Journal of Computer-Aided Molecular Design **18:** 167–173, 2004.
© 2004 *Kluwer Academic Publishers. Printed in the Netherlands.*

VEGA – An open platform to develop chemo-bio-informatics applications, using plug-in architecture and script programming

Alessandro Pedretti, Luigi Villa & Giulio Vistoli*
Istituto di Chimica Farmaceutica, University of Milan, Viale Abruzzi, 42, I-20131 Milan, Italy

Received 11 November 2003; accepted in revised form 29 March 2004

Key words: molecular editor, molecular visualization, open application, plug-in architecture, script programming

Summary

In this paper we present the expandability and flexibility features of the VEGA program (downloadable free of charge at http://www.ddl.unimi.it), for the development of custom applications, using it as a multipurpose graphical environment. VEGA can be customized using both plug-in architecture and script programming. The first is useful to add new features and functions, using homemade routines, written with the VEGA Plug-in Development Kit (SDK). With the second approach it is possible to design scripts in VEGA, using the REBOL language, in order to (1) add new functions or customize existing ones; (2) automate common procedures; and (3) allow network communications, by creating a bridge between VEGA and other applications (or other PCs) through the TCP/IP protocol.

Introduction

Scientific applications can be viewed as black boxes, which perform ‘magical’ operations, following well-documented computational protocols. This vision is the easiest for the final user, but is not fertile for the scientific community, since it leads users to blindly accept computational capabilities of scientific software.

The diffusion of open source applications [1] seems to address this behaviour: the quest for open applications is rapidly growing and several groups freely share their computational resources [2]. Numerous researchers would like to create their own scientific applications, even if the *ab initio* design is a time consuming and not always feasible process. In contrast, the development of specific programs, by assembling and/or modifying computational tools and source libraries can be simpler.

In particular, specific programs can be achieved by using: (1) routines libraries that collect several pieces of code to perform the most important operations, by combining them as a puzzle game; (2) open applications, that can be used as a platform, including basic operations (essentially visualization and manipulation of molecular structures), in order to enrich them with new features to perform specific operations [3–7]. We have decided to follow this second approach and this paper describes the features of VEGA1.5.1 [8] programming under the Windows operating system. The VEGA expandability can be obtained by both plug-in architecture [9] and script programming [10]. The former is more suitable if new features must to be added to the program, while the latter is useful to automate computational procedures or to modify pre-existing functions. It should be noted that the plug-in programming is more complex, as it requires programming knowledge to work on the VEGA code, while the script programming is user-friendlier, as the scripting languages are more intuitive and less laborious.

In VEGA a plug-in is a DLL (Dynamic Link Library) that automatically runs when VEGA starts. The Software Development Kit (SDK) is tested with GCC [11] and Borland C/C++ [12] Builder and because the code is ANSI compliant, it should work with any conforming compiler. Moreover, it is also possible to write plug-ins in the Fortran language, using specific routines, which encapsulate the features of the C/C++

*To whom correspondence should be addressed; e-mail: giulio.vistoli@unimi.it

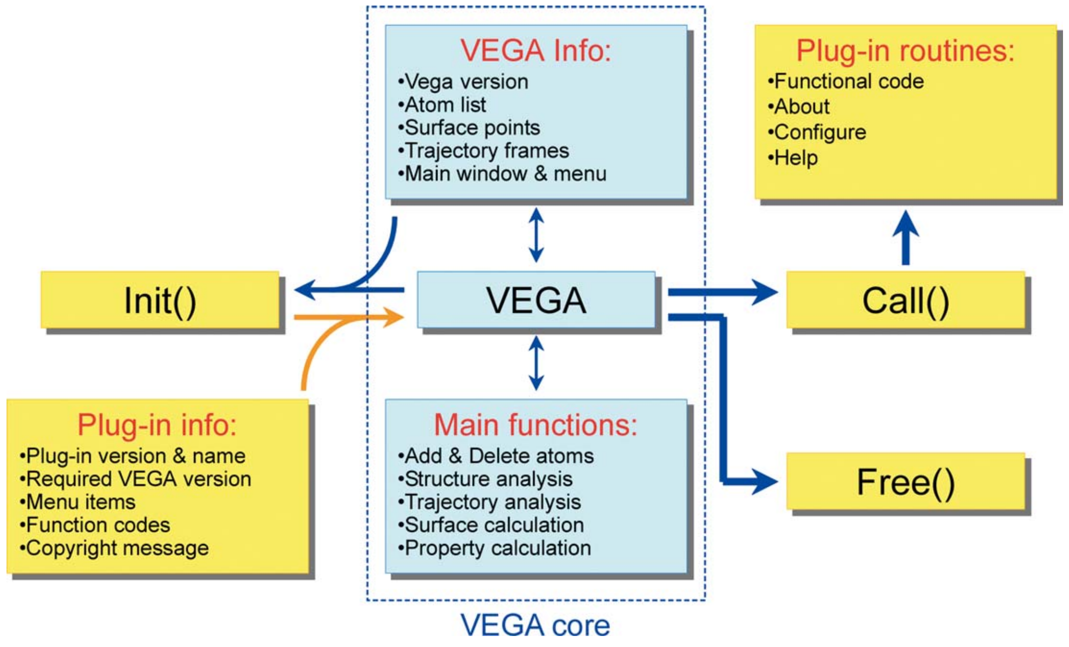


Figure 1. Main logical components of the plug-in architecture.

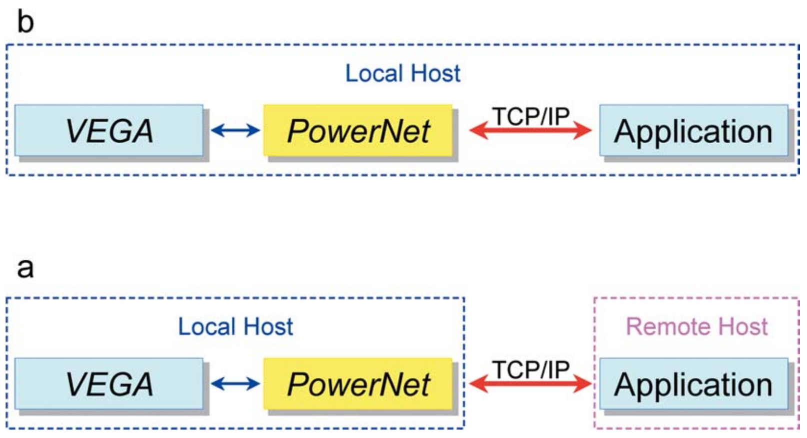


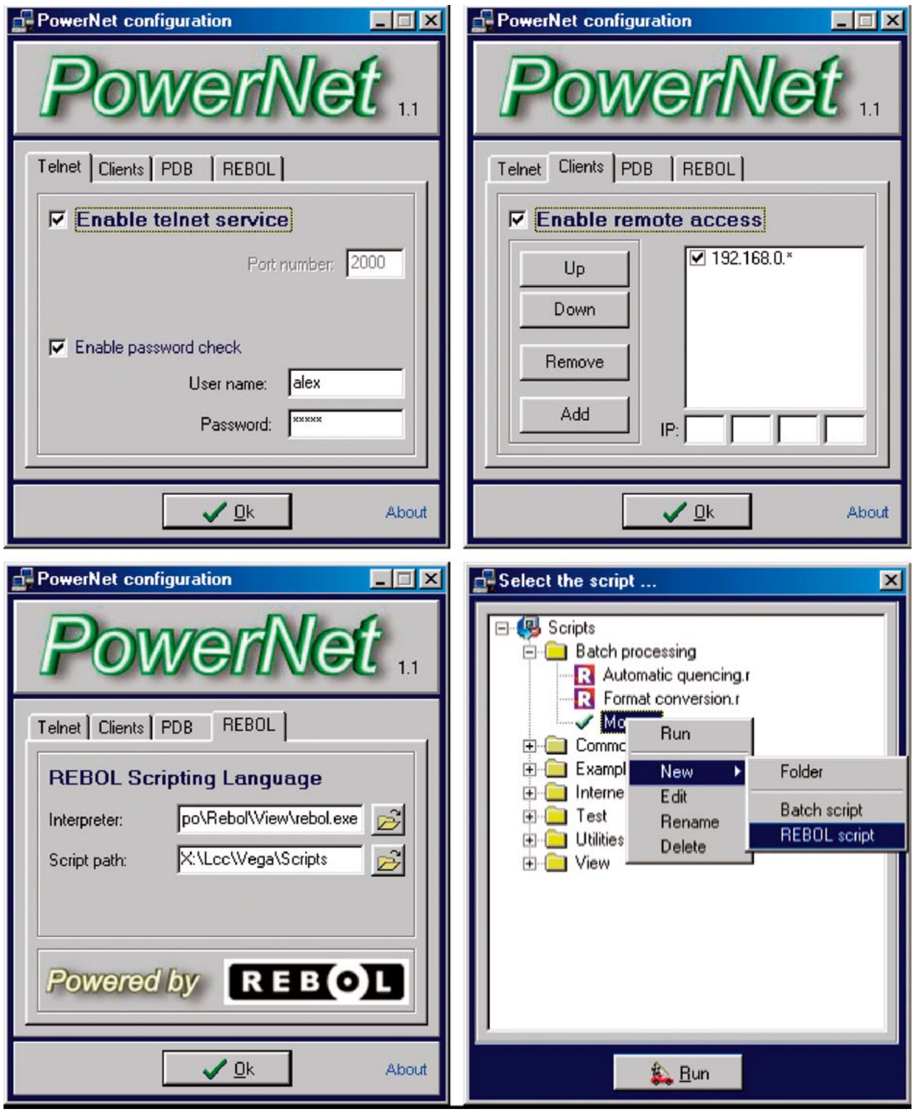
Figure 2. VEGA communications via TCP/IP port (a) application on local machine; (b) application on remote machine.

language (e.g. the pointers) to call VEGA routines from Fortran code [13].

Furthermore, it is possible to write scripts in VEGA, using the REBOL language [14]. VEGA accepts scripts via the TCP/IP network protocol, hence linking VEGA to other programs or PCs.

Plug-in architecture

A plug-in DLL must have three exported functions, Init(), Call() and Free(), whose definitions, constants and macros are included in the *plugin.h* header file. Figure 1 reports the main features of these three functions. The Init() function allows the plug-in initialization, describing plug-in data and menu items, that will be added to the VEGA main menu. Further, the Init() function sets two structures, VGP_VEGAINFO and VGP_PLUGINFO: the first includes information on the VEGA software (program version and release), the atom list, the atom number, the surface points and the handle for the main menu and the main windows. The second includes plug-in information as name, version, copyright, and VEGA version required. VGP_PLUGINFO must also define a list of functions that contains data such as menu item position,



☑ Enable telnet service
Port number: 2000
☑ Enable password check
User name: alex
Password: *****

☑ Enable remote access
☑ 192.168.0.*

Interpreter: po\Rebol\View\rebol.exe
Script path: X:\Lcc\Vega\Scripts

Figure 3. PowerNet plug-in main window: Telnet tab (top left), client tab (top right), REBOL tab (bottom left). At the bottom right the script manager can be seen.

menu item string, menu item image and corresponding source code. The Free() function releases all resources once the program has terminated. In this subroutine it is important to include all code necessary to free the resources used by the plug-in. Finally, the Call() function is the core of the plug-in, as it defines the code associated with a specific menu item. When one selects a plug-in menu item or presses a button in the plug-in window, VEGA calls the corresponding subroutine, hence passing the function code. There are four types of function code: (1) About routine: displays information and a copyright message; (2) Configure routine: configuration of the plug-in for activation; (3) Help routine: shows the help instructions; (4) Function routine, namely the true plug-in function code.

A plug-in can control all VEGA functions, read and modify the atom list, add a new atom and remove an atom from the list. VEGA stores atom information in a memory list. Each atom has its own allocated memory space and each atom structure has the pointer to the next atom. Thus, this list can be scanned only in one direction, namely from first atom to last one. The end signal of the atom list is a null pointer to the next atom structure. In VEGA there is a pointer to the first atom in the atom list, to the last element and to the total number of atoms in memory. Each atom is described by a set of data including: atom name, atom number, residue name, residue number, chain indicator, 3D coordinates, atomic radius, atomic charge, atom type, connectivity matrix and bond order. A plug-in can allocate corresponding memory to add a new atom, updating pointers to total atoms and last atom. A plug-in can delete an atom from the list, by changing the corresponding pointers. These operations can be carried out on the atom list of a molecule, of a trajectory and on the point lists of a surface, which includes 3D coordinates and their associated value (it can be point color or a point parameter, e.g. MEP, MLP [15], ILM [16], PSA, etc.). Using these functions as well as the specific VEGA features, any molecular structure can be analyzed and modified. All VEGA function could be directly controlled by the plug-in, using the standard Windows messaging system.

With the *SendVegaCmd* function, it is possible to send specific commands to the VEGA interface. It is a little Win32 shell command that can be used to control the VEGA main menu in batch files. The user can send extended commands and Window commands. The menu commands allow to control the VEGA functions calling the menu items. The extended and Windows commands control several other VEGA features that will be briefly examined in the following sections.

The Plug-in SDK, available in the current VEGA release, includes the full source code of Dhrystone plug-in, an old benchmark to test the CPU performance. The plug-in was developed in C/C++, using Reinhold P. Weicker's original code, to show how the user can take advantage of plug-in architecture. Moreover, the VEGA SDK also includes the code of a Null plug-in, which does not perform operations, but can be used as a scaffold to write new plug-ins.

The VEGA program also includes plug-ins to interface VEGA to Predator [17], NJPlot [18] and ClustalX [19]. These plug-ins encapsulate external programs in VEGA and the user can control all options (e.g. select the files, the databases, the sequences to be predicted, etc.), using an intuitive graphical interface.

Power Net – network interface

PowerNet is the most powerful plug-in, as it allows VEGA to communicate with other applications through the TCP/IP protocol. These applications can run on either a local or remote machine (Figure 2). Indeed, the PowerNet TCP/IP port can send and receive command-line instructions, working as a POP3 server (Post Office Protocol) [20]. An example of an interface application is the REBOL scripting language that allows the design of scripts to customize VEGA features. PowerNet also allows virtual communication with applications, which include a simple TCP/IP client. PowerNet includes access control in order to protect the system from Web hacking attacks.

The PowerNet functions can be summarized as follows (Figure 3):

(1) *Telnet service*: the principal function to enable the TCP/IP port for local and/or remote connection. The user can change the port number (to start more VEGA sessions without conflicts) and enable password checks, by specifying user name and password.

(2) *Client service*: enables remote access granting VEGA control by other PCs. Access can be denied to specific hosts, defining their IP.

(3) *Log Service*: in order to check TCP/IP connections and script execution, it is possible to enable the PowerNet event logging. The log service is a text file containing the time of the event, the client information, the submitted commands, the results and the monitored errors.

(4) *PDB service*: connects VEGA to PDB archive and enables structures to be directly downloaded from databases. The user can specify the preferred PDB server and the query string. Downloaded files can be stored in specified folders in order to build a local database.

(5) *REBOL service*: PowerNet adds the ability to manage REBOL scripts in VEGA. PowerNet includes the REBOL/core package [21], but it is possible to use other distributions with more features, like the REBOL/view, to design graphic interfaces.

REBOL scripting language

REBOL is a messaging language, which provides a broad range of practical solutions to the daily challenges of Internet computing. REBOL/Core is the foundation for all REBOL technology. While designed to be simple and productive for novices, the language extends a new dimension of power to professionals. REBOL offers a new approach to the exchange and interpretation of network-based information over a wide variety of computing platforms. REBOL scripts are as easy to write as HTML or shell scripts. A script can be a single line or an entire application.

Three main functions, included in the 'Vega.r' interface file, allow communications between REBOL/core and the VEGA program, through a TCP/IP port:
(1) VegaOpen: connects the REBOL interpreter to the VEGA TCP/IP port, by defining a host name (or IP), TCP/IP port number, user name and password. Changing the *VegaOpen* parameters the user can run a script on a PC to control remote VEGA hosts.
(2) VegaCmd: sends a command to VEGA in synchronous mode. If an error occurs, the command returns the standard VEGA error code, or 0. The error code is also reported in the *VegaErrCode* while the error description is given in the *VegaErrStr* REBOL variable. If the sent command returns an argument value, this is placed in *VegaRes* REBOL variable.
(3) VegaClose: closes and frees the communication port. Each VEGA session can run one script at a time.

Users can easily exploit all VEGA functions via VegaCmd. For example, the script:

VegaCmd {*Open* "*Mymolecule.pdb*"}

opens the *Mymolecule.pdb* file. With VegaCmd the user can also obtain information from VEGA. For example, the following script:

VegaCmd "*GetTotAtm*"

print VegaRes

takes the *TotAtm* variable value (the total number of atoms), saves it in the *VegaRes* the REBOL variable and shows it in REBOL console.

VEGA interprets commands with extended syntax that can be sent via console window, windows class port and TCP/IP connection (*PowerNet* plugin).

The users interested in VEGA scripts can consult the full REBOL manual at the REBOL Technologies website [22] and the complete list of VEGA extended commands in the VEGA manual [23]. These commands can be divided into three main classes:
(1) *Chemical commands*: analyze, modify and manage molecules, surfaces and trajectories; examples: ANGLE: calculates the angle between three atoms; CHARGE: assigns the atomic partial charges; SRFCALC: calculates the molecular surface; TRJSEL: opens a selected trajectory.
(2) *Graphical commands*: set and modify the graphical environment; examples: COLOR: changes the color for selected atoms; LIQPROP changes the liquorice visualization properties; REFRESH: forces the main window refresh.
(3) *System commands*: control the VEGA program and the PC; examples: CHDIR: changes the directory; MESSAGEBOX: shows a message box; PLUGINCALL: calls a user plug-in.

VEGA includes several examples of REBOL scripts that are grouped in five classes: Batch processing, Common, Utilities, Internet and View. Each class contains examples that the user can modify for specific purposes.

Discussion: Why choose VEGA as open application?

The question that spontaneously arises, analyzing the VEGA expandability, is why use VEGA to design new applications? In our opinion there are at least four main reasons to prefer VEGA in scientific development:
(1) *Atom typing*: The Atom Type Description Language (ATDL) [24] is a molecular pattern language for substructure searching and atom types assignment. The atom type recognition is of pivotal importance in several computational procedures like force field calculations (molecular mechanics, dynamics and docking), drug design (database mining) and atom based property calculations. ATDL can support any atom classifications using a suitable template file and the user can implement new atom type categorizations, by writing relative file. ATDL is basically based on atomic number, hybridization, bond type and bonded partners [24]. Several template files are already included in the VEGA program, both for force field attribution (CVFF, CHARMM, Tripos) and property calculation (Broto–Moreau and Ghose–Crippen for logP calculation, Gasteiger atomic charges calculation).
(2) *Topology File*: It is well known that a major difference between free programs and commercial products is that the latter are much simpler to use. For example, several free MD programs require the molecular topology file, whose definition is not simple and does not permit the work for novice researchers. On the contrary, commercial packages automatically compile the topology file without the user even realizing it. The topology file includes the geometrical data for a considered system: a list of bonds, planar angles, torsions, improper torsions and, sometimes, a list of proton acceptor and donor groups. Some free programs include topological templates for common blocks such

as amino acids, nucleotides, sugars, but the user must write the topological file for small ligands. VEGA can resolve this problem, as it compiles the full topological file for any molecular system and stores it in PSF format. The user can take advantage from the compiled information to write topological files in different formats. On this ground VEGA can become a tool to assist the use of freeware MD programs, giving them a user-friendly graphical interface.

(3) *Database*: The growth of combinatorial chemistry induced a connected development of computational technologies for the *in silico* screening of large molecular databases. For this reason VEGA implements a database manager able to read sets of molecules in several file formats. VEGA is able to create a database, read or modify (adding or deleting a molecule) an existing one. The script programming suits the database handling well, as the user can write scripts that perform specific computational protocols (MM, MD, docking, property calculations, etc.) on each molecule in a database. Moreover, the database can also include MD trajectories and the user can write scripts that perform specific operations on each frame of each trajectory.

(4) *Molecular Editor*: The VEGA program has several features to build and modify molecular structures, preparing them for calculations. VEGA includes: (1) a molecular editor to build small compounds using libraries of selected fragments; (2) the ability to add, modify and delete atoms, bonds, segments or chains; (3) an algorithm to add hydrogen atoms based on both residue structure (for proteins and nucleic acids) and structural properties (for small organic compounds); and (4) a solvent editor to create clusters with any solvent and to solvate a molecule.

These fundamental features of the VEGA program as well as its graphic facilities make it a molecular modelling package which, in combination with a freeware MM/MD package (NAMD [25], for example), can be used to perform computational research at no cost using a PC. Indeed, we can imagine a computational protocol in which the user downloads a protein structure from PDB, refines the structure by adding hydrogen atoms or modifying side chains, adds or modifies a ligand into a protein structure, adds an appropriate solvent, assigns atom types and atomic charges (using the preferred force field), saves the obtained structure in a suitable file format, compiles the topological file, runs the calculations with a free MD/MM program and finally analyzes the obtained trajectory. All operations can be performed with VEGA, which can compete with commercial programs, especially regarding the capability to carry out MM and MD simulations with a user-friendly interface. On this ground VEGA expandability becomes very important: indeed, the user could both automate and customize these computational procedures, using script programming and modifying or adding new features, via plug-in architecture.

Conclusions

This paper describes how VEGA tackles the problem of application expandability. This program includes the two typical solutions that are based on both object-oriented programming (plug-in) and components oriented programming (scripts). The last approach is especially interesting because (1) it requires less laborious programming as users do not need to know how the components work, but only how they can be combined together; (2) it allows both to automate the most common procedures and to add (or modify) new features in the VEGA program; (3) using PowerNet, it is possible to allow VEGA to communicate with other applications (or PCs).

As a final thought, we believe that the VEGA program allows the scientific community to approach molecular modelling. Researchers can also contribute to this approach, by customizing VEGA with their components (plug-ins or scripts).

References

1. The Open Source Initiative (OSI): www.opensource.org
2. Several scientific open source applications can be found at www.sourceforge.net
3. Csizmadia, F., J. Chem. Inf. Comput. Sci., 40 (2000) 323.
4. Blauch, D., J. Chem. Inf. Comput. Sci., 42 (2002) 143.
5. Steinbeck, C., Krause, S. and Willighagen, E., Molecules 5 (2000) 93.
6. JOELib – a java based computational chemistry package, http://joelib.sourceforge.net/
7. Steinbeck, C., Han, Y., Kuhn, S., Horlacher, O., Luttmann, E. and Willighagen, E., J. Chem. Inf. Comput. Sci., 43 (2003) 493.
8. Pedretti, A., Villa, L. and Vistoli, G., J. Mol. Graph., 21 (2002) 47.
9. Gamma, E., Helm, R., Johnson, R. and Vlissides, J., Design Patterns: Elements of Reusable Object-Oriented Software. 1995, Addison-Wesley, Boston, MA.
10. Andreoli, J.M., Arregui, D., Pacull, F. and Willamowski, J., Lect. Notes Comput. Sci. 2480 (2000) 429.
11. Free downloadable at http://www.delorie.com/djgpp/. You can use also mingw gcc (http://www.mingw.org/).

12. Borland Software Corporation, 100 Enterprise Way, Scotts Valley, CA, USA.
13. Contact the Authors for further information about the Fortran programming in VEGA.
14. http://www.rebol.com
15. Gaillard, P., Carrupt, P.A., Testa, B. and Boudon, A., J. Comput.-Aided Mol. Design, 8 (1994) 83.
16. Pedretti, A., Villa, A.M., Villa, L. and Vistoli, G., 'Lipohilicity profile of WB-4101 analogues using the ILM approach', Internet Journal of Chemistry, Vol. 3, Art. 13 (30/06/2000).
17. Frishman, D. and Argos, P., Proteins, 27 (1997) 329.
18. Perrière, G. and Gouy, M., Biochimie, 78 (1996) 364.
19. Thompson, J.D., Gibson, T.J., Plewniak, F., Jeanmougin, F. and Higgins, D.G., Nucl. Acids Res., 25 (1997) 4876.
20. Gay S. and Simon M., Lect. Notes Comput. Sci., 1576 (1999) 74.
21. REBOL/core and REBOL/view are free downloadable for non commercial use at http://www.rebol.com/download.html
22. http://www.rebol.com/docs/core23/rebolcore.html
23. http://users.unimi.it/∼ddl/vega/manual/pages/gl_index.htm
24. Pedretti, A., Villa, L. and Vistoli G., Theor. Chem. Acc., 109 (2003) 229.
25. Kalé, L., Skeel, R., Bhandarkar, M., Brunner, R., Gursoy, A., Krawetz, N., Phillips, J., Shinozaki, A., Varadarajan, K. and Schulten, K., J. Comput. Phys., 151 (1999) 283.