Michał **BUGAŁA**

HYDROGEN – A DEVELOPMENT ENVIRONMENT FOR THE VIRTUAL BATTLESPACE SYSTEM

Abstract. The article presents the capabilities and technological solutions of the Hydrogen programming environment devised at OBRUM's Simulator Department and intended for the Virtual Battlefield 3 system. Hydrogen is in addition to the tools provided by Bohemia Interactive, the VBS system manufacturer, and facilitates expanding the system with new functionalities.

Keywords: Virtual Battlespace, development environment, programming, scripting language, GUI editor.

1. INTRODUCTION

Many experienced programmers compare programming to art, and it is no wonder that each of these people is distinguished by remarkable aesthetics of the record of their computer programs, commonly called the source code. Only a well and clearly written instruction code can become an incentive to go deeper into the subject of any programming language and a creative inspiration to develop one's own code. The main instrument used by the programmer is the development environment, that is an application that enables editing the source code and facilitates completing the creative process (compilation, assembling, etc.).

This paper presents the capabilities and technological solutions used in the Hydrogen application designed to create scripts and configuration files in the Virtual Battlespace System (VBS) [1] in a faster, more efficient and more elegant manner. VBS is a military solution used by the majority of simulation centres of NATO member states, and is designed to enact various scenarios of the virtual battlefield and to carry out tactical training [2]. Scripts and configuration files serve to expand VBS with new functionalities; they enable incorporating vehicles, weapons and aircraft not yet present in the system.

The Hydrogen development environment helped eliminate the need to use Script Debugger Plugin (Fig. 1) [3], a tool of limited capabilities supplied with the developer version of VBS. The VBS's Script Debugger Plugin has a hint system, but it is limited only to events of the SQF scripting language, and its debugging[1] capabilities are applicable only to SQF scripts that run within a mission[2].

The Hydrogen development environment was created by the author as a response to the lack of similar solutions on the market and to the need for using an effective tool enabling the setting up of a training system for engineering troops by OBRUM's Simulator Department carried out at the request of the Military University of Technology.

[1] Interpretation of source code in order to detect errors [4].

[2] In addition to SQF, the system requires a number of configuration files that cannot be debugged with the Script Debugger Plugin. Debugging of SQF scripts after launching a mission (simulation) requires conducting many additional operations which impede the basic programming work.

Michał **BUGAŁA** – Ośrodek Badawczo-Rozwojowy Urządzeń Mechanicznych "OBRUM" sp z o.o., Gliwice

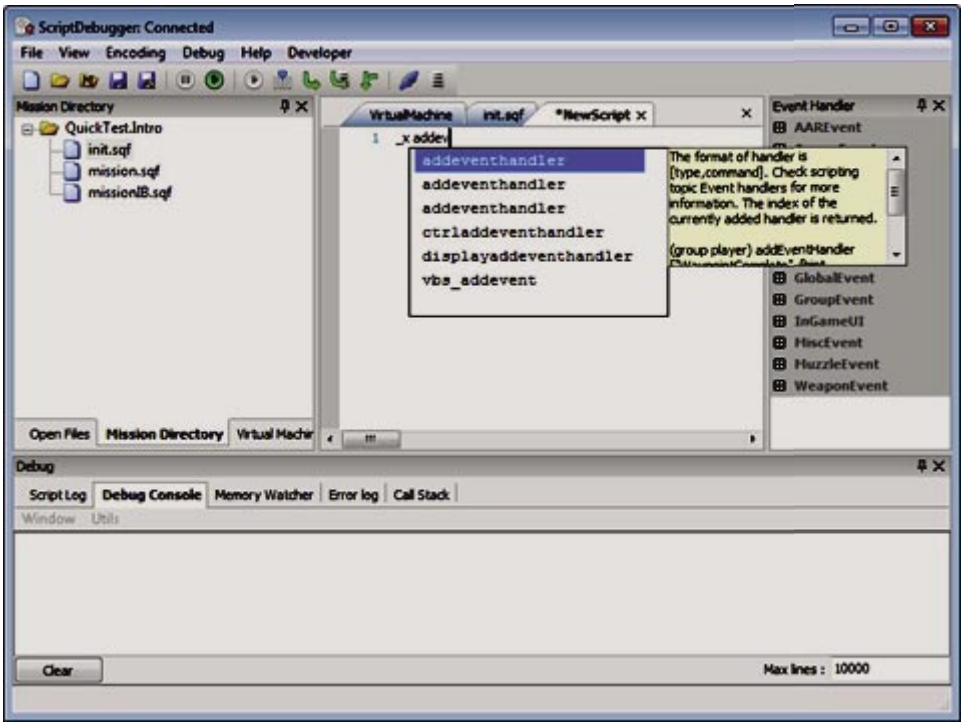

Fig. 1. Main screen of the standard VBS development environment [3]

2. IMPLEMENTATION OF THE HYDROGEN ENVIRONMENT

Hydrogen has been written in a modern Pascal dialect (Free Pascal compiler) [5] using a Lazarus development environment [6]. The compiler and the Lazarus development environment are distributed as free software (open source code), and its architecture became an inspiration to create the Hydrogen development environment. Hydrogen has all the functionalities required of a good text editor or source code editor (cursor position, information on changes in the text, list of recently opened files, etc.). In addition it features a graphic user interface editor, extensive help system and a VBS classes viewing system. A list of the main features of Hydrogen environment tools is included in the Appendix.

2.1. Compilation

Hydrogen is not a compiler. VBS scripts, i.e. programs written in SQF, are compiled using a VBS tool named Addon Packer [7]. Hydrogen strives to facilitate working with VBS scripts. What's more, after preparing appropriate Addon Packer program paths, it is possible to obtain information about errors in the source code after pressing the *Start packing!* button(Fig. 2) when resource packing and error detection is launched - information is displayed in the program console.

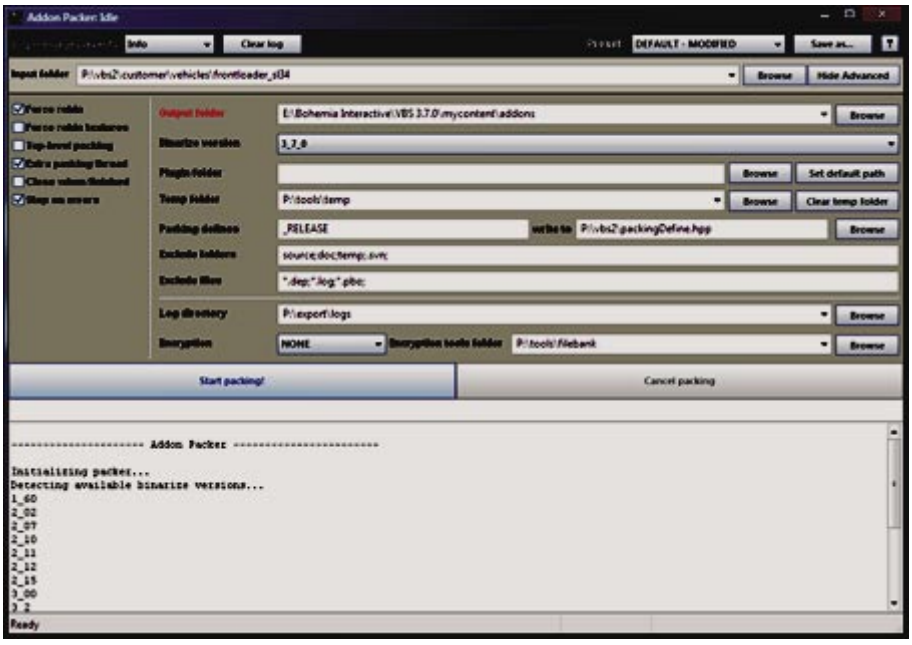

Fig. 2. Main screen of the VBS's Addon Packer application

Unfortunately, when working on new VBS functionalities, in most cases, additional resources are used, e.g. 3D models created with Oxygen [8] (Fig. 3) and, depending on the complexity of the model, compilation and packing can take up to several minutes.

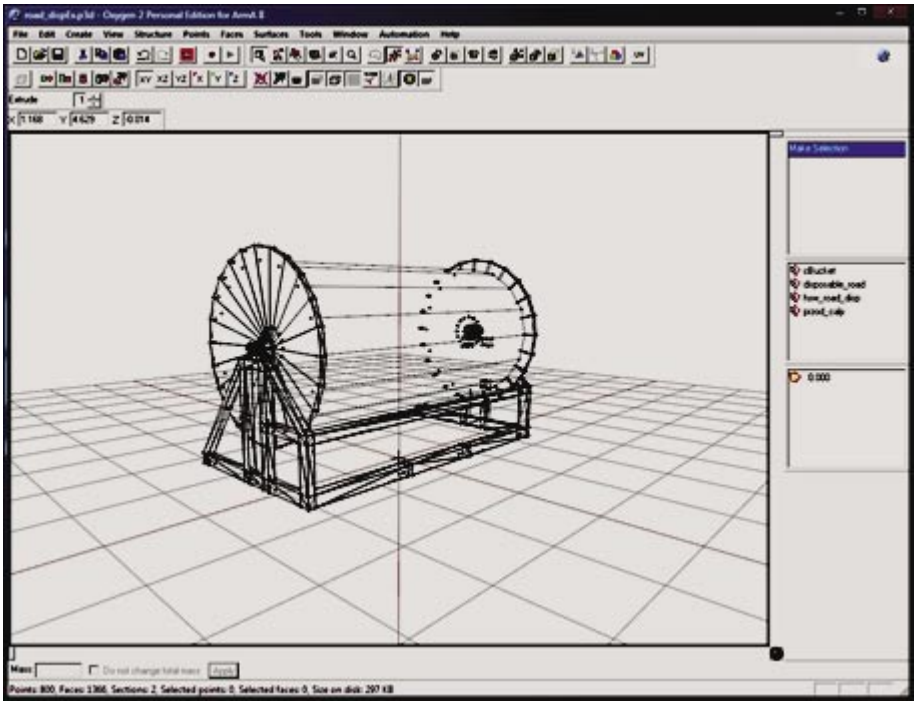

Fig. 3. Main screen of the Oxygen application with an example of a 3D object being edited

2.2. Interpretation, lexication and tokenization

The author attempted to interpret the code of SQF language and configuration files in order to detect errors at the level of the Hydrogen environment. This, however, was effectively prevented by the complexities and lack of access to the VBS system kernel. The VBS scripting

system is in a continuous phase of development, its syntax is unique (not an advantage) and the SQF language itself is not an object-oriented language - it has features of a structured language. All these features have limited the functionality of the Hydrogen environment to lexical analysis[3] and tokenization of SQF scripts and configuration files. As Bohemia Interactive provided information on the categories (tokens) of the SQF language and of configuration scripts [9], it was possible to set up appropriate structures (Tables 1, 2, 3) which became the main source of information for the syntax highlighting system in the Hydrogen environment.

Table 1. Tokens of configuration files and SQL scripts

Token – represented category	Description or examples of lexemes
Key value	class
Identifier	Names of variables introduced by the programmer
Number or string	Numeric and string constants, e.g. 1, 20, 'text'
Comment	Expressions following // or placed between /* and */
Preprocessor command	#define, #ifdef
Editable field[1]	Fields to be filled in marked with an ellipsis (three dots)

Table 2. Additional SQF tokens

Token – represented category	Examples of lexemes
Control command	and, for, or, not
Regular command	setMarkerColor, setMass, sleep
Function	fn_vbs_absSpeed, fn_vbs_vectorAdd

Table 3. Additional configuration file tokens

Token – represented category	Examples of lexemes
Config property	additionalSound, advancedWeaponSelection
Config property prefix	actionBegin, soundHit
Dialog event	onChanged, onKeyDown
Config dialog event	colorText, animPeriod
Class name	Turrets, Grid, Sounds

2.3. Source code editor

Syntax highlighting is not the only advantage of the source code editor of the Hydrogen environment. Great attention was paid to the system of hints and tips, which includes a complete list of applicable methods (Fig. 4) and is used to auto-complete text (Figs. 5 and 6).

[3] Lexical analysis of the source code groups characters into corresponding tokens (categories) [4].

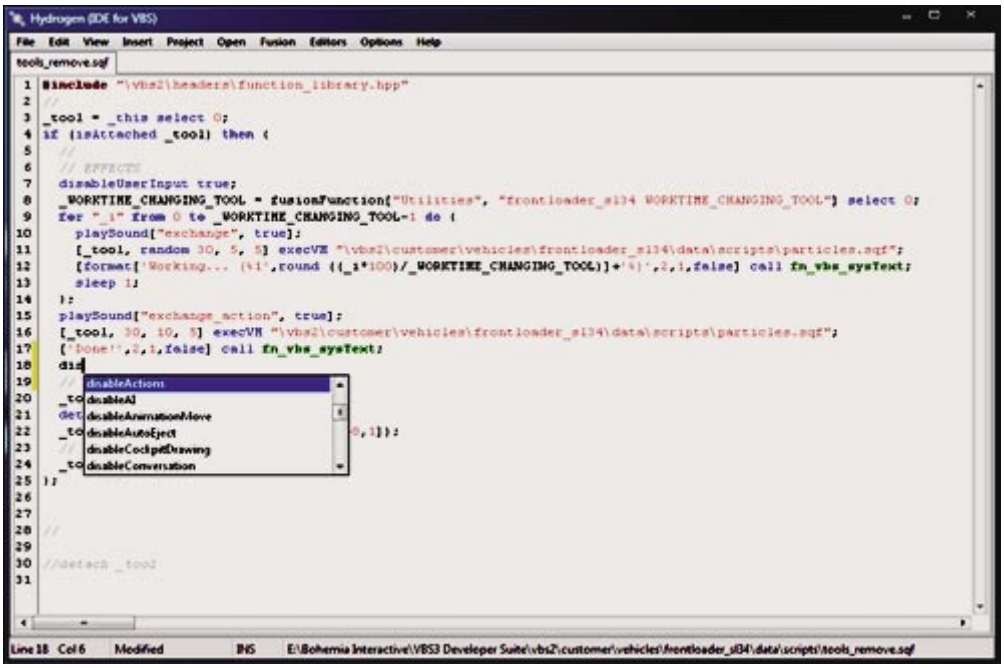

Fig. 4. Main screen of the source code editor in the Hydrogen environment with a hint list displayed (keyboard shortcut: [ctrl+space])

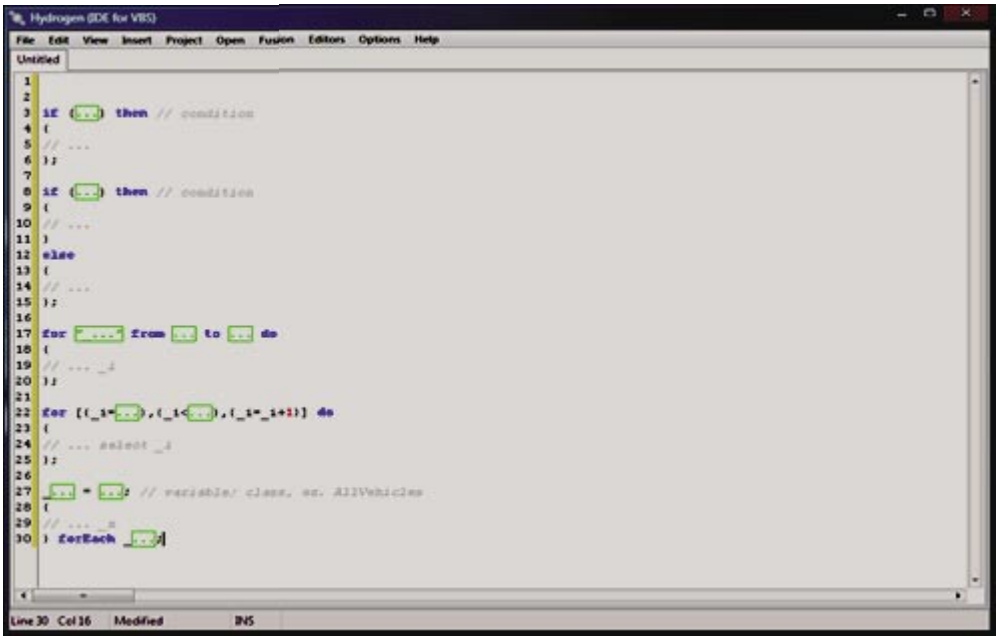

Fig. 5. Main screen of the source code editor with auto-completed SQF syntax ([shift+space] for examples of expressions: if, ifelse, for, fori, foreach)

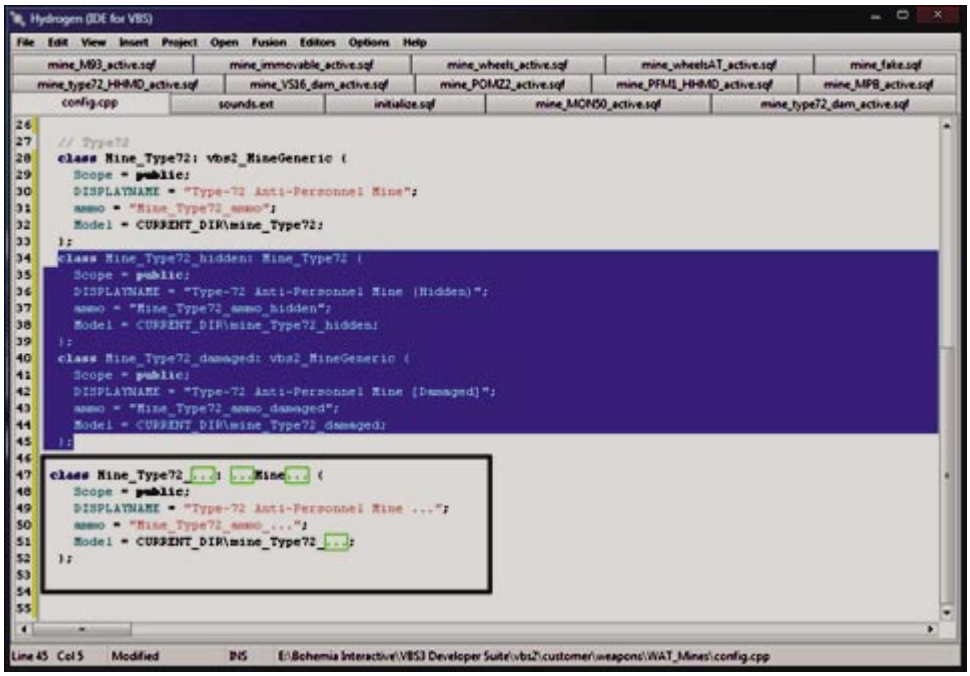

Fig. 6. Illustrative view of class structure copy operation using an algorithm for computing the Damerau–Levenshtein distance [10] ([ctrl+D] copies a class structure based on one or more classes)

Syntax description presented in the form of an HTML browser (Fig. 7) can be obtained by selecting an expression and pressing [F1]. Information is taken from the https://resources.bisimulations.com [11] website using the SSL protocol (OpenSSL library) [12].

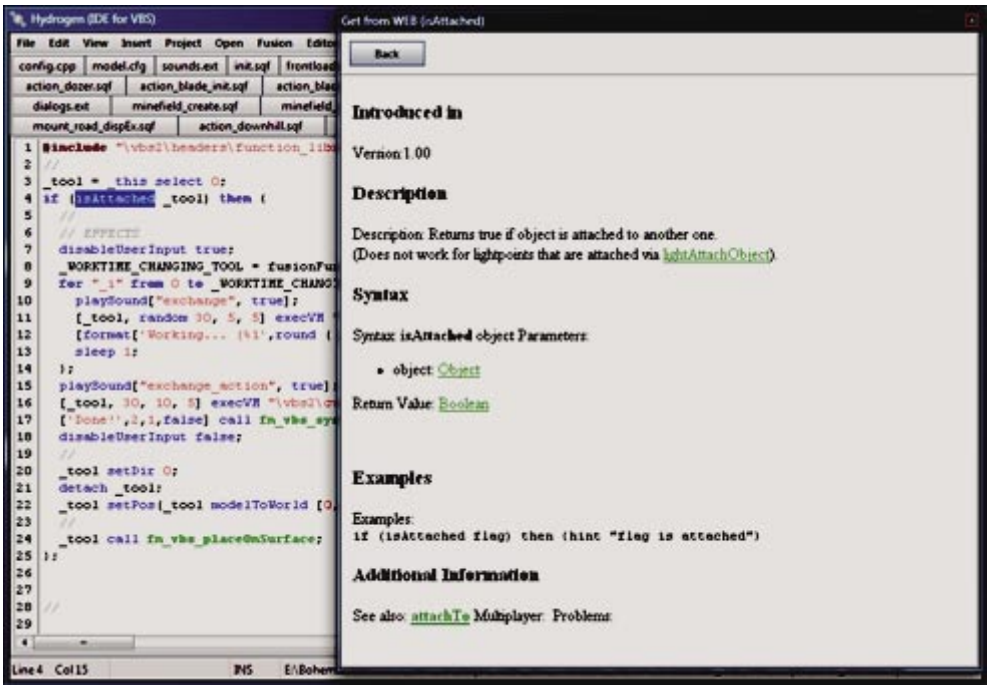

Fig. 7. Main screen of HTML browser ([F1] pressed after selecting a source code expression)

2.4. Object templates and fusion plugins

The Hydrogen development environment was created at OBRUM's Simulator Department in parallel with the development of a project for the Military University of Technology, which included the rendering and adding a number of vehicles, mines, IEDs and handheld mine detectors to the VBS system. In order to expedite the work on the project, the Hydrogen environment was expanded with templates for these types of objects. Figure 8 shows an example of a vehicle design, created from the vehicle template, which consists of nine basic files. The use of templates greatly accelerated and simplified the work and imposed an aesthetic division of the source code making it more readable for the less experienced programmers.

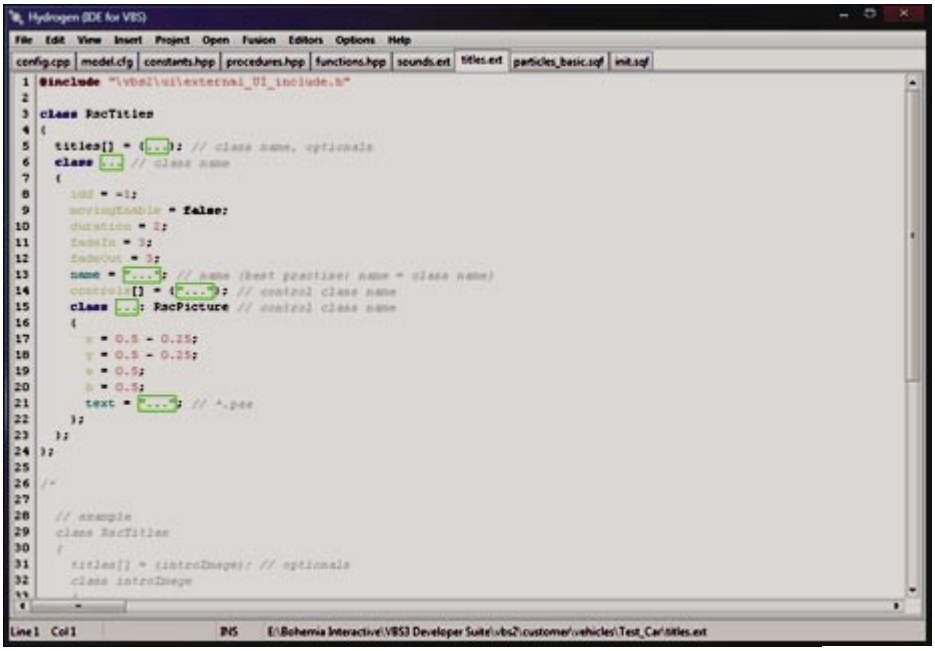

Fig. 8. Main screen of the source code editor with an example of a config file created using a vehicle template in the Hydrogen environment (distinct edit fields indicate the spaces to be completed by the programmer)

Not every functionality can be entered into VDS using scripts and configuration files. Operations on objects defined in VBS can also be conducted using Fusion plugins[4] [13]. The Fusion plugin, that is a DLL library, is created based on a set of C++ classes and constitutes an external expansion of the system [14]. Fusion plugins can be compiled in the Microsoft Visual Studio [15] development environment.

Hydrogen allows designing a plugin for the Visual Studio environment (Fig. 9), which can promptly be compiled into a DLL (Fig. 10).

[4] SQF scripts and Fusion plugins greatly complement each other. However, it is not possible to use only one or the other method in more demanding projects.

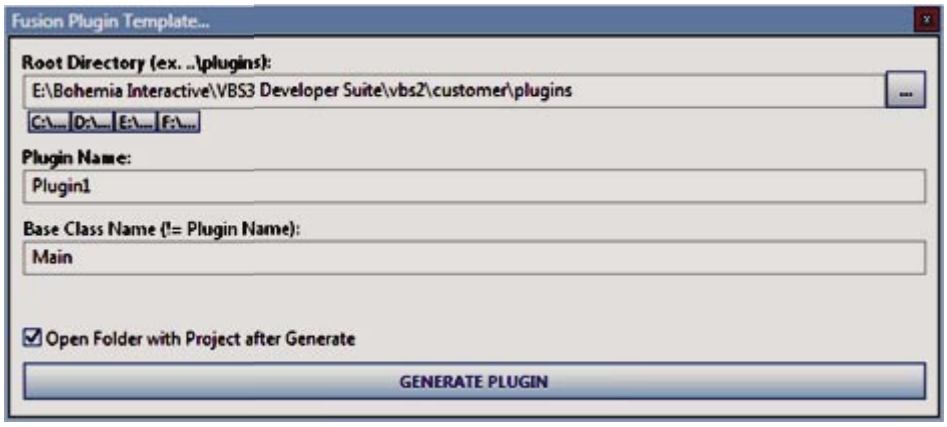

Fig. 9. Edit window for designing a Fusion plugin in the Hydrogen environment

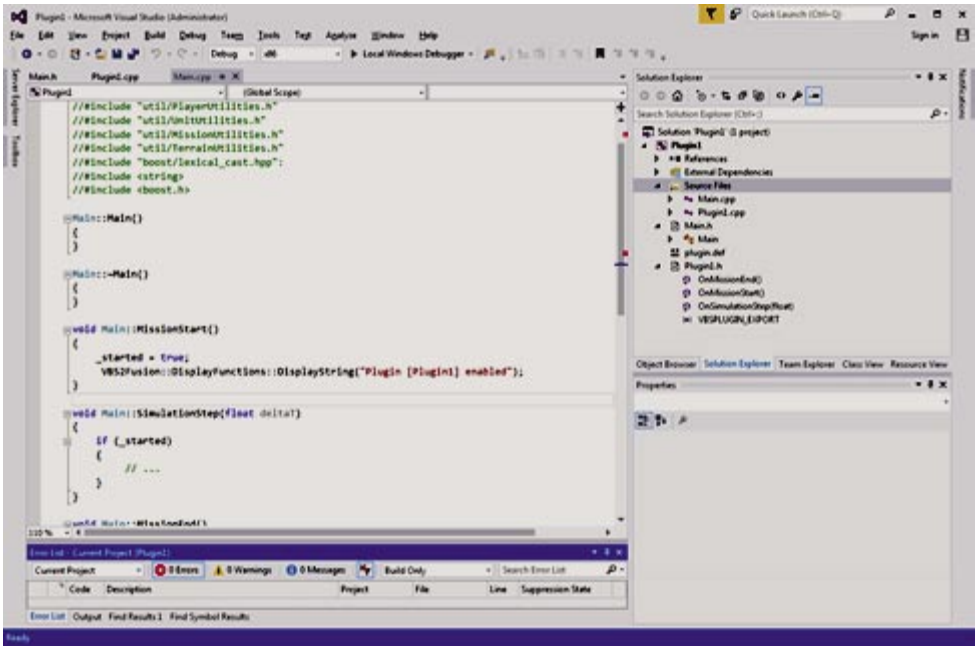

Fig. 10. Main screen of the Visual Studio environment with the source code of a fusion plugin created in Hydrogen environment

2.5. Class browser

VBS comprises a huge set of objects (classes) described by means of SQF scripts and configuration files. Information about classes (class properties and their parameters) contained in the *AllConfig.cpp* and *CfgVehicles.txt* files [16], enabled the development of a system for browsing classes with a search engine (Fig. 11). The tool further includes a tree of class inheritance, which facilitates finding all parameters of a class[5].

[5] Inheritance is a mechanism of sharing a functionality between classes, e.g. *soldier* class (subclass) inherits the functionality of the *human* class (base class). To find all properties and parameters of a class in VBS, the base classes of the subclass must be analysed.

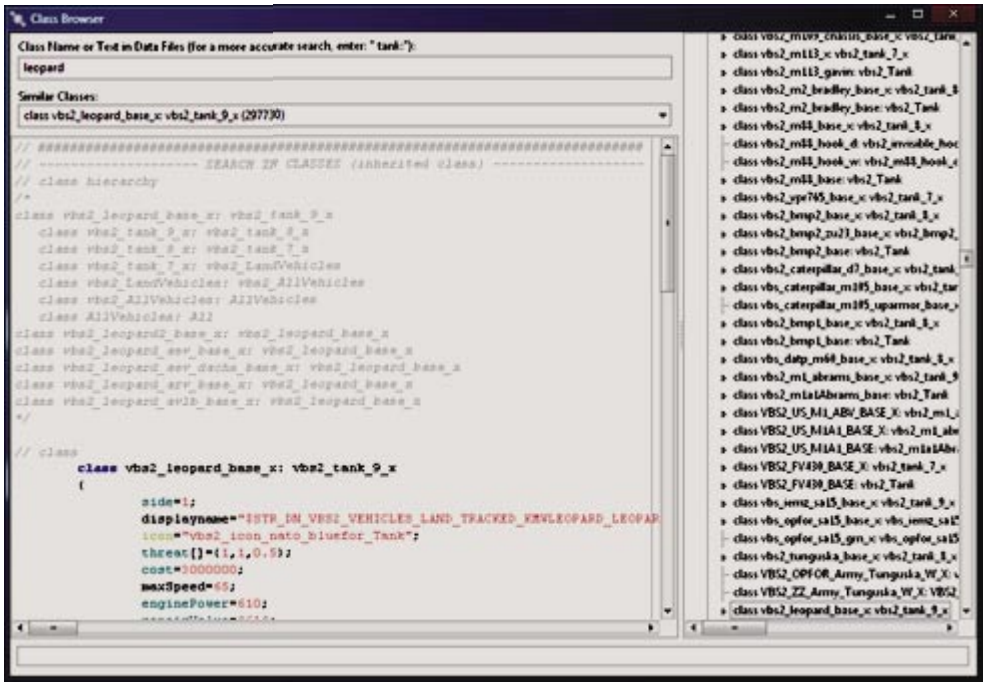

Fig. 11. Main screen of VBS class browser and search engine in the Hydrogen environment

2.5. User interface editor

Designing graphical user interface by means of VBS configuration files (text files) is a very time-consuming, complicated and unnatural[6] process. The author found no information on existing VBS-dedicated user interface editors, and the system itself does not have one (or it is not supplied).

Developing and implementing a GUI (*Graphical User Interface*) visual editor for the preparation of a graphical user interface for VBS (Fig. 12) has become an important factor in accelerating and facilitating work. The editor allows to freely add and manipulate visual components[7], such as: RscButton, RscBox, RscText, RscEdit, RscPicture, RscTitleBar, RscTrackBar, RscProgressBar, RscToolBox, RscListBox, RscComboBox, RscMap, and to automatically generate configuration files (Fig. 13).

[6] Nowadays WYSIWYG (*What You See Is What You Get*) editors are regular content-editing tools. They enable viewing the final effect during editing.

[7] Visual components are given names in accordance with VBS convention.

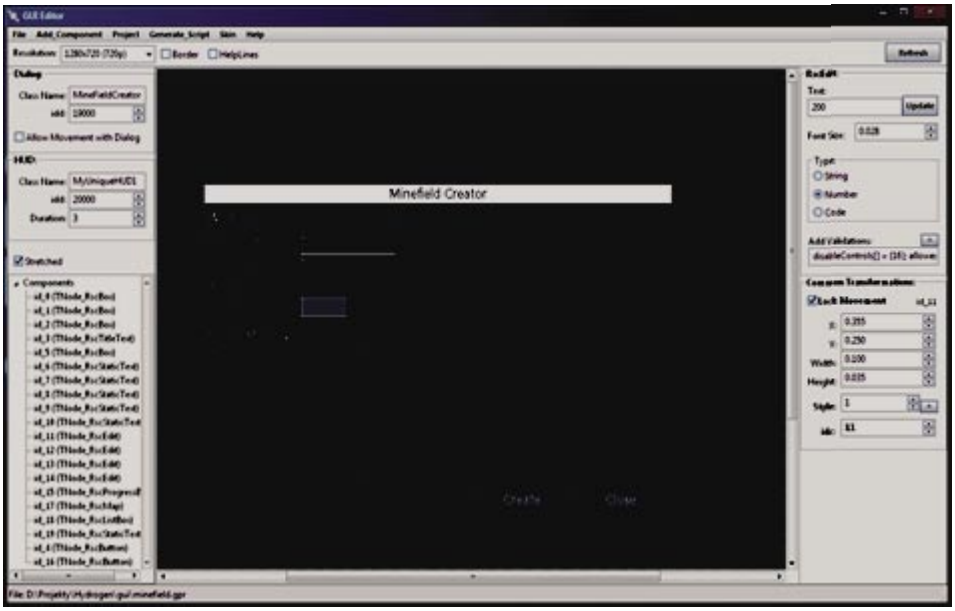

Fig. 12. Main screen of user interface visual editor in the Hydrogen environment showing an example of an interface generating a minefield

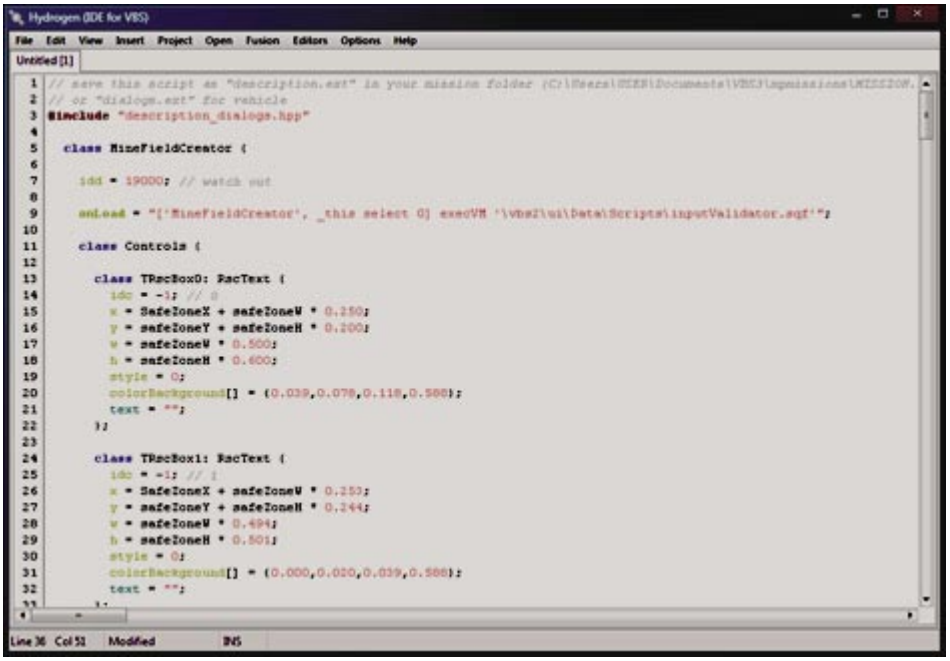

Fig. 13. Main screen of text editor with an automatically generated config file

The end result, i.e. the interface shown in VBS simulation (Fig. 14), almost completely coincides with the interface designed in the Hydrogen environment – differences may relate to fonts and to the lack of dynamic graphic elements in the editor (in the example shown it is the GPS map).

Additional functionalities of the editor include the ability to select multiple visual components (sequencing, aligning), the ability to preview the interface at different resolutions, drawing auxiliary lines, changing colour schemes.

Visual components of the interface editor in the Hydrogen environment were implemented using the Graphics32 graphics library [17].

Fig. 14. User interface in the VBS simulation

3. SUMMARY

The paper describes basic features and properties of the Hydrogen development environment. The article is designed for developers who do not have much VBS experience. It presents the new tool in a straightforward manner. It shows the difficulties that would have to be overcome when working with VBS in the absence of such environment (e.g. when designing a graphical user interface using the configuration script being a text file) and indirectly indicates the work and time savings which can be achieved, for instance, by using templates.

The Hydrogen development environment does not stand out among modern development environments, but it is an example of applying programming expertise and technology in the methods of their formation for convenient creation of software for the Virtual Battlespace simulation system. In comparison to the tools available on the market designed for the VBS simulation system, Hydrogen is a new tool and does not contain any proprietary or commercially licensed sources. Practical usefulness of the Hydrogen environment eliminates the need to use the functionally limited Plugin Script Debugger environment supplied with the developer version of VBS, and that environment itself enables scripting in a quick, orderly and slick manner. The Development of the Hydrogen environment at OBRUM's Simulator Department ensures the continuity of advancement of software (complete source code) as VBS evolves.

Key components of the Hydrogen environment include an advanced source code and configuration file editor, VBS class browser and graphical interface editor, which enables interface designing using visual elements.

4. APPENDIX Hydrogen environment tools - main features

Text editor features:

- source code editor with SQF language syntax highlighting (*.sqf format);
- configuration file editor with syntax highlighting for *.cfg, *.ext, *.hpp, *.config, *.cpp files;
- loading and saving projects with editable list of files included in the project;
- creating backup project copies (archiving files into *.zip file format);
- autocompleting of SQF words and syntax;
- list of SQF and configuration file keywords;
- copying class structure;
- object templates;
- creating templates for Fusion plugins;
- help systems (HTML browser included).

Class browsing system features:

- class finder;
- class hierarchy tree;
- syntax highlighting in the browser and help system included.

Graphical interface editor features:

- WYSIWYG-type editor;
- more than ten visual components;
- more than ten graphical schemes;
- free editing of visual components;
- VBS scripting based on the created interface;
- scalability – the interface may be viewed at various screen resolutions with proportions either preserved or not (*stretch* parameter in VBS).

5. REFERENCES

[1] Bohemia Interactive: VBS3, https://bisimulations.com/virtual-battlespace-3 [Retrieved: 18.09.2016].
[2] Bohemia Interactive: VBS3 NATO, https://nato.bisimulations.com/ [Retrieved: 18.09.2016].
[3] Bohemia Interactive: Script Debugger Plugin, https://manuals.bisimulations.com/vbs3/3-6/manuals/Content/Editor_Manual/SM_ScriptDebugger.htm [Retrieved: 19.09.2016].
[4] Aho A.V., Sethi R. Ullman J.D.: Compilers: Principles, Techniques, and Tools, 1986, Addison Wesley, ISBN-10 0-321-48681-1.
[5] Free Pascal Team: Free Pascal, http://www.freepascal.org/ [Retrieved: 20.09.2016].
[6] Lazarus and Free Pascal Team: Lazarus, http://www.lazarus-ide.org/ [Retrieved: 20.09.2016].
[7] Bohemia Interactive: Addon Packer, https://manuals.bisimulations.com/vbs2/2-00/devref/Content/Adding_Models/AM_Packing.htm [Retrieved: 21.09.2016].
[8] Bohemia Interactive: Oxygen Manual, https://manuals.bisimulations.com/vbs3/3-4/devref/Content/Oxygen_Manual/Oxy_Oxygen_Manual.htm [Retrieved: 21.09.2016].

[9] Bohemia Interactive: Text Editors, https://resources.bisimulations.com/wiki/Text_editors [Retrieved: 22.09.2016].
[10] Damerau F. J.: A technique for computer detection and correction of spelling errors, 1964, http://portal.acm.org/citation.cfm?id=363994 [Retrieved: 20.05.2016].
[11] Bohemia Interactive: VBS Scripting Reference, https://resources.bisimulations.com/wiki/Main_Page [Retrieved: 21.05.2016].
[12] OpenSSL Software Foundation: Welcome to OpenSSL! ,https://www.openssl.org/ [Retrieved: 21.05.2016].
[13] SimCentric Technologies, VBSFusion, https://www.simct.com/products/vbs3/vbsfusion [Retrieved: 16.08.2016].
[14] Wantoch-Rekowski R.: VBS2: Programowalne środowisko symulacji wirtualnej, 2015, PWN, ISBN 978-83-01-18170-3.
[15] Microsoft, Visual Studio 2013, https://msdn.microsoft.com/pl-pl/library/dd831853(v=vs.120).aspx [Retrieved: 17.08.2016].
[16] Bohemia Interactive: Startup Parameters, https://community.bistudio.com/wiki/Virtual_Battlespace:_Startup_Parameters [Retrieved: 17.08.2016].
[17] Graphics32 Team: Graphics32, http://graphics32.org/wiki//Main/HomePage [Retrieved: 4.09.2016].