BMS TRAINING SCRIPTS

Miguel "Revientor" Victores Franco

BenchMarkSims - Escuadron111

12/06/2010

This document tries to describe how to create basics training scripts.

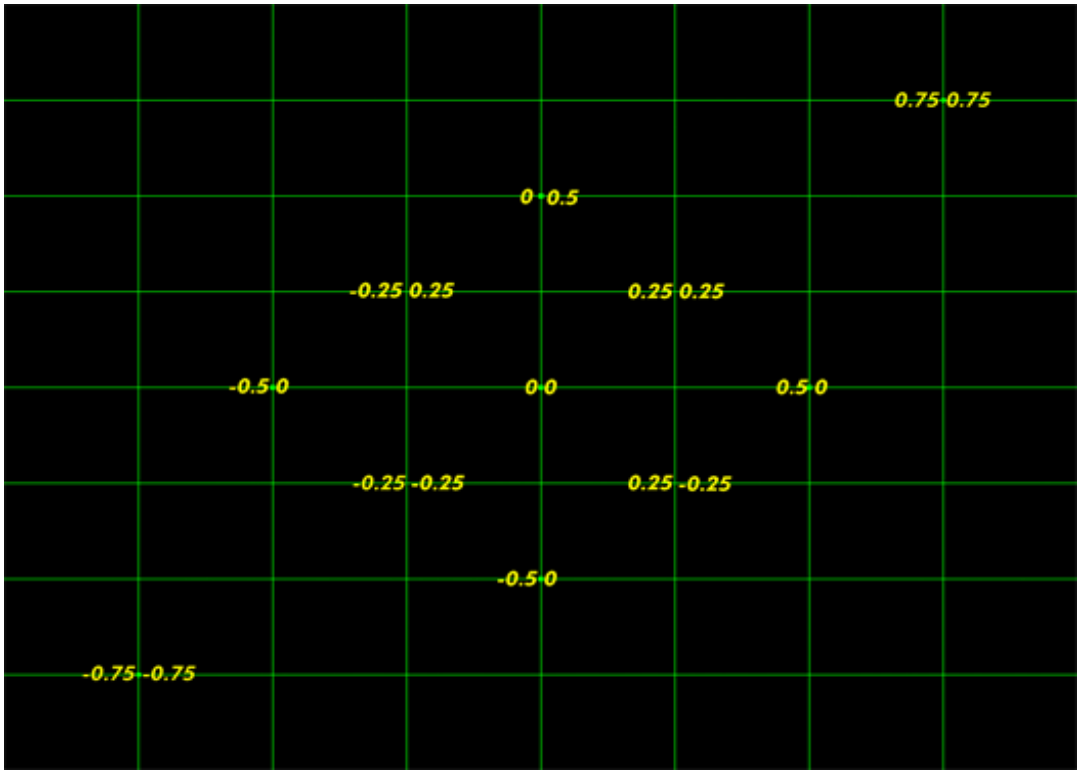
First of all I have to say "*sorry about my English*" ☺

The most of the info here present has been looking in the "*F4-BMS 2.0 Manual.pdf*" page 151.

With BMS5 you can see the most of the training has training scripts, this ones are saved in the folder "*…\campaign\SAVE*" in txt format.

Those script are very simple to create, you have to know some things before starting your own scripts.

- **First**: The screen is divided in coordinates. The left-up part of the screen is -1.0 1.0 , the left-down is -1.0 -1.0 , right-up 1.0 1.0 , right-down 1.0 -1.0 and the center of the screen is 0.0 0.0. You can use a picture like this to know where the text or actions that you design will occur. The first number is the left-right coordinate, and the second one is the up-down coordinate. This coordinates are the same in each resolution of the screen (1024*768, 800*600…)



- **Second**: Always that you want to type or draw something on the screen, you will have to move your cursor where you want to create the event.

- **Third:** The language is designed to never block the main Falcon process. This is accomplished by only running one command in the language per frame. One effect of this is that bigger scripts do NOT slow the simulation down. This is because, regardless of the size, only a single command executes per frame.

- **Fourth**: Mission with scripts enabled start out in PAUSED mode

- **Fifth:** The script language has no concept of variables. All functions take actual values. For the purpose of interactive training, this lack of variables in the language should not be a significant limitation.
- **Sixth**: The code in a script can be divided into "sections" These sections are defined by simply putting the name of the section on a line by itself.

- **Seventh**: The action of the language is composed entirely of functions. There are no operators. So, each line of a script can contain one of the following things:
  - A function name followed by arguments
  - A section name
  - A comment starting with //
  - A blank line
- **Eighth**: The language has a unique and very limited form of return values. Specifically, most functions when they run have a return value of true. Certain functions, however, return false under certain circumstances (usually when they time out). The only way that return values can be used is by the use of them if and while type functions. These functions simply check the return value - either true or false - of the last executed function to determine whether to execute. This is ONLY use of return values.
- **Ninth**: Mission with scripts enabled have the AP mode set to STEERPOINT in order to enable the script writer to guide the player´s aircraft. The autopilot still has to be enabled manually by the script writer.

Now it is time to know how works the commands, you must write the commands as are here, with the capital letters in the correct place, this is too important.

Print <time> <string>: This will type on the window. The time (*1.0 is 1 second. 0.001 is 1 millisecond*) section is the time that the massage will be showed on screen. Example:

```
Print 20 "THANKS FOR USING BMS TRAINING SCRIPTS"
```

WaitPrint <time> <string>: This makes that the script type a message and waits to the time to show the message finish and then the script will continue with the next command.

```
WaitPrint 40 "Revientor Trainer V1.0 "
```

WaitInput <time> <command/callback>: The script will wait for the user pres a specific key. The time is time that the script will wait, if it expires will give a “false” response and will continue with the script. You can see the name of the commands in the key file.

```
WaitInput 111 SimTrigger
```

Wait <time>: This make a pause during the specific time, and then the script will continue.

```
Wait 60
```

WaitMouse <time> <float (target x)> <float (target y)> <float (distance)>: The script will wait to “see” the cursor near the position selected. The float distance is the distance where the script will recognize the cursor.

```
WaitMouse 40 -0.34 0.75 0.05
```

WaitSoundStop <time>: It waits for a sound ends during some time, if the time ends before the sound, the script will continue anyway giving a “false” response.

```
WaitSoundStop 60
```

WaitSound <string>: This will play a specific sound, and will stop the script during the sound is playing.

```
WaitSound campaign\save\13A-ARadar\AARadar33.wav
```

Sound <string>: The same of WaitSound but this one will not stop the script.

```
Sound campaign\save\13A-ARadar\AGRadar33.wav
```

EndSection: This function can be used to end a section. If there is any previous function on the call stack (generally created by using the "call" function), then execution will return to the instruction immediately after the "call" when "EndSection" is encountered. If there is nothing on the call stack, this line is ignored.

```
EndSection
```

EndScript: Immediately ends the script.

```
EndScript
```

Block <command/callback>: Blocks the system from recognizing any commands or callbacks listed as arguments. If you don´t list callbacks, it will block everything.

```
Block
```

Allow <command/callback>: Allows the system from recognizing any commands or callbacks listed as arguments. If you don´t list callbacks, it will allow everything.

```
Allow SimTrigger
```

If: If the return value of the last command is true, then execution moves to the next instruction. If the return value is false, then the next instruction is skipped.

```
If
```

IfNot: Functions identically to "If", except the next instruction is executed if the return value of the last statement is false.

```
IfNot
```

Jump <section>: Immediately jumps to the section identified by <section>. NOTE this does not add anything to the call stack.

```
Jump Start
```

Call <section>: Immediately jumps to the section identified by <section>. The current location is added to the call stack, and execution will return to this location when an "EndSection" FUNCTION IS ENCOUNTERED.

```
Call Start
```

SetCursor <float (x)> <float (y)>: Sets the cursor position to the location specified in <float (x)> and <float (y)>. The range of the coordinate system is from -1.0 to 1.0. For example x = -1.0 is the left of the screen, 0.0 is the center and 1.0 is the right side of the screen. This cursor location is used for all print and drawing functions

```
SetCursor 0.0 0.95
```

SetColor <hex>: Sets the current cursor color to the color specified by <hex>. This cursor color is used for all print and drawing functions.

```
SetColor 0xffffffff
```

SetFont <hex>: Sets the current font to the font specified by <integer>. Examples of valid fonts are 1, 2 and 3. This font is used for all print functions.

SetFont 1

Oval <time> <float (x)> <float (y) optional>: Draws an oval on the screen for <time> duration. The size of the oval is specified by the two arguments. If only one argument is supplied, then a circle is drawn with that radius.

Oval 30 0.05

Line <time> <float (x1)> <float (y1)> <float (x2)> <float (y2)>: Draws a line for <time> duration from x1, y1 to x2, y2

Line 40 0.55 -0.10 0.75 -0.50

EnterCritical: This experimental function can be used to execute multiple commands in a single frame. Execution will proceed in one frame until an "EndCritical" function is encountered. This command may be very dangerous, as the use of ANY waiting calls like "WaitPrint" will cause Falcon to stop running until it is finished.

EnterCritical

EndCritical: Causes one command per frame execution to resume.

EndCritical

SimCommand <command/callback>: Causes <command> to be executed just as if the user had pressed the keystroke.

SimCommand SimTogglePaused

While: If the prior return value is true, runs the commands until the next "EndSection". When "EndSection" is found, execution returns to the command prior to the "While" function. If the prior return value is false, then execution moves to the command after the next "EndSection"

While

WhileNot: Operates identically to the "While" statement, with the exception that a return value of false causes the while to execute.

WhileNot

Call <section>: If the return value of the prior statement is true, then the section identified by <section> will be called. Once an "EndSection" is encountered, execution will return to the instruction after "Call"

Call Ramp

CallNot <section>: Functions identically to "Call" except that a false return value triggers the call.

CallNot Ramp

Clear: Immediately removes all drawn elements from the screen (Such as those created by "Print" "Line" and "Oval" statements)

Clear

ClearLast <Integer - optional>: If no argument is provided, the last created drawn element will be removed from the screen.

ClearLast

SetFlash <Hex>: Sets the rate of flashing that should occur for drawing calls. "SetFlash 0" disables flashing.

SetFlash 0x100

SetCursorCallback <Callback>: Sets the current cursor position to the x and y position of the button specified by <callback>. If the <callback> is not found on the current panel, the location of the cursor will remain unchanged.

SetCursorCallback SimWarnReset

SetCursorDial <Callback>: Sets the current cursor position to the x and y position of the dial specified by <callback>. If the <callback> is not found on the current panel, the location of the cursor will remain unchanged.

I don't know how do an example of this one ☹

WaitCallbackVisible <time> <command/callback>: Pauses execution of the script until <time> expires, or until the button specified by <callback> is found on the current panel. If the timer expires, then the function returns false.

WaitCallbackVisible 30 SimWarnReset

WaitDialVisible <time> <command/callback>: Pauses execution of the script until <time> expires, or until the dial specified by <callback> is found on the current panel. If the timer expires, then the function returns false.

I don't know how do an example of this one ☹

SetTextBoxed <Interg>: Sets the type of text boxing that should apply to print statements. Valid values are 0, 1, 2 and 3. 0 = no text boxing, 1 = text is boxed by lines, 2 = black text is drawn with a colored box, 3 = colored text is drawn with a black box. This function affects all print functions. You have to rewrite these lines when you use Jump command.

SetTextBoxed 2

MoveCursor <float (x)> <float (y)>: Moves the cursor from its current position by x and y amount. The range of the coordinate system is from -1.0 to 1.0. For example x = -1.0 is the left of the screen, 0.0 is the center and 1.0 is the right side of the screen. This cursor location is used for all print and drawing functions.

MoveCursor 0.0 0.05

SetTextOrientation <Interg>: Sets the orientation of the text relative to the cursor position. Acceptable values are 0, 1 and 2. 0 = left justified text, 1 = centered text, 2 = right justified text. This function affects all print functions.

SetTextOrientation 2

SetViewCallback <callback>: Searches the panels for a button using <callback>, and then sets the current view to that panel. If the callback is not found, the view is left unchanged. This can be used to write scripts that are compatible with multiple different 2d cockpits.

SetViewCallback SimWarnReset

SetViewDial <callback>: Searches the panels for a dial using <callback>, and then sets the current view to that panel. If the callback is not found, the view is left unchanged. This can be used to write scripts that are compatible with multiple different 2d cockpits.

I don't know how do an example of this one ☹

SetPanTilt <float (x)> <float (y)>: Sets the current 3d view to the coordinates specified by <float (x)> and <float (y)>. These coordinates are in radians, and 0.0 is straight ahead in the 3d cockpit. Follow the next table to move the pilot`s head:

Degrees	Radians
22º	0.392699081698724154807830422909
45º	0.785398163397448309615660845819
90º	1.570796326794896619231321691639
180º	3.141592653589793538462643383279
360º	6.283185307179586476925286766559
Positive numbers move the pilot`s heads to the left or down and negative ones to the right or up.	

SetPanTilt -0.40 1.13

MovePanTilt <float (x)> <float (y)>: Offsets the current 3d view from its current position by the amounts specified by <float (x)> and <float (y)>. These coordinates are in radians.

MovePanTilt 0.03 -0.12

SetCursor3D <float (x)> <float (y)> <float (z)>: Takes a camera centric point defined by the x, y and z coordinate, converts that into 2d screen space, and sets the cursor to that location.

SetCursor3D 0.3 -0.22 0.50

Now we are going to see an example of a small script:

```
////////////////////////////////////////////////////////////////////////////////
// Revientor Trainer V1.0
////////////////////////////////////////////////////////////////////////////////
Jump MainLoop
MainLoop
SimCommand SimSafeMasterArm
Block
////////////////////////////////////////////////////////////////////////////////
// Display Start Info
////////////////////////////////////////////////////////////////////////////////
SetColor 0xffffffff
SetTextBoxed 2
SetTextOrientation 1
Print 20 "THANKS FOR USING BMS TRAINING SCRIPTS"
Wait 4
Allow
Clear
////////////////////////////////////////////////////////////////////////////////
// Instructions
////////////////////////////////////////////////////////////////////////////////
SetCursor 0.0 0.92
Print 40 "Today you can do everything that you want with your plane"
MoveCursor 0.0 -0.05
Print 40 "I am going set your Comm 1 to the frequency 232.400 (Pusan Airbase ATC)"
MoveCursor 0.0 -0.05
Print 40 "I do this by pushing the COM 1 Button in the ICP and then by typing the"
MoveCursor 0.0 -0.05
Print 40 "frequency with the ICP numbers, finally push enter in the ICP."
Wait 20
Clear
EndSection
Jump actions
```

```
////////////////////////////////////////////////////////////////////////////
actions
SimCommand SimTogglePaused
SimCommand SimICPCom1
SimCommand SimICPALOW
SimCommand SimICPTHREE
SimCommand SimICPALOW
SimCommand SimICPStpt
SimCommand SimICPZERO
SimCommand SimICPEnter
EndSection
////////////////////////////////////////////////////////////////////////////
EndScript
```

You can do different configurations on the script presentation playing with the commands, but the basic idea always will be the same, show info on screen or sounds.

Let us with the description of this script:

```
////////////////////////////////////////////////////////////////////////////
// Revientor Trainer V1.0
////////////////////////////////////////////////////////////////////////////
Jump MainLoop                    The script look for the MainLoop section
MainLoop                         Name of a section
SimCommand SimSafeMasterArm      Script selects the weapons safe mode
Block                            Script blocks all command to the user
////////////////////////////////////////////////////////////////////////////
// Display Start Info
////////////////////////////////////////////////////////////////////////////
SetColor 0xffffffff              Giving color to the text
SetTextBoxed 2                   Selecting the type of box
SetTextOrientation 1             Selection how the text will be placed
Print 20 "THANKS FOR USING BMS TRAINING SCRIPTS"   Shows this text on the screen
Wait 4                           Stops the script during 4 seconds
Allow                            Script unblock all command, now the user can use them
Clear                            Cleans the screen
////////////////////////////////////////////////////////////////////////////
// Instructions
////////////////////////////////////////////////////////////////////////////
SetCursor 0.0 0.92               Move the cursor to the middle-up part of the screen
Print 40 "Today you can do everything that you want with your plane"   Shows this text on the screen
MoveCursor 0.0 -0.05             Move down the cursor from its last position
Print 40 "I am going set your Comm 1 to the frequency 232.400"   Shows this text on the screen
MoveCursor 0.0 -0.05             Move down the cursor from its last position
Print 40 "I do this by pushing the COM 1 Button in the ICP and then by typing the"   Shows this text on the screen
MoveCursor 0.0 -0.05             Move down the cursor from its last position
Print 40 "frequency with the ICP numbers, finally push enter in the ICP."   Shows this text on the screen
Wait 20                          Stops the script during 20 seconds
Clear                            Cleans the screen
EndSection                       The last section (MainJump) ends here
Jump actions                     The script look for the actions section
////////////////////////////////////////////////////////////////////////////
actions                          Name of a section
SimCommand SimTogglePaused       Script quit the pause mode
SimCommand SimICPCom1            Script selects Com1 button
SimCommand SimICPALOW            Script push in the ICP the number 2
SimCommand SimICPTHREE           Script push in the ICP the number 3
SimCommand SimICPALOW            Script push in the ICP the number 2
SimCommand SimICPStpt            Script push in the ICP the number 4
SimCommand SimICPZERO            Script push in the ICP the number 0
SimCommand SimICPEnter           Script push in the ICP the enter button
EndSection                       The last section (actions) ends here
////////////////////////////////////////////////////////////////////////////
EndScript                        The script ends here
```

That is a basic script, but you can do script a little more complex using sounds (wav format).