
| MSR                                 |       |       | Radar-Bar scan                                    |          |
| MSD                                 |       |       | Radar-Antenna tilt down                           |          |
| MSL                                 |       |       | Radar-Air to Air                                  |          |
| SPDF                                |       |       | Ckpit-Air brakes close                            |          |
| SPDB                                |       |       | Ckpit-Air brakes open                             |          |
| BSF                                 |       |       | Wpn-A to G cycle hardpoints                       |          |
| BSB                                 |       |       | Wpn-A to A cycle hardpoints                       |          |
| CHF                                 |       |       | Ckpit-Steerpoint next                             |          |
| CHB                                 |       |       | Ckpit-Steerpoint previous                         |          |
| PSF                                 |       |       | Lights-Extl Power<br>Lights-Extl Anti Coll        |          |
| PSB                                 |       |       | Lights-Extl Power<br>Lights-Extl Anti Coll        |          |
| LTB                                 |       |       | Teamspeak                                         |          |
| EFLNORM                             |       |       | Ckpit-Fuel Master Switch<br>Ckpit-Fuel Switch Inc |          |
| EFRNORM                             |       |       | Ckpit-EPU step                                    |          |
| APUON                               |       |       | Ckpit-Landing gear Down                           |          |
| LDGH                                |       |       | Ckpit-Betty Master Caution                        |          |
| FLAPU                               |       |       | Flaps-Set to Full                                 |          |
| FLAPD                               |       |       | Flaps-Set to Full                                 |          |
| EACON                               |       |       | Ckpit-Power Main Inc                              |          |
| RDRNRM                              |       |       | Ckpit-Radar Altimeter ON                          |          |
| EORIGN                              |       |       | Ckpit-JFS switch                                  |          |
| EFLOVER                             |       |       | Ckpit-Fuel Pump Switch Dec                        |          |
| APUOFF                              |       |       | Ckpit-Landing gear Up                             |          |
| EACOFF                              |       |       | Ckpit-Power Main Dec                              |          |
| RDRDIS                              |       |       | Ckpit-Radar Altimeter STDBY                       |          |
| APDIS                               |       |       | Ckpit-Autopilot                                   |          |
| IDLERON                             |       |       | Ctrl-Throttle Idle Detent                         |          |
| CSU                                 |       |       | View-Virtual cockpit                              |          |
| CSD                                 |       |       | View-2D Cockpit                                   |          |
| CSR                                 |       |       | MFD-Step 2                                        |          |
| CSL                                 |       |       | MFD-Step 1                                        |          |



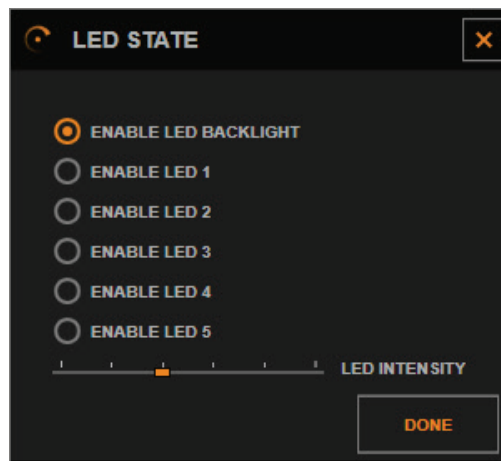
The "Controllers" area simply displays the Thrustmaster T.A.R.G.E.T compatible controllers connected to your computer. Some of these devices include some options.  
HOTAS Warthog **Get Firmware**: displays the version number of the controller's firmware. Available for the Joystick and the Throttle.



HOTAS Warthog Joystick **Control center deadzone**: displays a window used to enable and disable the center deadzone on the joystick. You will probably not notice the deadzone with a "stock" joystick, but if you use a "grip extension" mod, we recommend that you deselect this box.



**Control LED State**: opens a lighting management window where you can turn LEDs "ON" or "OFF" and adjust backlighting intensity. T.A.R.G.E.T supports some devices that were developed a long time before the software was created. The MFD lighting support in the T.A.R.G.E.T software is a bonus. If you are experiencing issues like MFDs not responding, simply unplug and then reconnect the MFDs. This can happen if the USB port was in "sleep" mode while the MFDs were not being used. You will also notice that if you move the Backlight intensity slider too quickly on a large number of values, the MFD will freeze for a few seconds (saturated by data), before reaching the right intensity value.







## The controllers

What is a game controller?

Today, there are many different types of game controllers, such as joysticks, throttles, steering wheels, rudders and so on. While they may look different from one another, they all measure and transmit the same types of information to the computer. In fact, we only distinguish between 2 types of information generated by a game controller: the all or nothing state information (ON or OFF) for the buttons, and the position information within a value range (for example, 54 in a range between 0 and 256) for axes, such as turning the steering wheel or the position of the throttle.

To make the best use of your controllers, it is important to understand the information they send. Programming a controller simply means linking events to the status and status changes for its buttons and axes.

## The buttons

The buttons may have a range of appearances: a joystick trigger, a push button, and also more complex assemblies that feature several buttons within the same mechanism, such as a Hat, a tilt button or lever (toggle switch with several positions).

A temporary push button such as the OSB of an MFD or the S2 (Weapon Release) of a Warthog delivers two types of state information:

- ON
- OFF

We also consider that it gives us two other items of information; namely, its state changes:

- OFF to ON
- ON to OFF

Therefore, this gives us four pieces of information that we can use on a simple push button. It is up to us to choose the information or several pieces of information which correspond to what we want to do.

There are different types of buttons: although the information delivered is identical, their mechanisms may vary. We therefore need to know how the button behaves. In the previous example, we have chosen an MFD's OSB button. This is a very simple component: it changes to **ON** when you press it, and returns to **OFF** when you release it.

A toggle switch, such as the HOTAS Warthog's APU START, behaves differently. When you activate it, it goes from **OFF** to **ON**. When you release it, it remains in the ON position; only intervention by the user will allow it to return to **OFF**. We therefore have the same amount of information that can be used, but the button behaves differently. We will need to take this into account or use it when allocating functions.

If you fail to take this into account, you may end up with what is called a "sticky key": a keyboard key that is permanently pushed down (in a virtual sense). This may saturate the keyboard buffer after a few seconds, and cause unwanted reactions.

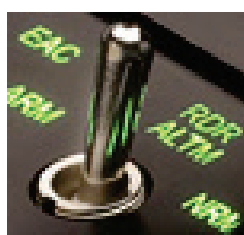
A few button examples:



Temporary push button



4-push button Hat



2-position toggle switch

## The axes

The axes are used to control "proportional" commands (steering wheel, joystick). An axis has a physical run comprised of a fixed range of values (0 to 256, for example) according to the position of the control lever; this will be on one of the values within the range. There are two types of axis, namely centred and slider axes:

- A centred axis is fitted with a mechanical system which will return the control to the centre of the value range (such as in a joystick or steering wheel).
- A slider axis will not necessarily have a mechanical recall system. If there is a recall, it will be to one of the extreme axis values, such as an accelerator pedal.



T.A.R.G.E.T. allows you to modify the sensitivity of axes, to customize their response characteristics and, if necessary, to create virtual buttons triggered according to the value of the axis.

Axes are therefore generally used to control a mechanical or electronic element which has a large number of values. Depending on their ergonomics, they will be more or less suited to a particular control category:

- An axis controlled by a throttle “lever” will naturally be intended to control an engine speed or an element which may be retracted or partially deployed, such as an air brake.
- An axis controlled by a rotary button will be more adapted to controlling a parameter such as radar range. You are of course free to choose the role of each element.

*Note: Certain simulators do not necessarily allow you to control all parameters via an axis. In that case, you can program your controller's axis to behave like a keyboard: this is called a digital axis. In this way, you will compensate for any gaps in the software, while enhancing ergonomics.*

## The keyboard

In practice, 80% of a profile will be dedicated to reproducing keyboard actions. You must therefore understand the keyboard fully.

The keyboard is, of course, made up of keys: these are no more or less than temporary push buttons. Windows therefore provides us with state change information:

- The key has just been pressed.
- The key has just been released.

From these two pieces of information, we can easily deduce that:

- The key is now pressed.
- The key is **OFF**.

We find the same types of data as on a joystick button. We therefore have the information we need to analyze and reproduce the keyboard behaviour easily.

*Now that we have a better idea of what's involved, the most significant part of the process involves understanding the chronology of an "event" that we want to reproduce with our commands. To get the most out of the possibilities offered by T.A.R.G.E.T., we sometimes need to combine several events to “automate” certain actions. You will create the best configurations by thoroughly analyzing your requirements and the processes involved.*



## 5. Operating principles

---

T.A.R.G.E.T is not just a simple interface: it is also a powerful software driver which is able to simulate virtual peripherals such as a keyboard, joystick or mouse. Its virtual equipment will allow us to fully simulate all of the actions needed to use a software application. These actions will be measured on the axes and buttons of our physical controllers, but will be sent to the software via a single virtual controller.

When creating a profile, you will therefore choose the peripherals that will be used to interact with this virtual controller.

Here is what happens when a profile is launched:

- The physical controllers are disconnected virtually.
- The virtual controller is activated.

You can then launch the simulator of your choice.

Some advantages of this system are:

- Total interaction between different USB controllers.
- Configuration in simplified simulators.
- Reduces the number of operations: one global profile for everything, rather than one profile for each controller.
- Guaranteed compatibility when using older software.

T.A.R.G.E.T provides you with two tools, to help you get the most out of its features:

- A very easy-to-use graphical interface which covers most needs. It generates .fcf and .tmc files.
- A text language interface (Script) to integrate highly advanced functions into your files: the format for Script files is .tmc.

Its two methods are complementary: one is highly accessible, while the other is very powerful.

*Note: The files created with the graphical interface may be modified with the script editor, in order to enhance them with very advanced possibilities. However, the opposite is not possible.*

The default location for script files created via the graphical interface in Windows 7 is the following:

C:\Users\USERNAME\AppData\Roaming\Thrustmaster\TARGET\Scripts

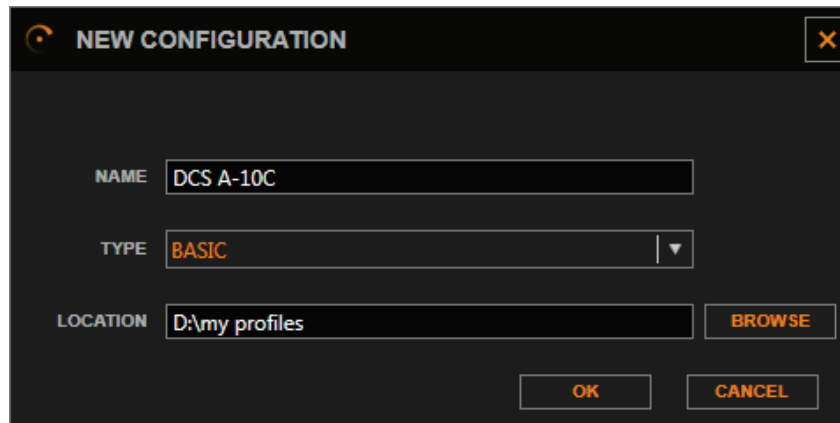


## 6. Creating a Basic configuration

When you launch T.A.R.G.E.T, you arrive at the main **Dashboard** page.

To create a profile, click the **New Configuration** button in the right-hand **My Controllers** window.

A **New Configuration** window then appears.

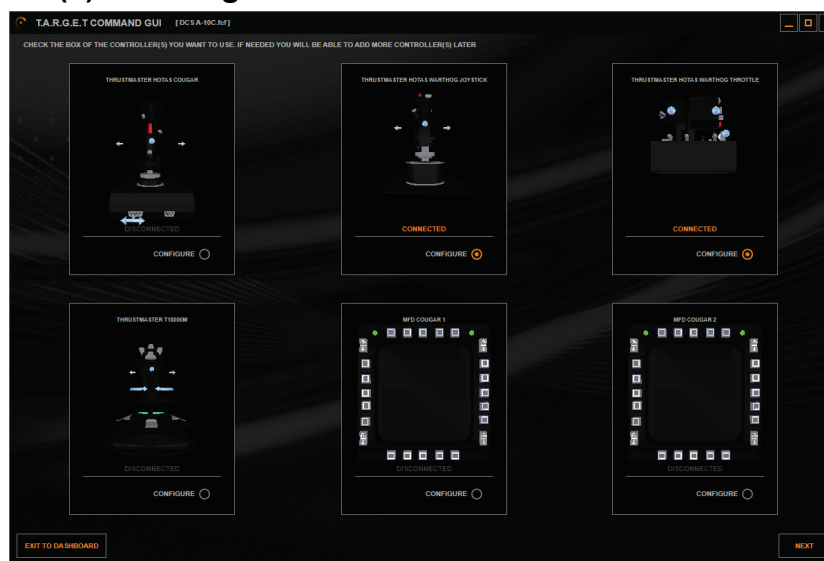


- **Name** will be filled in with the name we want to give the configuration.
- **Basic** or **Advanced** will be selected according to the complexity of the key assignments we wish to create.
- **Location** determines the directory where the file will be saved.

For this example, we will choose to create a **Basic** configuration.

Once this information is complete, click **Ok**.

### Select Controller(s) to configure



This page lets you select the controllers you want to use in your configuration. Simply tick the box to the bottom right of each element for that element to be integrated into your configuration.

Once you have made your selection, click **Next** in the bottom right corner.

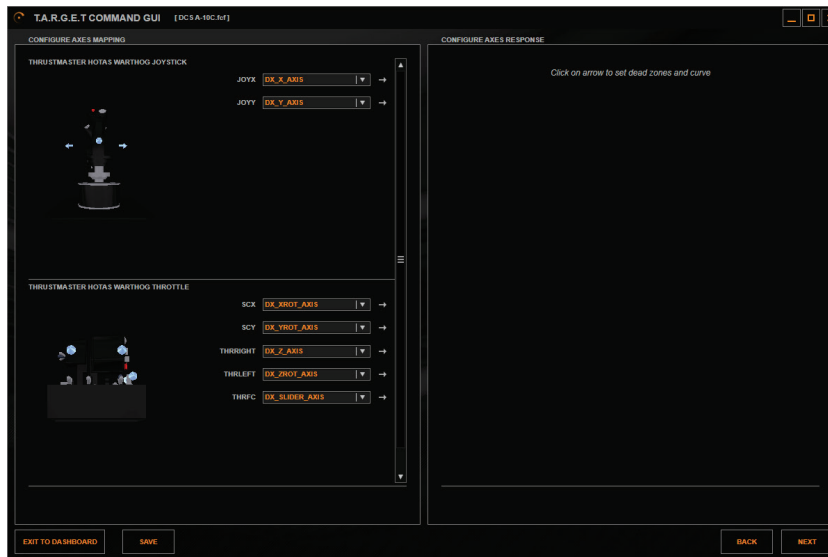
The tick boxes are green for controllers that are currently connected. The boxes are yellow for controllers that are not connected, to warn you that they are missing.



Note: the selected combination is not definitive. You will be able to add or remove peripherals later on using the **Load Configuration** button via the Dashboard.

You can create a configuration without any peripherals being connected, or create configurations which include peripherals you do not have; this will not adversely affect the proper functioning of your configuration in any way.

## Configure axes mapping



This page lets you link the physical axes of your controllers with the axes of the T.A.R.G.E.T virtual controller and refine their sensitivity.

Note: the selected combination is not definitive. You will be able to adjust these values later on by using the **Load Configuration** button via the Dashboard.

The left-hand window called **Configure axes mapping** is a list of the axes provided by your peripherals. As the axes' names are not necessarily very clear, simply place the mouse pointer over the name of the axis (in blue) to see the 3D representation of the controller become animated, and the arrows corresponding to the axis flash.

Next to each axis is a drop-down menu which allows you to assign one of the 8 axes supported by Windows (via DirectX). We suggest that you keep the default configuration.

The standard assignments for the axes are as follows:

| DirectX Axis name | Product Axis Name |               |              |          |
|-------------------|-------------------|---------------|--------------|----------|
|                   | Script Axis Name  | HOTAS WARTHOG | HOTAS COUGAR | T-16000M |
| X                 | DX_X_AXIS         | JOYX          | JOYX         | JOYX     |
| Y                 | DX_Y_AXIS         | JOYY          | JOYY         | JOYY     |
| RZ                | DX_ZROT_AXIS      | THR_LEFT      | RUDDER       | RUDDER   |
| Z                 | DX_Z_AXIS         | THR_RIGHT     | THROTTLE     |          |
| RX                | DX_XROT_AXIS      | SCX           | RDR_X        |          |
| RY                | DX_YROT_AXIS      | SCY           | RDR_Y        |          |
| Slider0           | DX_SLIDER_AXIS    | THR_FC        | MAN_RNG      | THR      |
| Throttle          | DX_THROTTLE_AXIS  |               | ANT_ELEV     |          |



You can also control the mouse axes by linking them to 2 axes. We do, however, recommend that you assign them the “relative” mode, using the **Configure Axes response** panel.

With certain peripherals such as the HOTAS Cougar, you will have to load a specific program into the Cougar Control Panel to get the most out of its possibilities. In particular, you will have to make the RDR CURSOR axes visible.

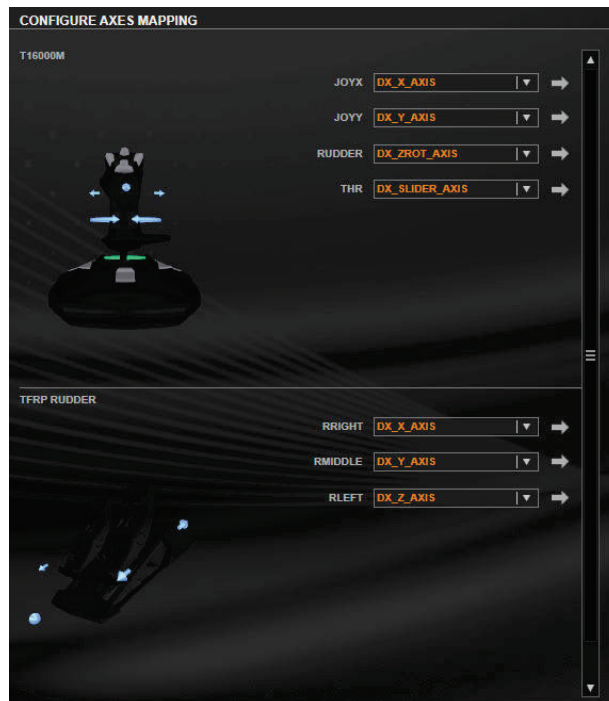
**WARNING:** the default mapping for axes used by T.A.R.G.E.T employs the original default controller mapping. Most of the time, this may result in redundant uses for axes. These redundancies may be a source of control conflicts or strange “in-flight” reactions. To avoid this, be sure to select different DX axes for each physical axis on the controller. DirectX standards limit the number of input axes per device to 8. If you create a virtual device using a combination with more than 8 physical axes, you will have to unlink/disable some of these axes. You will be able to use the disabled axes digitally – using them for keyboard emulation, for example (please see the “Programming an axis” section of this manual for more information).

To disable a physical axis linking to a DirectX axis, simply select the status “NONE”.

Example:

#### Default configuration of a Thrustmaster T.16000M joystick combined with a TFRP rudder

The rudder shares axes already used by the joystick (DX\_X\_AXIS and DX\_Y\_AXIS). This will create a conflict and make your favorite game unplayable as soon as you start to use the TFRP rudder.

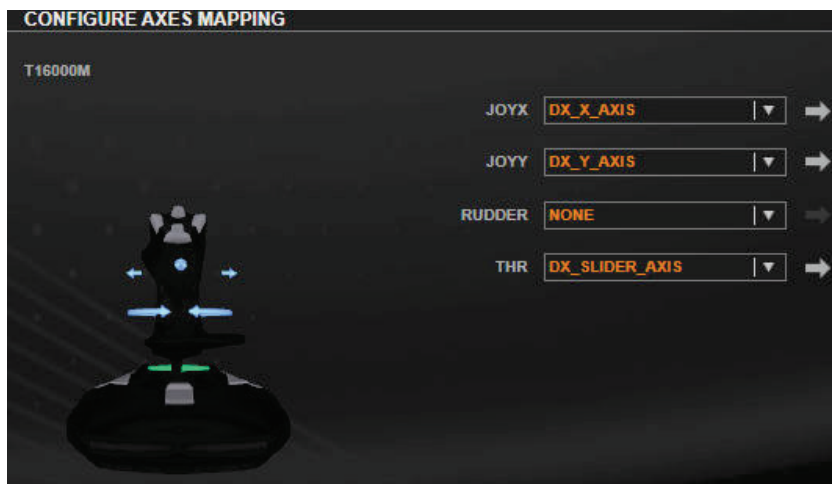


Use the interface to replace both the TFRP RUDDER RRIGHT and RMIDDLE DirectX axes with an unused axis in each instance.

Doing so will resolve the conflict:



If you no longer want to use the T.16000M joystick's Twist grip axis (usually used for a rudder), simply set it to "NONE".



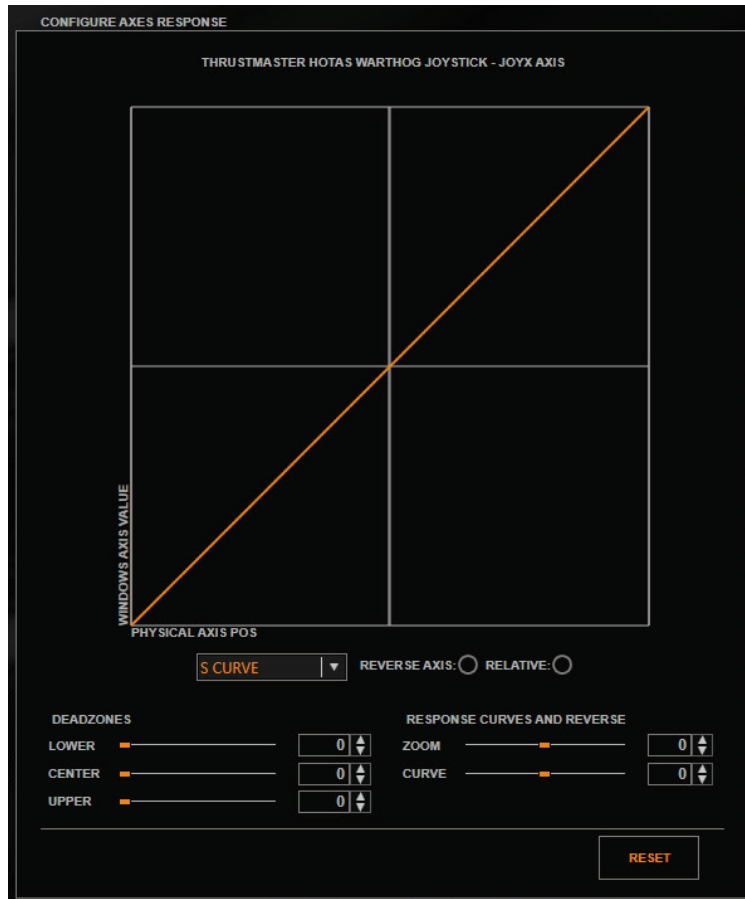
## Configure axes response

This window lets you configure the sensitivity for each axis.

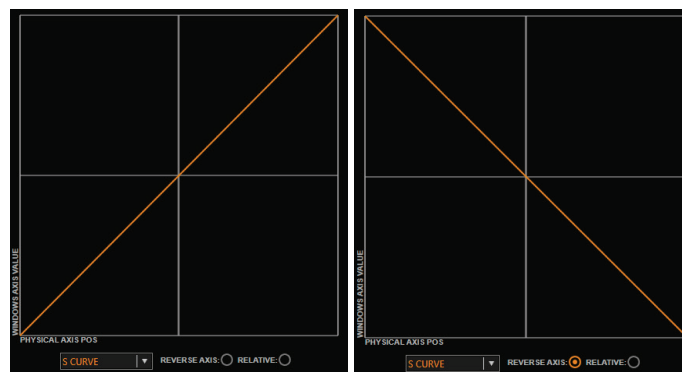
For example, if you find that the joystick is too sensitive around the neutral point when you are flying in a game, you can modify the axis response curve in order to soften the reactions around the neutral point.

To customize a curve, simply click on the green arrow to the right of the axis name that you want to edit in the **Configure axes mapping** window.

The following graph appears:



- **J Curve & S Curve** type window. Depending on the type of axis or function assigned to the axis, you will need to specify if you want symmetrical adjustments in relation to the centre of the axis, or along the whole axis. We will use the **S Curve** parameter for all axes which have a mechanical recall to the neutral point, or which “work” from a central position. The **J Curve** type will be used for all “Slider” type axes (with no recall to the centre) such as an engine speed control axis, a brake, etc.
- **Reverse Axis:** by ticking this box, you will reverse the operating direction of the axis, and the minimum value will become the maximum. In DCS Flaming Cliffs 2 and Blackshark, you will need to reverse the SCY axis if you use the Warthog Throttle Slew Control.

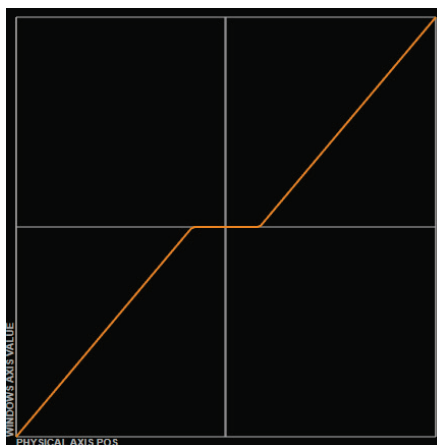


- **Absolute or Relative:** by default, your axes operate in an “absolute” manner: the value sent to DirectX directly reflects the position of the axis. If you are using mouse axes, however, you will see that this control mode is unsuitable: if your axes are equipped with a recall to neutral, the mouse pointer will constantly return to the centre of the screen. To avoid this, simply switch the axis to **Relative** mode by ticking the corresponding box: in general, this is vital when using a mouse pointer. In applications such as Lock On Modern Air Combat and its add-on Flaming Cliffs 1, this function lets you correct the behaviour of the TDC (Target Designation Cursor) displayed on the HUD. By default, assigned to an axis, the TDC will behave in an absolute manner, when it should move in a relative manner.

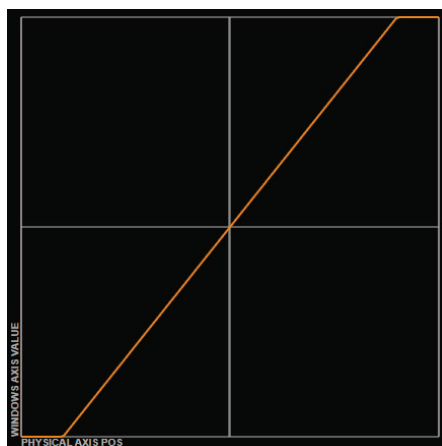




- The **Deadzones** are physical travel zones on an axis where nothing will happen (the axis will not change value) if you move the joystick. There are two types:
  1. The central deadzone, which will be used for axes which have a mechanical recall to the centre. This lets you create a rest zone, if you want to have a neutral zone in the centre.



2. The Deadzones at the ends of the axis will be used to limit the travel necessary to attain the minimum and maximum values before reaching the physical stops. This increases the sensitivity.

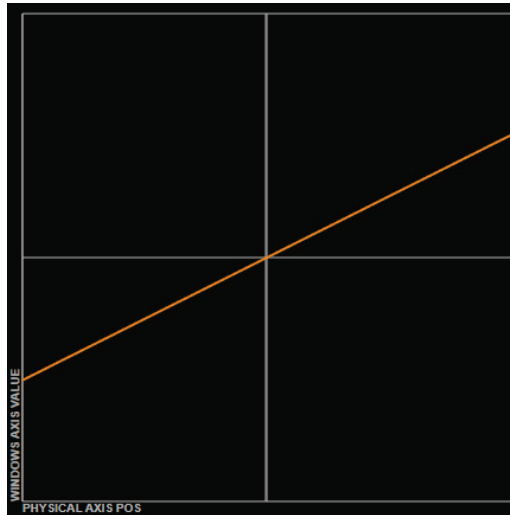


- Zoom lets you limit or increase the sensitivity of the axis in a linear manner.

Example: If you want to control a designation slider in an application, and in practice it turns out to be too sensitive, simply set the Zoom parameter to negative values to directly reduce the sensitivity of the axis. In fact, the zoom in the negative values is a travel limitation (we zoom into the central values, which are projected onto the total run of the axis). You will see that this parameter is very useful for setting the “Slew Control”, which is much too sensitive in most applications.

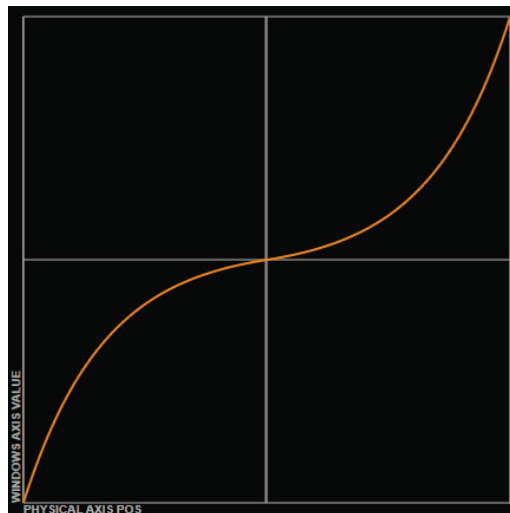
The travel limitation may cause the axis not to be detected in the simulator’s control panel. We therefore recommend that you leave Zoom at zero when assigning axes in the simulator. Once the detection has been carried out and saved, you can edit your T.A.R.G.E.T configuration and adjust the values according to your requirements.

*Note: the zoom function is different from a curve setting, as if you set in to negative on an axis dedicated to flight controls, you limit the travel of controls, and therefore the maneuverability of the plane. To alter a flight control’s sensitivity, it is best to use the “curve” function, which can reach all the values on an axis, regardless of the setting.*

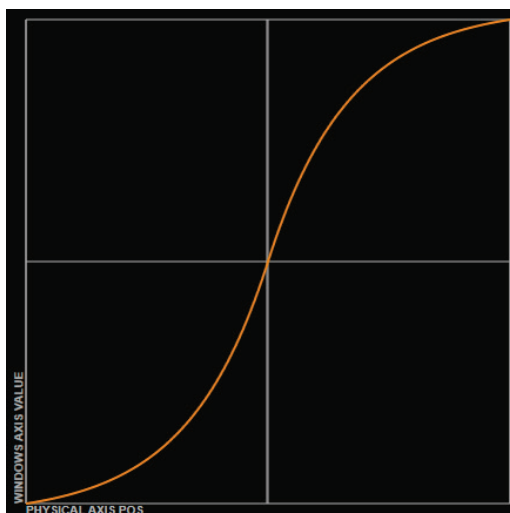


- The curve is used to modify the sensitivity of the axis.

Principle: in the zone where you are seeking precision: on the graph, the weaker the “slope”, the more precise the control will be.



If you choose the **S Curve** parameter, a positive setting will reduce the sensitivity of the control around the neutral point and will increase responsiveness at the extreme values. A negative setting increases the sensitivity around the neutral point and decreases responsiveness at the extreme values.



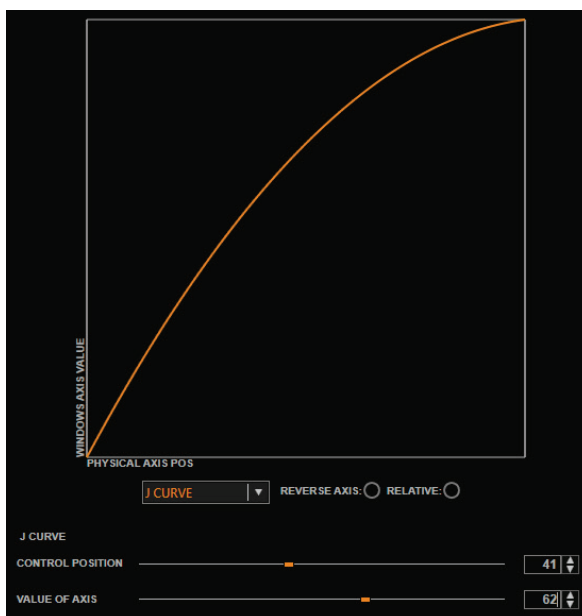


If you select **J Curve**, this functions differently: you will be able to specify an axis value to be reached when the axis control is at a certain position.

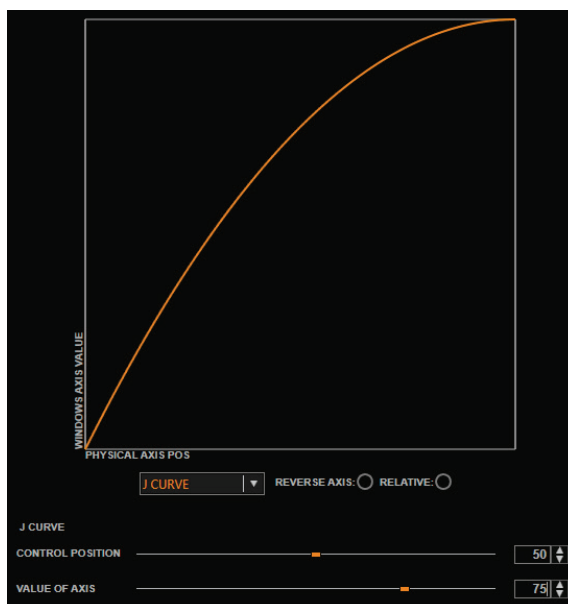
- The **Control Position** slider corresponds to the controller's physical position.
- The **Value of Axis** slider corresponds to the axis value sent by DirectX to the simulator.

As the 2 parameters are interdependent, you must proceed by making small corrections. There are several ways to use this setting.

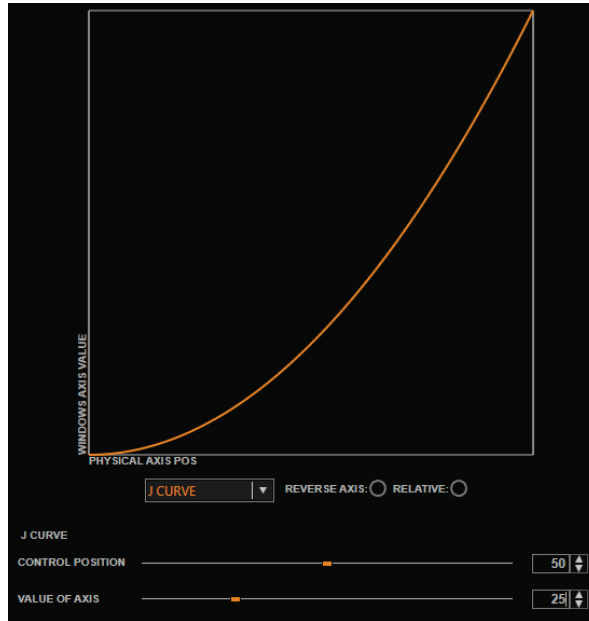
Example for an axis which controls engine speed:



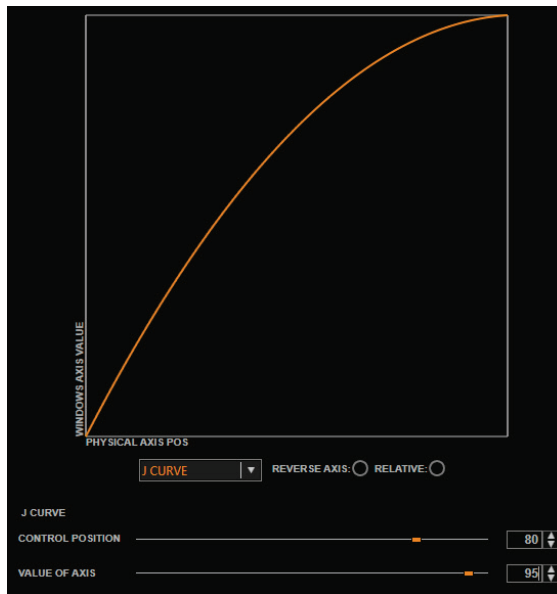
To obtain very precise control at high speed, increase the **Value of Axis** setting to around 75. With this setting, we have reached 75% speed at 50% of **Control Position**, the physical run of the axis. We therefore still have 50% of the upper run to control the last 25% of speed.



For very precise control at low speed, we will keep the **Control Position** at 50% and set the **Value of Axis** at 25%.



To synchronize the afterburner trigger value with that of the HOTAS Warthog detent, we will set the **Control Position** at 80% (as the breakdown of the run on the HOTAS Warthog throttle is 80/20), and we will adjust the **Value of Axis** according to tests in the simulator. For example, for the DCS family, the afterburner is triggered at 95% of engine speed; therefore, we will choose 96%, to be sure that the afterburner does not trigger when we arrive at the stop.



Now that we have defined and set the behaviour for our axes, we only need to assign events to the buttons on our controllers.

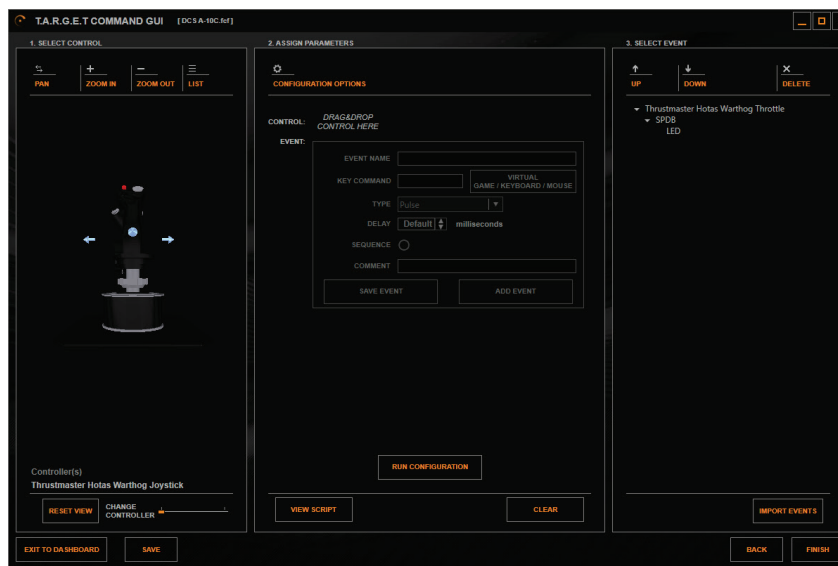


## 7. Graphical User Interface

We are finally at the page that will allow us to assign functions to our buttons. This is where most of the editing is done.

This page contains 3 windows.

- Select Control
- Assign Parameters
- Select Event



### Operating principles

We will associate events with a button or axis. These events may be “made-to-measure”, or selected from a list that we will have imported.

If you have a list of prefabricated events (macros), the configuration will be limited to:

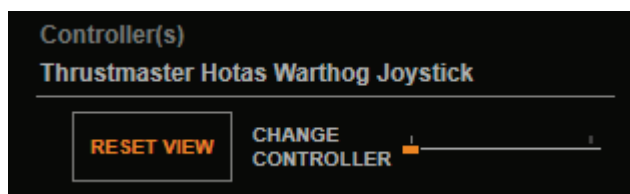
- Selecting a button in the left-hand window and sending it to the central window.
- Selecting an event in the right-hand window and sending it to the central window, which contains the button selected earlier.
- Possibly defining the type of behaviour, and then saving the button/events association.

This operation will be repeated as many times as necessary.

### The “Select Control” window

This window contains a 3-dimensional representation of one of the controllers selected to create the profile. Here you will specify the controller, and button or axis you want to program.

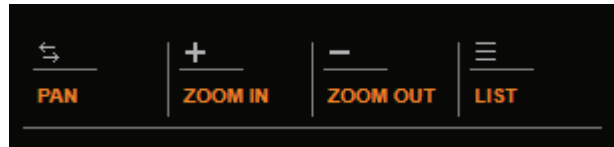
The bottom of the window contains the name of the controller and a **Change Controller** slider. This slider is used to scroll through the different devices that you have selected in your “configuration”.





The upper toolbar lets you modify the viewing angle for the device. It contains 4 buttons:

- **Pan** pivots the controller; a click plus a slight movement of the mouse stops the rotation.
- **Zoom In** lets you bring the controller closer for better visibility.
- **Zoom Out** moves the controller away, for a wider view.
- **List** lets you display a complete list of the buttons and axes available on the device. Once you are familiar with the names of the buttons, this function will save you precious seconds. At the same time, each time the mouse pointer passes over a button, the 3D object pivots to show which control it is.



You can also interact with the 3-dimensional view using only your mouse:

- You can turn the controller along 3 axes by clicking on the 3D view and holding down button 1 on your mouse.
- You can shift the 3D object and the point of view by clicking in the 3D view with button 2 on your mouse and holding it down.
- The wheel on your mouse lets you zoom on the controller, exactly like the **Zoom In** and **Zoom Out** buttons.
- Double-clicking a button automatically selects and sends it to the **Assign Parameters** window.

## Selecting a button

You can use the mouse to select a button or an axis (symbolized by a blue arrow) on the controller's 3D view.

When your pointer passes over a button, the button lights up in green. Simply click on it and drag the mouse pointer into the **Assign Parameters** window, keeping the mouse button pressed down. Double-clicking lets you avoid you having to drag the button.

Once you are over the **Assign Parameters** window, just release the mouse button. The button name then appears beside **Control:**. All we need to do now is assign it an event.

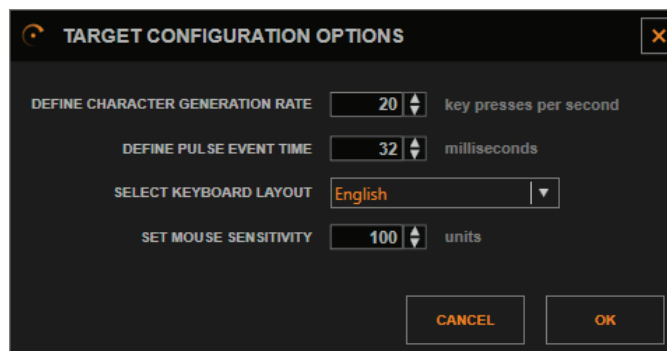




## The “Assign Parameters” window

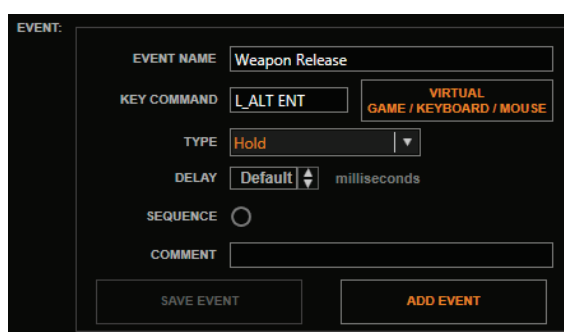
We will determine the actions and their behavior in this window. The contents may vary, according to the type of control to be “programmed”.

At the top of this window, you will find the **Configuration Options** link. This page lets you define a variety of operating parameters. We recommend that you leave them at their default values. However, you may need to adjust them with certain old applications or if you encounter some local keyboard language issues.



- **Define character generation rate:** this parameter lets you set the speed for sending keyboard characters. You may need to modify it if you see that the keyboard buffer is saturated (in that case, you will need to increase this value).
- **Define pulse event time:** this parameter defines the time for which a temporary keyboard combination is pressed (when a key is pressed and then released right away). If you see that some of your pulsed keys are not taken into account, increase the value for this parameter.
- **Select keyboard layout:** this option is used to define the type of keyboard that will be used to enter keys in the Wizard (please see the [Graphical User Interface](#) section of this manual for more details).
- **Set mouse sensitivity:** Define the mouse cursor speed when the mouse is controlled with a button (like a 4-way hat).

Now let's focus on the main tool of the "Assign Parameters" Window: the Event configuration panel.



For a button:

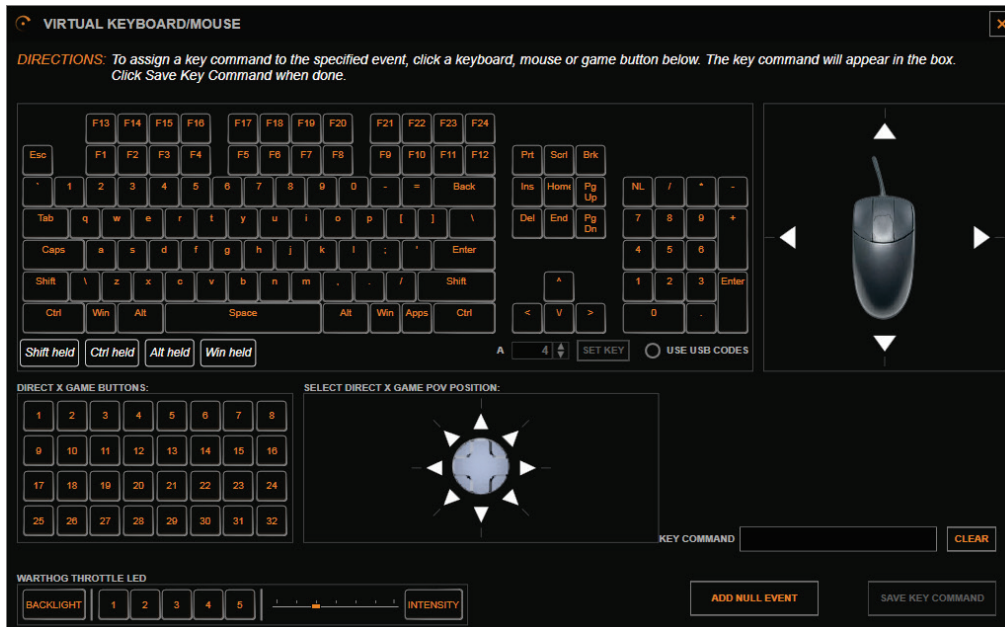
The **Event Name** field must be filled in using the keyboard. Its role is limited to displaying the name of the function we are going to create. This is the name which will be used in the **Select Event** window tree structure: therefore, it is important that it be simple and clear. Examples: **Fire Missile**, **Deploy air brake**, **Retract air brake**, **Navigation lights ON**, etc.

The **Key Command** field is an indicative field. It can only be filled in via the window activated by the **Virtual Game/Keyboard/Mouse** button located to its right. It indicates the keyboard combination entered for this “Event”.



The **Virtual Game/Keyboard/Mouse** button is used to open a window depicting the virtual keyboard/mouse/controller. This window is used to define the combinations of keys, control backlighting and LED or DirectX buttons which will be used in the function.

This is a keyboard that we will click on with the mouse. For example, for "Alt+B" we will click on the "Alt" key and then the "b" key. We will see our combination in the **Key Command** field in the bottom right.

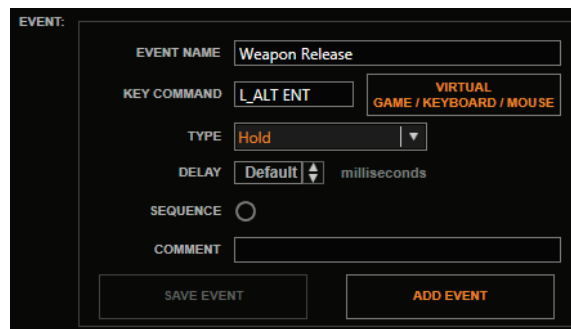


Once you have entered the combination, click the **Save Key Command** button located just below.

You must remember to click this button each time you have finished working with an event, whether during its creation, or after making a subsequent modification.

This virtual keyboard also allows you to generate keyboard codes using the **Use USB codes** option. This is the best way to create a file which is compatible with different international keyboards.

The **Key Command** field in the **Assign Parameters** window now contains the combination we have just entered.



The **Type** field defines the type of behavior for the key combination:

- **Pulse** simulates a brief (temporary) press on the keyboard, even if the button remains activated or in the "on" position.
- **Hold** simulates pressing and holding the keyboard keys until the button is released.
- **Press** simulates a continuous press on the key combination, even if the joystick button is released.
- **Release** releases a combination of keys that has been "pressed" before.





You must pay attention when using **Press** and **Release**: if you use a **Press**, the keyboard key will be virtually held down indefinitely. The only way to release it will be to use a **Release**. If you use a **Press**, you must use a **Release** on the same key. As there is a major risk of locking a key down virtually, you must be very careful in this regard.

The **Delay** field lets you define a duration (in milliseconds) before executing the keyboard event. It may be useful when generating several events from a single button.

*Usage example:*

*We want to simulate a press on the spacebar for 2 seconds. Simply assign 2 functions to the same button: the first will contain the **Key Command: SPC** and **Type: Press** settings. The second will again use **Key Command: SPC**, but **Type: Release** with a **Delay: 2000** [milliseconds]. When the button is pressed, a spacebar press order will be sent, and then (2 seconds later) a spacebar release order will be sent.*

The **Sequence** box lets you vary the event generated. An event list is created, and each time the button is pressed, an event from this list will be sent.

The **Comment** field is simply used to note any comments you may have. It will be very useful to remind yourself why you have used a particular function, if the event is not the only one associated with this button...

The **Save Event** button must be pressed to save the action and its assignment to the button, even if you are simply modifying a parameter.

The **Add Event** button lets you add several events to a single button.

## Assigning several functions to the same button

It is sometimes necessary or practical to be able to assign several events to the same button. There are two separate ways to do so:

- Create a chain where all the events will be sequenced in a few milliseconds (except if some of them have a defined **Delay** value).
- Create a sequence where a different event will be generated each time a button is pressed.



## Creating a chain

Select the button of your choice and send it to the **Assign Parameters** window.

Create the first "Event" and save it by clicking the **Save Event** button.

The contents of the **Assign Parameters** window has not changed, but the **Add Event** button is now clickable. We won't touch it for now, though.

Modify the **Event Name** field and the options to create the second event.

Once you have defined the new event, click **Add Event**.

The event is then saved and added to the tree structure in the **Select Event** window. You can add as many of these as you like.

*Note: you can choose the event sequence order within a chain. In the **Select Event** window, select the event that you want to move or cancel. In the **Select Event** window toolbar, the **Delete**, **Up** and **Down** buttons light up: they are used to move or delete an event.*

## Creating a sequence

Select the button of your choice and send it to the **Assign Parameters** window.

Create your "Event" by ticking the **Sequence** box.

Click the **Save Event** button: the event is created in the tree structure in the **Assign Parameters** window. It is preceded by an [S] to indicate that it is part of a sequence.

To add a new event to this sequence, modify the different fields that define the "Event", and then click **Add Event**.

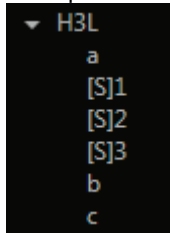
The second element in the sequence then appears in the tree structure in the right-hand window.

You can add as many events as you like.

Using the **Select Event** window toolbar, you can change the order of elements in the sequence, and delete elements.

You can insert a sequence into the middle of a chain.

Example:



Here, each action will generate the following

keyboard codes:

1<sup>st</sup> press: a1bc

2<sup>nd</sup> press: a2bc

3<sup>rd</sup> press: a3bc

4<sup>th</sup> press: a1bc

... and so on.

## LED Management

T.A.R.G.E.T allows you to manage the LED and backlighting power of the HOTAS Warthog Throttle and the MFD Cougar Pack.

There are 2 different kinds of lighting control:

- **LED:** you can control the status of the 5 green LEDs on the Warthog Throttle and the Left and Right LEDs on each MFD Cougar.
- **Backlighting:** you can control the backlighting intensity from null to full. The Warthog throttle offers 6 levels of intensity, while MFDs have 256 different levels of intensity.

To be able to perfectly manage the lighting features of these devices, we suggest that you create an "Advanced Configuration".

All the lighting actions are events: you will not find them in the wizard **control** list or **imported event** tree, but only in the Virtual Keyboard/Mouse window (accessible via the **2. Assign Parameters** wizard area). You must link these light events to one control (button or axis).

### Main rules:

LED commands can only be defined via the **Virtual Game/Keyboard/Mouse** panel.

To light up a LED, you must give it the **Press** event type. To shut it off, simply use the **Release** event type.

If the Warthog Throttle **Backlight** has been released, don't expect to light up anything on the Throttle (backlighting).

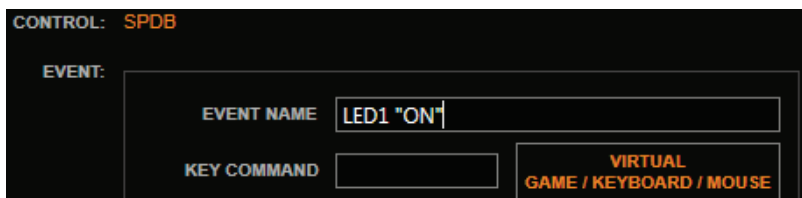
If the MFD **Intensity** has been set to zero, you will not be able to see any "ON" LEDs.



**Backlight** power cursor: Full left is zero, full right is 100% of lighting.

### Control a LED

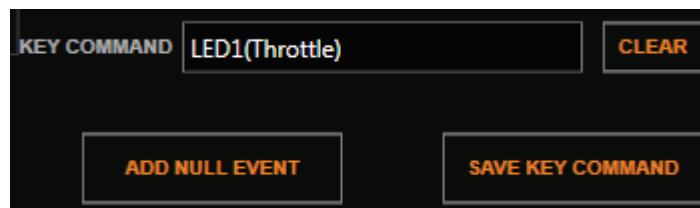
While editing a GUI configuration, first choose a control in the **1. Select Control** left area of the main wizard page. When you're done, name your event (such as "LED1 ON") in the **Event Name** dedicated field of the **2. Assign Parameters** window in the center of the screen.



Now it's time to select the LED that you want to control, so let's click on the **Virtual Game/Keyboard/Mouse** button on the right of the **Key Command** field.



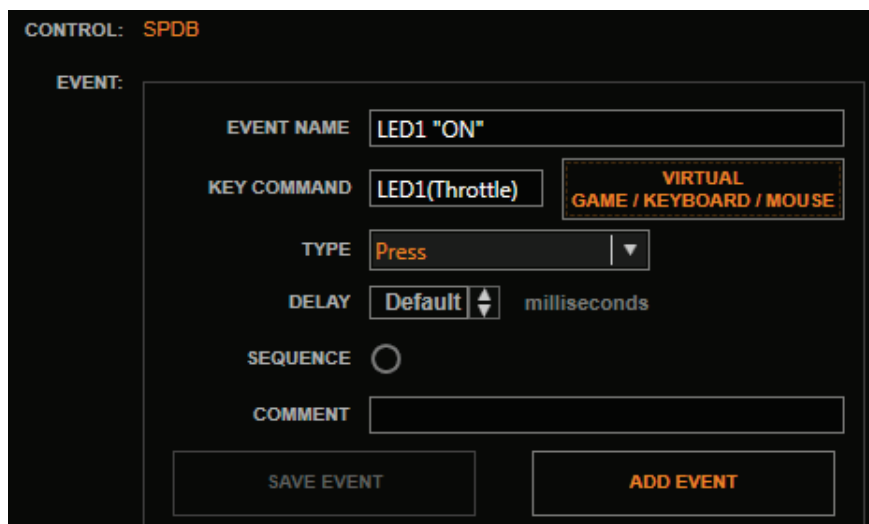
In the lower left corner of the new window, you will find a line used to control the Throttle lighting features. Click on the **1** square button and check that the **Key Command** field is displaying the right LED. When you're done, click the **Save Key Command** button in the lower right corner of the screen.

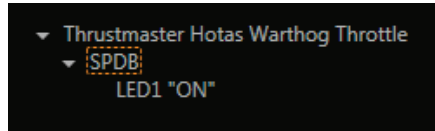


Now that we're back in the main Wizard screen, check the type of event defined for your control:

- **Press** will turn "ON" the LED.
- **Release** will turn "OFF" the LED.

Once you've defined the Event type you need, simply click the **Add Event** button. Your new event is now displayed in the "Event tree" of the **3. Select Event** window.





Typical example: When I lower the landing gear, I also want to activate HOTAS Warthog Throttle LED 1. When I raise my landing gear, I want LED 1 to be shut off.

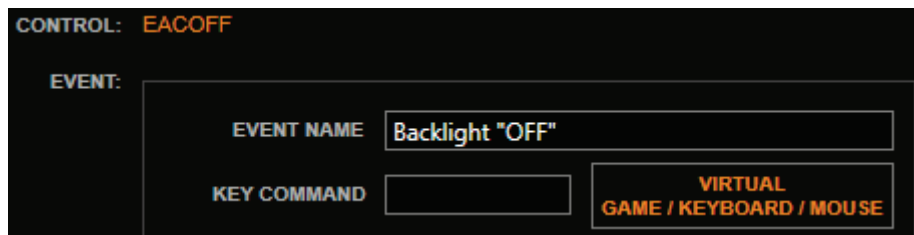
Our favorite simulator offers 2 keystroke commands to manage landing gear positions: "Shift + g" to lower gear, "Ctrl + g" to raise gear. What we have to do is simply program 2 events each time there's an action on the gear button: the landing gear keystroke command and the LED control order.

Read the **Creating a CHAIN** chapter (page 23) to create multiple events generated by one action on a control.

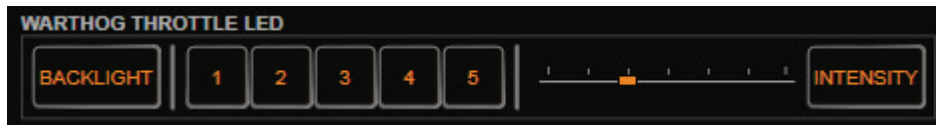
### Backlight intensity assignment

While editing a GUI configuration, first choose a control in the **1. Select Control** left area of the main Wizard page. Here we will use the HOTAS Warthog Throttle EACOFF button.

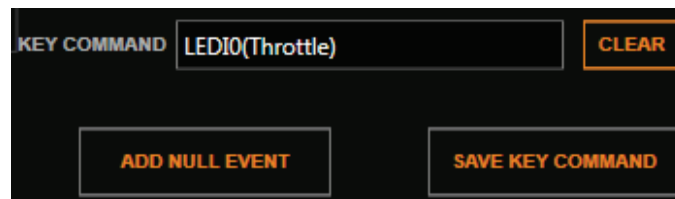
When you're done, name your event (such as "Backlight OFF") in the **Event Name** dedicated field of the **2. Assign Parameters** window in the center of the screen.



Now it's time to select the backlighting intensity you want to apply to the HOTAS Warthog throttle. Click on the **Virtual Game/Keyboard/Mouse** button on the right of the **Key Command** field.



In the new window, in the lower left corner, you will find a line used to control the Throttle light features. Move the green slider completely to the left: this will define a zero Backlight value. Once the cursor moved, click the **Intensity** button. The "LEDIO(Throttle)" command should appear in the **Key Command** field in the lower right part of the window. When you're done, click **Save Key Command**.



Now that we're back in the main Wizard screen, check the type of event defined for your control. Notice that you can't change the **Type** of event, as we have already defined the power status of the backlight in the previous screen. Now, simply click the **Add Event** button. Your new event is now displayed in the "Event tree" of the **3. Select Event** window.



CONTROL: EACOFF

EVENT:

EVENT NAME: Backlight "OFF"

KEY COMMAND: LED10(Throttle) VIRTUAL GAME / KEYBOARD / MOUSE

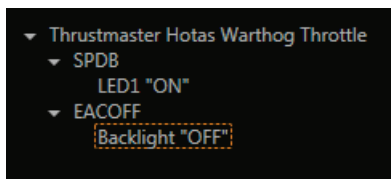
TYPE: Pulse

DELAY: Default milliseconds

SEQUENCE:

COMMENT:

SAVE EVENT ADD EVENT



If you want to turn "ON" the Backlight, you simply have to create the same configuration on another button or create a sequence (see the multiple outputs chapter); except that in the **Virtual Keyboard Mouse** window, the green slider for the HOTAS Warthog throttle must be in one of the 6 right positions.

## 8. Programming an axis

If you want to allow an axis to trigger keyboard commands, simply select it and send it to the **Assign Parameters** window.

A key is defined in the same way as for a button.

You can generate a keyboard event from an axis. Simply define a value range on the axis which will trigger the event. If you want to generate several events, simply create several zones.

*Note: the zones cannot overlap.*

There are 2 types of programming for an axis: Axmap1 & Axmap 2.

### Axmap1: generates events that depend on the axis movement direction

Use this when the simulator offers one key to increase a value and another key to decrease it (engine power or zoom for, example).

Axmap1 mode is quite easy to use.

First fill in the **Event Name** field, then the **Up Key** and **Down Key** fields. **Center Key** is optional.

**Markers** define how many events are going to be generated when you move an axis from its minimum value to the maximum value.

If in the simulator you need to press a key 5 times to go from the lowest radar range to the highest one, simply set the **Markers** value to 5 areas. When you're done, simply click the **Add Event** button.



CONTROL: THR\_FC    Axmap1    Axmap2

EVENT:

EVENT NAME: Increase & decrease RADAR range

UP KEY: u    VIRTUAL GAME / KEYBOARD / MOUSE

CENTER KEY: Null    VIRTUAL GAME / KEYBOARD / MOUSE

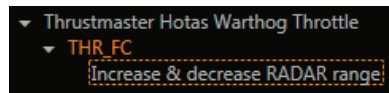
DOWN KEY: d    VIRTUAL GAME / KEYBOARD / MOUSE

MARKERS: 5 areas

COMMENT:

SAVE EVENT    ADD EVENT

The axis programming event name should now be displayed in the "event tree" of the **3. Select Event** window.



### Axmap2: generates an event when the axis reaches a predefined value range

Imagine that we want to control a Target Designation Cursor in a simulator with the HOTAS Warthog Throttle Slew Control.

Each axis will be divided into 3 zones. The following is an example for the X axis:

Slider to the left / central deadzone / slider to the right

We will choose a central deadzone of 20% of the run. We must therefore create  $(100-20)/2 = 40\%$ , i.e. 2 zones of 40% at the ends of the axis.

Let's start with the zone on the left. We'll define the zone using the 2 red sliders. For more accuracy, you can also click the range adjustment buttons using your mouse.

EVENT NAME: LEFT

KEY COMMAND: LARROW    VIRTUAL GAME / KEYBOARD / MOUSE

TYPE: Hold

DELAY: Default milliseconds

COMMENT:

0%    40%

SAVE EVENT    ADD EVENT

Once this zone has been created along with a **Hold** keyboard event, we will save the EVENT. A pulsed event will not simulate a continuous press on the left directional key, which is vital here.

Now that the first zone has been created, we just need to define the right-hand zone.

Let's modify the contents of the **Event Name** and **Key Command** fields, and create a zone between 60 and 100%.



We can see that a red line indicates the first zone used. Once the changes have been made, click the **Add Event** button.

We have now programmed our axis so that the “Left arrow” and “Right arrow” keys will be generated according to the Slew Control position, while creating a neutral zone in the centre, making sure that no keyboard keys will be pressed when the Slew Control is idle. Dozens of zones can be created per axis.



## The “Select Event” window

This window combines two features.

It displays a tree structure of the functions that you have assigned to your controller's buttons and axes, and lets you manage the priorities when you have several functions associated with the same button.

By clicking the **Import Events** button, you can import the list of events from another file. Simply click on the event and drag it to the **Assign Parameters** window. It is very simple and quick to create a file; you do not even need to know the keys used by the simulator. If you have imported a .fcf type file, you will find all the event characteristics (**Event Name, Key Command, Type, Delay** and **Comment**).

If you import a .ttm file, only **Event Name** and **Key Command** are available.

Once you have carried out your configuration, the window displays a real list of “Events” assigned to each button. Double-click on the event to send it to the central **Assign Parameters** window.



The **Up** and **Down** buttons in the toolbar are used to change the order of events, when several events are assigned to the same button.



The **Delete** button is used to delete an event.

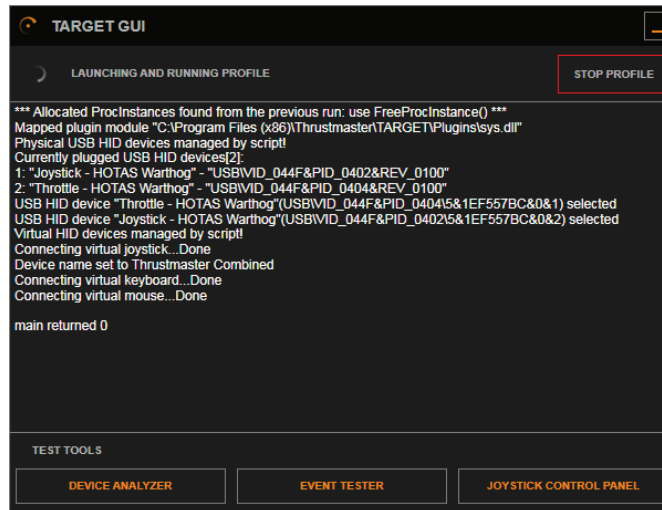
Right-clicking on an event opens a menu allowing you to **copy** and **paste** events from one button to another.

## Launching the configuration

The bottom of the window contains the **Run Configuration** button: it allows you to launch your configuration. A window then appears, describing the standard launch process:

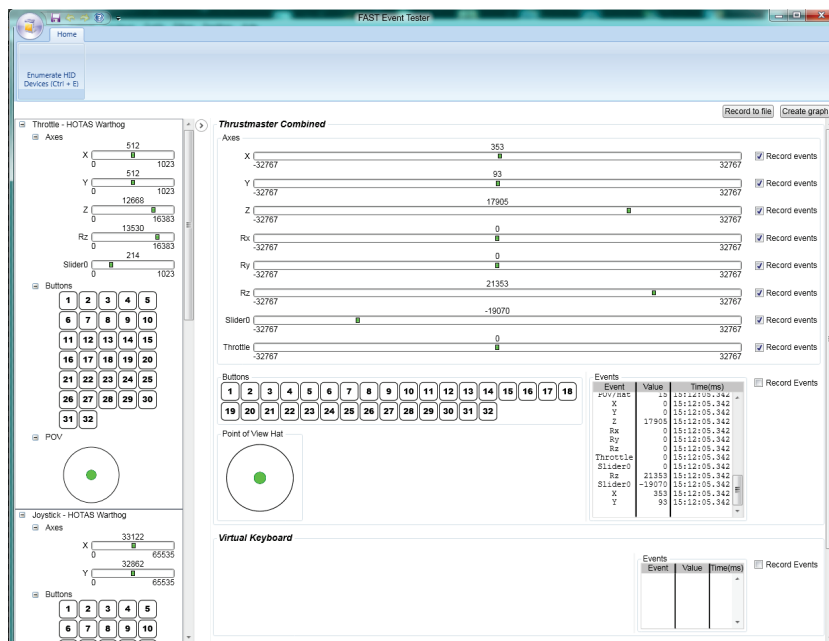
- The physical controllers are disconnected virtually.
- The virtual controller is activated.

Once you have launched the configuration, you now just need to test it.



To test your configuration, you have 3 tools that can be accessed directly via the 3 buttons at the bottom of the **T.A.R.G.E.T Wizard Running Script** window.

## The Device Analyzer

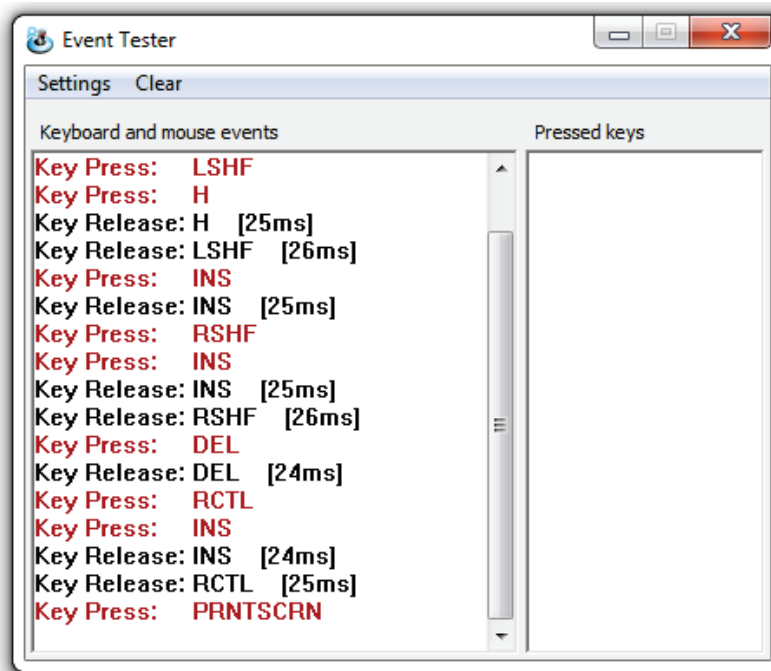


The Device Analyzer allows you to view controllers' axes and buttons. The left-hand window displays the actual status of the different elements, while the window on the right displays the status sent by DirectX to the simulator. It is the ideal tool for testing your reworking of the curves on the axes. You may select other controllers to be tested by clicking on the round button in the top left.





## The Event Tester



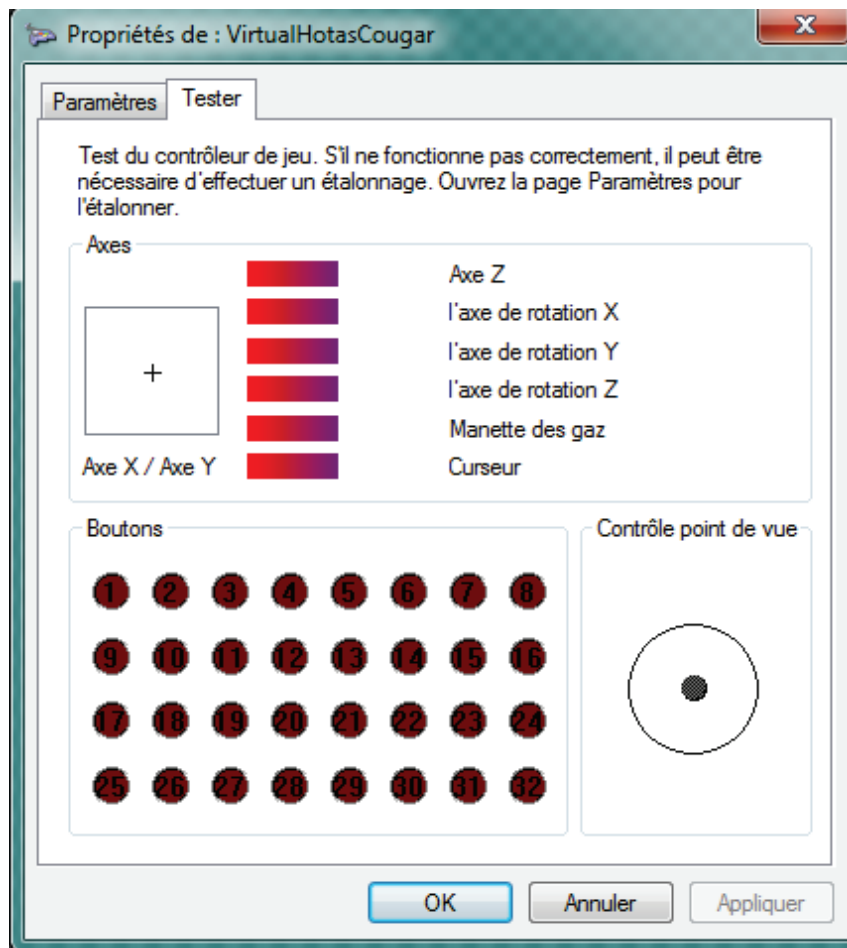
The Event Viewer is intended above all to test the keyboard events generated by your configuration. Each action on the keyboard is broken down into 2 components:

- The moment when the key is pressed: **Key press**.
- The moment when the key is released: **Key release**.

All of the keyboard actions are read as QWERTY. AZERTY keyboard users must therefore take this into account (for example, pressing the **a** key displays a **q**).



## The Joystick Control Panel



This control panel is used to test the DirectX axes and buttons sent by your configuration. It is a simple and effective tool, but will not help you test keyboard events.



## 9. “Basic” exercise

Now that you know the “Basic” mode's features, we will use them in a simple configuration which is adapted to the simulator and the controller. For this example, we will use the LockOn Platinum software (LockOn Modern Air Combat and its add-on: Flaming Cliffs 2) and a HOTAS Warthog; the A-10A will be the plane selected. The objective is not to produce a complete or 100% realistic profile, but simply to illustrate the interaction between the controller, the configuration and the software.

### Fire Weapon

Just like in real life, we will choose the Joystick S2 **Weapon Release** button.

In the software and for this airplane, this function is assigned to the key combination **Ralt + Space** (right Alt key + spacebar).

In Flaming Cliffs 2.0, missile firing requires an extended press on the button. Therefore, a “Pulse” type event will not be suitable, and we will have to select a “Hold” type event, able to replicate the long press.

The screenshot shows the configuration for an event named "Fire Weapon". The key command is "L\_ALT SPC". The type is set to "Hold". The delay is "Default" milliseconds. The comment is "Must be hold for missile". There are "SAVE EVENT" and "ADD EVENT" buttons at the bottom.

### Fire Cannon

This function will naturally be associated with the Joystick's **TG1** trigger. For this “event”, we want the firing to be continuous while the trigger is pressed: the event type will therefore be **Hold**. The key to press is the spacebar.

The screenshot shows the configuration for an event named "Fire Canon". The key command is "SPC". The type is set to "Hold". The delay is "Default" milliseconds. The comment is "hold for long fire". There are "SAVE EVENT" and "ADD EVENT" buttons at the bottom.

### Select Cannon

To select the cannon, simply pressing the **c** key is enough. We will therefore just pulse this key when the **S1** button is pressed.

The screenshot shows the configuration for an event named "Select Canon". The key command is "c". The type is set to "Pulse". The delay is "Default" milliseconds. There are "SAVE EVENT" and "ADD EVENT" buttons at the bottom.



### Using Hats (multidirectional buttons)

It makes sense to group functions together by theme on a Hat. For example, we will use the Joystick's Hat 4 to manage countermeasures.

- When pushed forward, we will enable/disable the continuous decoy release sequence, which corresponds to a pulsed event (**Lshift + q**).
- When pulled back, the Hat will activate the immediate launch of a thermal decoy and a radar decoy with a **Pulse + q** type event.
- Moving Hat 4 to the left will activate a radar decoy release via a **Pulse + Insert**.
- Moving Hat 4 to the right will activate a thermal decoy release via a **Pulse + Delete**.
- Finally, pressing the Hat will activate EMC electronic scrambling via a **Pulse + e**.

### Using the throttle's air brake button

By pressing the **Change Controller** button in the **Select Control** window, we select the Throttle. The air brake button offers three programmable positions: by sliding it back, we have a temporary button, while in the middle and to the front we have stable positions. In the real airplane, the button works like this:

Moving backward, the air brake is progressively deployed.

In neutral, the air brake is locked in its position.

Moving forward, the air brake is progressively retracted.

The Flaming Cliffs 2.0 commands do not allow us to correctly reproduce this behaviour. In fact, we only have all or nothing commands: deploy the air brake, and retract the air brake.

Out of the different possibilities available, we will use the one that benefits from the physical specificities of our 3-position button.

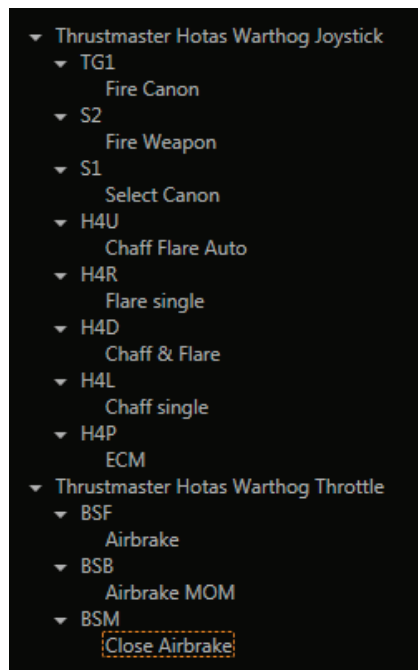
The extreme positions will send a pulse requesting air brake deployment (**LShift + b**), and the central position will send a pulse requesting air brake retraction (**LCtrl + b**).

In this way, we can deploy the air brake permanently by pushing the button forward, or give air brake pulses (very practical for formation flying) by moving the button backwards. In this last case, releasing the button automatically retracts the air brake.

We have converted a lack of realism in the application into an ergonomic advantage, thanks to the button's physical properties.

*Note: for all "toggle"-type buttons which remain in position after user intervention, it is important to ask the question whether the event should be pulsed or continuous.*

Here is what we get in the end:





## 10. Creating an Advanced configuration

NEW CONFIGURATION

NAME: DCS A-10C

TYPE: ADVANCED

LOCATION: D:\my profiles

BROWSE

OK CANCEL

Using an advanced configuration gives you more possibilities and flexibility. In addition to new functions, you will be able to multiply the events assigned to buttons.

From an interface point of view, the Advanced mode only differs in the **Assign Parameters** window, where new buttons appear:

- **Edit Layers** in the toolbar
- **Layers** and the **I, O, U, M, D** buttons
- **Press**
- **Release**

All of these buttons can be used to multiply the configuration possibilities, providing you with maximum control.

### Layers

#### U, M and D

Layers are a little like a building: on each floor, you can have different apartment configurations. In “Basic” mode, you only set up the ground floor. In “Advanced” mode, you can access new floors that you will set up to suit your needs. At this point, we have programmed the default layer for our virtual device. In Advanced mode, we will be able to access 3 other layers. Keeping up tradition, we have named the levels in correlation with earlier Thrustmaster products.

- **U** for the upper layer (Up)
- **M** for the middle layer, which is the default layer
- **D** for the lower layer (Down)

They have been designed to be assigned to a 3-position toggle switch, but may be used on other types of buttons, as we will see below.

You will store different or identical events in each layer. This therefore triples the number of control possibilities for each button.

#### Example:

*In the **U** layer, pressing the Joystick trigger will cause the **Spacebar** key to be pressed; in the **D** layer, the **Enter** key to be pressed; and in the **M** layer, nothing at all.*

*Note: all of the buttons on your Thrustmaster controllers included in your configuration can use these layers: with just a pair of MFD Cougars, you have 168 programmable buttons!*



A logical application of the layers will be to use each layer for a precise situation, for example:

- The **U** layer will be dedicated to Air-Air functions
- The **M** layer will be for navigation
- The **D** layer will be dedicated to Air-Ground operations

When used rationally, the layers may provide you with enhanced comfort and efficiency that you will truly appreciate. To select a layer, simply activate the button that calls it up.

To provide even more possibilities, there is a “sub-layer” within each layer: this is the **I/O Layer**. This sub-layer has been set up to function temporarily, but you are free to change how it operates. Access the **I** layer by pressing a button which will act like a “shift”.

- **I** to indicate that we are working in the layer where the “shift” button is pressed (**In**).
- **O** to indicate that we are working in the layer where the “shift” button is not pressed (**Out**).

Using the **I** layer is perfectly adapted to the use of reverse or secondary functions. The **I** layer will be activated by a button that is preferably temporary and easily accessible, such as the Paddle Switch (S4) on the Warthog handle.

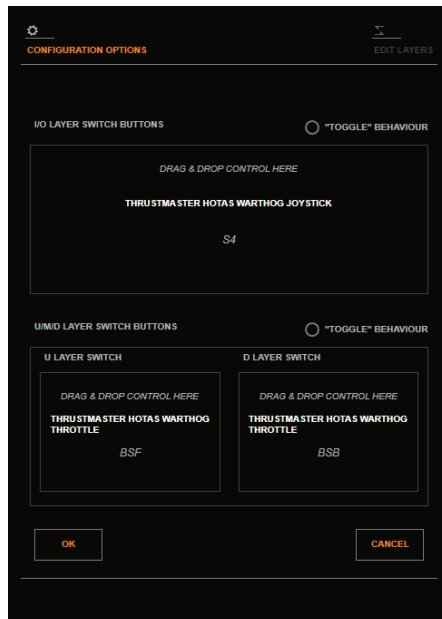
Application example:

- We can program the Slew Control button for the **Lock Target** function. It could be advantageous to program the opposite function on the same button, and access the **Unlock Target** function when we press the paddle (which acts like a Shift). This way it becomes easier to memorize a configuration.
- By default, Hat 1 (Trim Switch) on the HOTAS Warthog handle controls the direction in which the pilot is looking. If you press the Paddle Switch, the button is then used to adjust the trim. This provides enhanced comfort and realism.

By using the layers, you can assign up to 6 different functions per button. The 58 buttons on our pair of MFD Cougars therefore offers 348 programming possibilities!

**The first step in creating an “Advanced” configuration is therefore to define the keys that activate the layers.**

This is set in the toolbar in the **Assign Parameters** window, by pressing the **Edit Layers** button.





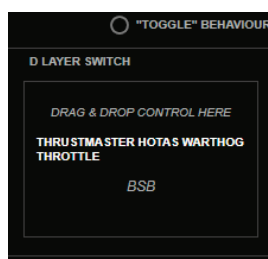
Simply click on a control in the 3-dimensional view in the **Select Control** window, and drag it to the zone that corresponds to the layer you want to control. It is perfectly possible to control the layer selection from different controllers: for example, on the HOTAS Warthog, we will use S4 (Paddle Switch) on the joystick to control the I/O layer, and the Boat Switch on the Throttle to select the U/M/D layers.

*Note: the **M** layer does not have a control button, as it is the default layer: if you are not in one of the other layers, then you must be in the **M** layer.*

By default, the U, M and D layers are intended to be associated with a single 3-position switch, such as the Dogfight switch on the HOTAS Cougar, or the Boat Switch on the HOTAS Warthog's right throttle.

If you want to control the selection of these layers on another type of button, you may need to change the layer's behaviour. In fact, by default the layers are temporary: a layer is active while the button associated with it is pressed. This works perfectly if you use traditional toggle switches (as they remain in the position you activate), but if you want to control the layer selection with the temporary buttons on the base of T-16000M joystick, an MFD Cougar or a Hat, it will soon become complicated to keep pressing this button while activating other buttons. To avoid this, you must change the layer operating mode.

To change the layer's behaviour from temporary mode to an "alternative" mode, simply tick the **Toggle behaviour** box associated with each type of layer in the **Edit Layers** menu.



By switching the I/O layer to toggle mode, a short press will activate the I layer permanently. You will need to press again to return to the O layer.

Now that the buttons and the layers associated with them have been configured, we just need to learn how to use them.

You can use the mouse to select a button or an axis (symbolized by a blue arrow) on the controller's 3D view. When the pointer passes over a button, the button lights up in green.

Simply click on it and drag the mouse pointer into the **Assign Parameters** window, keeping the mouse button held down. Once you are over the **Assign Parameters** window, simply release the mouse button. The button's name then appears beside **Control**:

Or else:

- Double left-clicking on a button automatically selects the button and sends it to the **Assign Parameters** window.
- Clicking the **List** button in the toolbar displays the list of buttons available on the controller. Simply select the one that interests you.

By default, this new event appears in all the layers. To assign it to a specific layer, we must "switch off" the layer or layers in which we don't want it to be present.



Example:

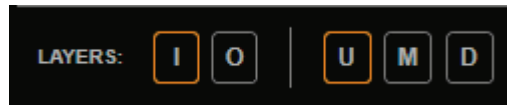
Present in all the layers:



Present only in the U layer:



Present only in the U layer, when the button which calls up the I layer is pressed:



If you want the same event in the U and D layers, you have to replicate it in each layer. If the event is a complex one, use the copy and paste function (accessible via the right-click menu when you place the mouse cursor over an event).

### Press and Release

Beside **Control:**, there are now 2 other buttons, **Press** and **Release**.

At this point, we have programmed events that were triggered when a button is activated. In Advanced mode, it is now possible to generate an event when a button is released. When you program an event, you must specify whether this event occurs when the button is pressed, or when it is released.

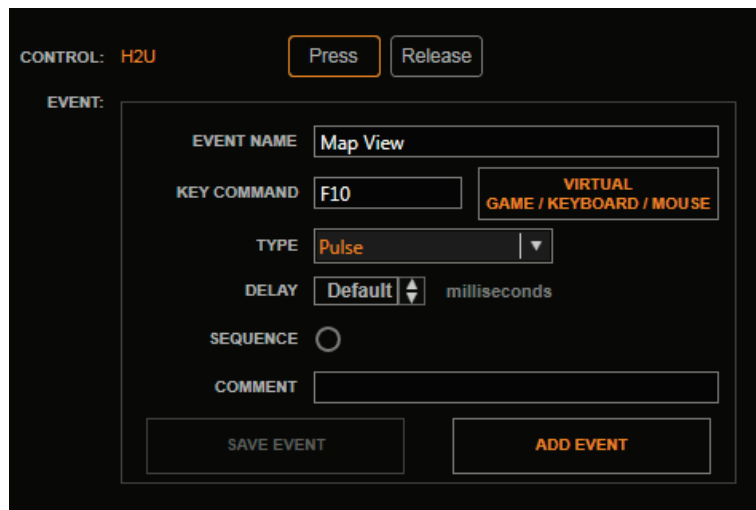
When associated with a temporary button, you can convert an alternative state function to a temporary function.

Example:

You have 2 commands to control the cockpit views and map view. If we program the map view when the joystick's Hat 2 is pushed up ("Press" function), and the return to cockpit view when the Hat 2 is released from its up position to its central position ("Release" function), we can consult the map and return to the cockpit view in the blink of an eye, or rather with a flick of the thumb.

Example:

We start by creating the **Press** event.



We then click the **Save Event** button, and create the release events.



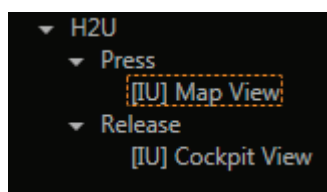


CONTROL: H2U    Press    Release

EVENT:

|             |                       |                                 |
|-------------|-----------------------|---------------------------------|
| EVENT NAME  | Cockpit View          |                                 |
| KEY COMMAND | F1                    | VIRTUAL GAME / KEYBOARD / MOUSE |
| TYPE        | Pulse                 |                                 |
| DELAY       | Default               | milliseconds                    |
| SEQUENCE    | <input type="radio"/> |                                 |
| COMMENT     |                       |                                 |
| SAVE EVENT  |                       | ADD EVENT                       |

In the end, the tree structure in the **3 Select Event** window displays that in the Hat 2 Up position, when the button is pressed, the map view is displayed in the **I, O, U, M** and **D** layers (i.e. in all the layers). When the button is released, the Cockpit view is displayed.



Advanced mode therefore lets you multiply the assignment possibilities for each button. Once you are familiar with this mode, if you want even more flexibility or to get the most out of T.A.R.G.E.T's power, we suggest that you get to know the advanced Script language.

## 11. A few words about the T.A.R.G.E.T text programming language

In fact, since starting out here, you have been using the Script language, as in parallel to the graphical editing, it creates the script that you will execute when you launch the configuration. Script therefore contains the same information that your file creates with the graphical interface. It is perfectly possible to start a file with the graphical interface and complete it "by hand" in the Script Editor. However, the script is "heavier" to handle: you must respect its syntax, which means that you must be extremely careful and take the time to read and understand the manual. However, if you would like more possibilities and control, we encourage you to give it a try.

For more information on the Script language, please refer to the **T.A.R.G.E.T Script Editor Basics User Manual**.



In order to continue to improve its T.A.R.G.E.T software, Thrustmaster invites you to report any bugs related to use of the interface or the software's functionalities, via a dedicated space at:

<http://target-bugtracker.thrustmaster.com>.

**Attention:** this space is reserved for reporting bugs in the T.A.R.G.E.T software only, and must not be used for reporting hardware problems under any circumstances (should you experience any problems with your hardware, please contact Thrustmaster Technical Support). Questions regarding other software such as games, or operating systems which Thrustmaster has indicated are not supported, are not accepted at this space either.

**THRUSTMASTER®**

TECHNICAL SUPPORT

<http://ts.thrustmaster.com>





### **Consumer warranty information**

Worldwide, Guillemot Corporation S.A., whose registered office is located at Place du Granier, B.P. 97143, 35571 Chantepie, France (hereinafter "Guillemot") warrants to the consumer that this Thrustmaster product shall be free from defects in materials and workmanship, for a warranty period which corresponds to the time limit to bring an action for conformity with respect to this product. In the countries of the European Union, this corresponds to a period of two (2) years from delivery of the Thrustmaster product. In other countries, the warranty period corresponds to the time limit to bring an action for conformity with respect to the Thrustmaster product according to applicable laws of the country in which the consumer was domiciled on the date of purchase of the Thrustmaster product (if no such action exists in the corresponding country, then the warranty period shall be one (1) year from the original date of purchase of the Thrustmaster product).

Notwithstanding the above, rechargeable batteries are covered by a warranty period of six (6) months from the date of original purchase.

Should the product appear to be defective during the warranty period, immediately contact Technical Support, who will indicate the procedure to follow. If the defect is confirmed, the product must be returned to its place of purchase (or any other location indicated by Technical Support).

Within the context of this warranty, the consumer's defective product shall, at Technical Support's option, be either replaced or returned to working order. If, during the warranty period, the Thrustmaster product is subject to such reconditioning, any period of at least seven (7) days during which the product is out of use shall be added to the remaining warranty period (this period runs from the date of the consumer's request for intervention or from the date on which the product in question is made available for reconditioning, if the date on which the product is made available for reconditioning is subsequent to the date of the request for intervention). If permitted under applicable law, the full liability of Guillemot and its subsidiaries (including for consequential damages) is limited to the return to working order or the replacement of the Thrustmaster product. If permitted under applicable law, Guillemot disclaims all warranties of merchantability or fitness for a particular purpose.

This warranty shall not apply: (1) if the product has been modified, opened, altered, or has suffered damage as a result of inappropriate or abusive use, negligence, an accident, normal wear, or any other cause unrelated to a material or manufacturing defect (including, but not limited to, combining the Thrustmaster product with any unsuitable element, including in particular power supplies, rechargeable batteries, chargers, or any other elements not supplied by Guillemot for this product); (2) if the product has been used for any use other than home use, including for professional or commercial purposes (game rooms, training, competitions, for example); (3) in the event of failure to comply with the instructions provided by Technical Support; (4) to software, said software being subject to a specific warranty; (5) to consumables (elements to be replaced over the product's lifespan: disposable batteries, audio headset or headphone ear pads, for example); (6) to accessories (cables, cases, pouches, bags, wrist-straps, for example); (7) if the product was sold at public auction.

This warranty is nontransferable.

The consumer's legal rights with respect to laws applicable to the sale of consumer goods in his or her country are not affected by this warranty.

### **Additional warranty provisions**

During the warranty period, Guillemot shall not provide, in principle, any spare parts, as Technical Support is the only party authorized to open and/or recondition any Thrustmaster product (with the exception of any reconditioning procedures which Technical Support may request that the consumer carry out, by way of written instructions – for example, due to the simplicity and the lack of confidentiality of the reconditioning process – and by providing the consumer with the required spare part(s), where applicable).

Given its innovation cycles and in order to protect its know-how and trade secrets, Guillemot shall not provide, in principle, any reconditioning notification or spare parts for any Thrustmaster product whose warranty period has expired.

In the United States of America and in Canada, this warranty is limited to the product's internal mechanism and external housing. In no event shall Guillemot or its affiliates be held liable to any third party for any consequential or incidental damages resulting from the breach of any express or implied warranties. Some States/Provinces do not allow limitation on how long an implied warranty lasts or exclusion or limitation of liability for consequential or incidental damages, so the above limitations or exclusions may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from State to State or Province to Province.

### **Liability**

If permitted under applicable law, Guillemot Corporation S.A. (hereinafter "Guillemot") and its subsidiaries disclaim all liability for any damages caused by one or more of the following: (1) the product has been modified, opened or altered; (2) failure to comply with assembly instructions; (3) inappropriate or abusive use, negligence, an accident (an impact, for example); (4) normal wear; (5) the use of the product for any use other than home use, including for professional or commercial purposes (game rooms, training, competitions, for example). If permitted under applicable law, Guillemot and its subsidiaries disclaim all liability for any damages unrelated to a material or manufacturing defect with respect to the product (including, but not limited to, any damages caused directly or indirectly by any software, or by combining the Thrustmaster product with any unsuitable element, including in particular power supplies, rechargeable batteries, chargers, or any other elements not supplied by Guillemot for this product).



## End User License Agreement

**IMPORTANT:** Please read the following End User License Agreement before using the device. The term "Software" refers to all executable programs, managers, libraries, data files and to any documentation relating to the programs, as well as the complete operating system included in the product. The Software is licensed, and not sold, to the User, exclusively for use complying with the terms of this License Agreement. You hereby accept and agree to abide by the Terms & Conditions of this License Agreement. If you disagree with the Terms & Conditions of this License Agreement, please do not use the Software.

The Software is protected by international copyright laws and agreements, as well as other international laws and agreements relating to intellectual property.

The Software (excluding the software sub-applications) remains the property of Guillemot Corporation S.A. All rights reserved. Guillemot Corporation S.A. grants only a limited and non-exclusive right to use the Software.

**Guillemot Corporation S.A. reserves the right to cancel this License Agreement in the event of failure to abide by its Terms & Conditions.**

### License granted:

1. The license is granted to the original Buyer alone. Guillemot Corporation S.A. remains the sole owner and holder of the Software (excluding the software sub-applications), and reserves all rights that are not expressly granted by this License Agreement. The User is not allowed to sub-license the rights granted by this License Agreement. The User is allowed to transfer this License, provided that the original Buyer retains no part of the Software and that the Transferee reads and accepts the Terms & Conditions of this License Agreement.
2. The Buyer may only use the software on one computer at a time. The machine-readable part of the Software may be copied to another computer, provided that the Software is deleted from the first computer and that it is impossible to use the Software on several computers at the same time.
3. The Buyer hereby acknowledges and accepts the copyright belonging to Guillemot Corporation S.A. The copyright shall in no event be removed from the Software, nor from any documentation, whether printed or electronic, provided with the Software.
4. The License grants the User the right to perform one (1) copy of the machine-readable part of the Software for archival purposes, provided that the User also copies Software's copyright.
5. Except within the limits expressly allowed by this License Agreement, the Buyer may not agree or allow any third party to agree to: grant a sublicense; provide or divulge the Software to other third parties; allow use of the Software on several computers at a time; perform alterations or copies of any kind; reverse engineer, decompile or modify the Software in any way or attempt to obtain information relating to the Software that is not accessible to the User; perform copies or translations of the User Manual.

### Warranty limitation:

The Software is provided "as is", with no guarantee whatsoever from Guillemot Corporation S.A. regarding its use and/or performance. Guillemot Corporation S.A. does not guarantee that the operation of the Software will be free from interruptions or errors. The use or the performance of the Software remain under the Buyer's entire responsibility, and Guillemot Corporation S.A. can provide no guarantee of any kind with regard to the performance and results obtained by the Buyer while using the Software. No guarantee of any kind, whether explicit or implied, is offered by Guillemot Corporation S.A. with regard to the non-violation of third party rights, the merchantability or the adequacy of the Software to a specific use.

In no event shall Guillemot Corporation S.A.'s liability be engaged in the event of damages of any kind arising from the use or the incapacity to use the Software.

### Applicable law:

The Terms & Conditions of this License Agreement are subject to French Law.

## COPYRIGHT

© 2016 Guillemot Corporation S.A. All rights reserved. Thrustmaster® is a registered trademark of Guillemot Corporation S.A. T.A.R.G.E.T is a trademark of Guillemot Corporation S.A. All other trademarks and brand names are hereby acknowledged and are property of their respective owners. Illustrations not binding. Contents, designs and specifications are subject to change without notice and may vary from one country to another. Made in China.

## ENVIRONMENTAL PROTECTION RECOMMENDATION



At the end of its working life, this product should not be disposed of with standard household waste, but rather dropped off at a collection point for the disposal of Waste Electrical and Electronic Equipment (WEEE) for recycling.

This is confirmed by the symbol found on the product, user manual or packaging.

Depending on their characteristics, the materials may be recycled. Through recycling and other forms of processing Waste Electrical and Electronic Equipment, you can make a significant contribution towards helping to protect the environment.

Please contact your local authorities for information on the collection point nearest you.

[www.thrustmaster.com](http://www.thrustmaster.com)

