The FHQ Debug Console

Introduction

I used other debugging aids in the past, but some of them missed featured I wanted, and others were too intrusive, defining objects on their own that would appear in the editor which I wanted to avoid.

So I decided to write my own tool for debugging purposes, with a set of goals in mind:

- It should not be intrusive, i.e. not define classes on it's own, not hook into the addAction menu, etc.
- It should be able to give me a few "live" debugging aids, like the possibility to monitor values while the console is closed.
- It should offer a few tools that are useful to have, like finding house positions etc.
- I liked the console from TakeOn: Helicopters, so it's design was used as a basis for this project.

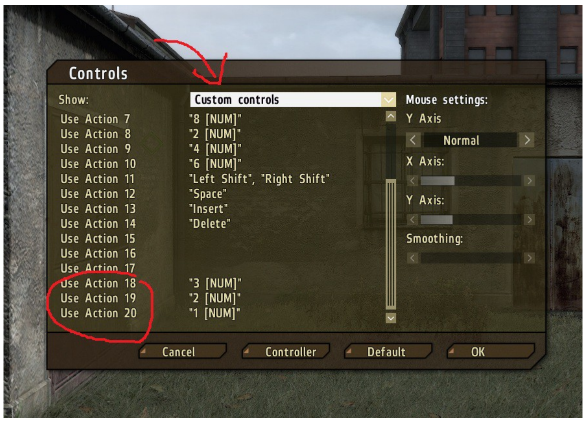
New Features

The new version 1.2 now supports the following new features:

- Saving/Restoring of values entered in the main console window
- New tools: Position/Direction init field generation, spawning of ammo crates and backpacks, and storing the current loadout in a script ready to be executed on a unit.

Installation

Unpack the mod-folder into a suitable place, and add @fhq_debug to your list of modules. The console currently uses two keys, defined in Options->Controls under the Custom Control header. The "Use Action 20" key will open the console, and the "Use Action 19" key will be used to select objects in the world for some tools (currently the house position tool). Note that currently, these keys are not checked for



Controls

Show: Custom controls

Use Action 7	"8 [NUM]"
Use Action 8	"2 [NUM]"
Use Action 9	"4 [NUM]"
Use Action 10	"6 [NUM]"
Use Action 11	"Left Shift", "Right Shift"
Use Action 12	"Space"
Use Action 13	"Insert"
Use Action 14	"Delete"
Use Action 15	
Use Action 16	
Use Action 17	
Use Action 18	"3 [NUM]"
Use Action 19	"2 [NUM]"
Use Action 20	"1 [NUM]"

Mouse settings:
Y Axis: Normal
X Axis:
Y Axis:
Smoothing:

Illustration 1: Key configuration

qualifiers, so "Page up" would be the same as CTRL+"Page Up". I use the number pad keys 1 and 2 since I do not use the view direction keys.

Usage: The main window

When in game, the console can be opened with the "Use Action 20" key. It will pop up a rather large window.

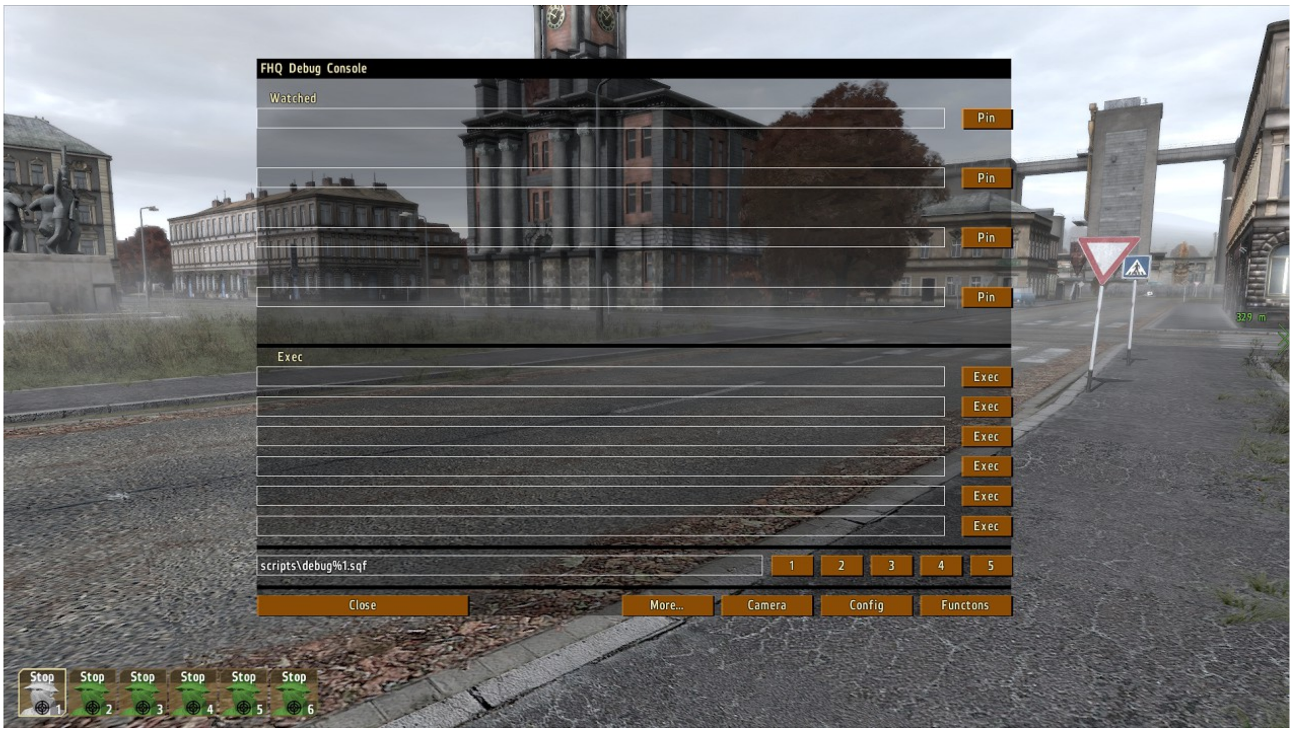


Illustration 2: Main console window

The upper half ("Watched") contains four text fields. Any code you enter here is constantly executed and the result shown below the text field. Each of the text fields has a button next to it, labelled "pin". If clicked, the result of the corresponding text field is continuously displayed even after the main console is closed. This can be used to monitor values throughout testing.
The lower half ("Exec") is built out of six text fields. Each field can contain code which can be executed by clicking the "Exec" button next to it. No result is displayed.
Below that area, you will find another text field and five buttons. The text field is pre-set to the value "`scripts\debug%1.sqf`". Clicking any button will execVM the appropriate script, so clicking "3" would execute "`scripts\debug3.sqf`". The template can be changed to suit your needs.
The buttons on the lower edge have different functions. Each one of them closes the main window when clicked, with different additional functions being executed. "Close" simply closes the console. "More..." opens another window with additional tools (more on that later). "Camera" executes the standard "`camera.sqs`" script (i.e. creating a controllable camera you can fly around with). "Config" opens the default BIS config browser and "Functions" opens the BIS function help dialog. Both require a function module in the current mission.
The values entered are stored across multiple invocations of the console, but are not permanently stored, so they will be gone when restarting a mission or restarting the game.

As has been said, "More..." opens another window with a number of additional tools. Each tool can be executed by clicking. The following sections describe the tools in more detail.

Usage: Show House Positions tool

Clicking the "Show House Positions" tool will at first close the window. You should see a red sphere somewhere in the middle of your screen. This serves as a cursor to select a house in the vicinity of your player. A house between you and the red sphere can be selected with the "Use Action 19" key. While looking around, a red arrow will appear on top of all houses that you look at.

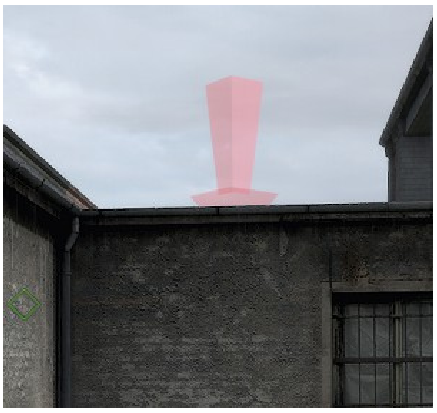


Illustration 3: Selecting a building

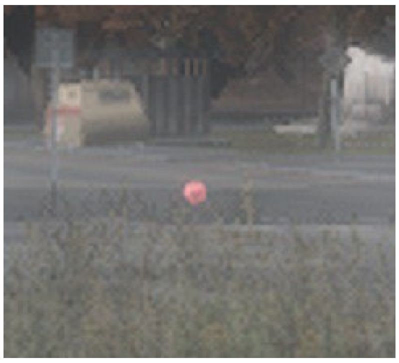
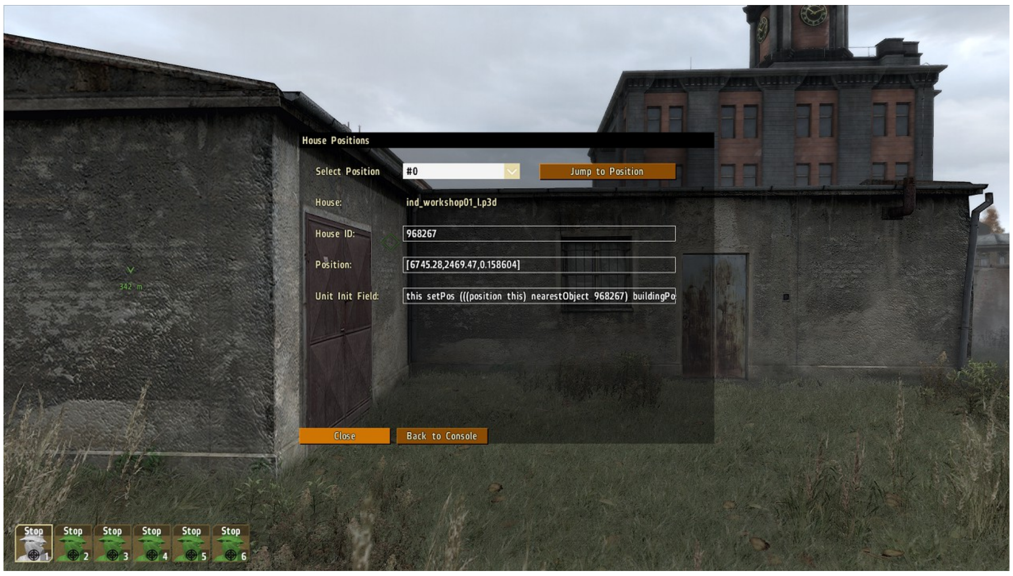


Illustration 4: Targeting

This will either say that the house has no positions, or it will open another window. The new window displays all available house positions in a drop box. You can select any of the positions by selecting them from the drop box. The "House" line displays the name of the model, "House ID" shows the map ID of the selected house. "Position" displays the coordinates of the house position in ATL (i.e setPosATL compatible). Finally, the "Unit Init Field" can by copied/pasted into a unit's init field, making the unit appear at the selected house position.


House Positions

Select Position: #0

House: ind_workshop01_l.p3d

House ID: 968267

Position: [6745.28,2469.47,0.158604]

Unit Init Field: this setPos (((position this) nearestObject 968267) buildingPo

Illustration 5: The House Position dialog

To test out the house position, clicking on the "Jump to Position" button will jump your current player to this position. You may close the window if you want to look around. If you need to continue scanning for suitable house positions, press "Use Action 19" again to return to the Show House Positions tool.

Usage: Toggle Group Icons

For mission makers, it is sometimes interesting to know what exactly is where on the map. The Toggle Group Icons tool toggles between showing and hiding icons that show the position, composition, strength and affiliation of all groups on the map. The icons are normal NATO symbols, with their appropriate colors (Blue for BLUFOR, red for OPFOR, green for Independent, and yellow for Civilian). Hovering the mouse pointer over an icon (either on the map, or the 3D view) will display additional information, like Group ID, side and faction, relative strength (i.e. health of all units), the composition of the group (infantry and vehicles), their combat mode, and behaviour.

Note that this function might case problems with High Command mode icons.

Usage: Toggle Debug Output

As a scripter, you are often in need of debug output inside your missions. This can be achieved with functions like "`diag_log`", "`hint`" or "`sideChat`". However, these all have their drawbacks. For example, `diag_log` has to be checked in the RPT file, hints can hide other hints, etc.

The FHQ debug console offers a "quake like" window that shows a few lines of text. The console offers a few functions that can be used for debug output (see below). Clicking the tool button will simply toggle display of this feature on and off. Output that is made while the window is not being displayed is still being buffered and show once the window is toggled to "on" state.

The debug console offers three functions for the user.

Code	Usage
`call FHQ_DebugConsole_Clear;`	Clear the console window.
`"string" call FHQ_DebugConsole_DebugOut;`	Output "string" to the console.
`"text" call FHQ_DebugConsole_SetOutput;`	Set the console window to show "text".

Note that the window can display structured text. The DebugOut function simply queues single lines, while the SetOutput function can be used for more complex monitoring when full control of the display area is used.

To allow the functions to be no-ops while the console is not loaded, the file "`fhq_debugconsole_support.sqf`" can be executed in your init code. It will automatically create empty dummy functions that do nothing if the console is not loaded, allowing you to test the code without the debug console loaded without running into problems. "`fhq_debugconsole_support.sqf`" checks whether the console is loaded or not, and only overrides the functions if the console is not found.

Usage: Position/Orientation Init

This simple tool will take the orientation and position of the current player and create an init string suitable for putting into a unit's init field. This is useful for positioning individual units by just positioning yourself somewhere and invoking the tool. The init string is dumped into the clipboard. Note that this also works for vehicles, so positioning a vehicle and invoking the tool while being inside the vehicle will use the vehicle's position.

Usage: Spawn Weapon Crates

This tool is capable of spawning weapon crates and backpacks. The combo box at the top of the tool allows you to filter for weapon crates, respectively. Anything spawned will appear in front of you.

Usage: Save Current Loadout

This tool simply writes a script to the clipboard that, when called for a unit, replicates the exact loadout that your player character was carrying at the time the tool was executed (weapons, ammo, items, backpack and backpack content). This can be used as a simple loadout editor for units when creating missions: Simply spawn backpacks and weapon crates via the Spawn Weapon Crates tool, take from the boxes what you want the respective unit to have, and save the loadout. The script will be rather large and will not fit in an init field (it will use local variables), so it is best pasted into a script file. Near the top of the script, you will find the line “`_unit = player;`”. Replace “`player`” with the unit you want to assign to (a valid unit name, or another variable containing a reference to a valid object).

Here's an example generated via the tool:

```
private ["_unit", "_backpack"];
/* Replace player below with the unit to assign the loadout to */
_unit = player;

if (!isNull _unit) then {
    removeAllWeapons _unit;
    removeAllItems _unit;
    removeBackpack _unit;

    /* Magazines */
    _unit addMagazine "30Rnd_762x39_SA58";
    _unit addMagazine "30Rnd_762x39_SA58";
    _unit addMagazine "30Rnd_762x39_SA58";
    _unit addMagazine "30Rnd_762x39_SA58";
    _unit addMagazine "30Rnd_762x39_SA58";
    _unit addMagazine "30Rnd_762x39_SA58";
    _unit addMagazine "30Rnd_762x39_SA58";
    _unit addMagazine "30Rnd_762x39_SA58";
    _unit addMagazine "HandGrenade_West";
    _unit addMagazine "HandGrenade_West";
    _unit addMagazine "HandGrenade_West";
    _unit addMagazine "HandGrenade_West";

    /* Weapons */
    _unit addWeapon "Sa58V_CCO_EP1";
    _unit addWeapon "NVGoggles";
    _unit addWeapon "ItemMap";
    _unit addWeapon "ItemCompass";
    _unit addWeapon "ItemWatch";
    _unit addWeapon "ItemRadio";

    /* Backpack */
    _unit addBackpack "CZ_Backpack_Ammo_EP1";
    _backpack = unitBackPack _unit;
    clearMagazineCargo _backpack;
    clearWeaponCargo _backpack;

    _backpack addMagazineCargo ["30Rnd_762x39_SA58", 6];
    _backpack addMagazineCargo ["PipeBomb", 1];
    _backpack addMagazineCargo ["HandGrenade_West", 3];
    _backpack addMagazineCargo ["SmokeShell", 2];
    _backpack addMagazineCargo ["SmokeShellRed", 1];
};
```

Known Problems

Currently, the console is no longer available when a mission has been saved and is subsequently reloaded. The reason is simply that the display is not available any more after loading. The reason for this is unknown; if you have any ideas, I've posted something on The BIS forum. I'm thankful for any pointers.

Future

I want to release a script only version of the console. The console works in both ArmA 2 and Iron Front, but of course, the latter does not allow for addons. A script-only version will be available for this purpose.

I also intend to add more tools to the console. If you have any idea for a useful tool, let me know.

Credits

The FHQ DebugConsole was written by Varanon, developed with ArmaDev by Alwarren. I can be reached by email at thomas@friedenhq.org or on the BIS forum under the username of Varanon.

If you are a fan of the ArmA series of games, please visit Help Ivan and Martin and show your support. Ivan and Martin have been innocently imprisoned for 100 days now, with no end in sight. Help get them the attention they need to end this travesty.