Jl. of Computers in Mathematics and Science Teaching (2005) 24(1), 43-72

Carbopolis: A Java Technology-Based Free Software for Environmental Education

MARCELO LEANDRO EICHLER
Federal University of the State of Rio Grande do Sul
Brazil
exlerbr@yahoo.com.br

PAULO RAFAEL XAVIER AND ROBERTO COSTA ARAÚJO
PROCEMPA
Brazil
aeq@iq.ufrgs.br

RAFAEL CASTRO FORTE AND JOSÉ CLAUDIO DEL PINO
Federal University of the State of Rio Grande do Sul
Brazil
aeq@iq.ufrgs.br

The goals of this paper are to describe some characteristics of the pedagogical project of the *Carbopolis* software and some programming solutions that were found during the computational implementation of this software. Relative to the first goal, some pedagogical features that are considered necessary to developing computerized learning materials for scientific education are analyzed. To that end, the design of scenarios is presented as a way of linking learning strategies, such as simulation, problem solving, and play. Related to the second goal, there is a brief description of some potential that JAVA technology holds for the computational implementation of pedagogical projects with such features. The context of the inclusion of the computational implementation using free software is also shown. Eventually, the programming solutions developed using a graphical interface, scenario mapping, data storage, bookmarking, notepad, browser (developed specifically for *Carbopolis*), and the software installa-

tion are described. In addition, some information obtained in two contexts of the pedagogical project evaluation is presented: one context attempted to evaluate conceptual learning; the second attempted to show the software usability in different school settings.

The Pedagogical Project

The pedagogical project, synonymous with *design* in this paper, is the kernel of learning materials. Through the pedagogical project, teaching strategies are developed and learning sessions are planned. It is within the general guidelines of the design that students' features of interaction and evaluation are decided (El-Tigi and Branch, 1997).

Thus, during the development of computerized learning materials, the most common errors are related to the decisions of the designer who plans the activity. These errors often happen because the main intention was only to update the instruction material and not to widen its usefulness (Carroll, 2000). For example, even if computers make the use of the highest instruction levels possible, in order to increase the student's control over her/his learning (such as the strategies of simulation and problem solving), they can also be used for mundane drill and practice (training and repetition). This means that the environment does not dictate the design (Starr, 1997). This is why the quality of the learning resources currently available on the Internet is quite variable, ranging from excellent to extremely poor (El-Tigi and Branch, 1997).

Several kinds of knowledge provide opportunity for guiding the design decisions. The software pedagogical project, which is described in this paper, was grounded on the insights of Jean Piaget's epistemology (1970, 1974a & 1974b) and on Paulo Freire's reflections on teaching (1996).

Concerning Piaget's theories, we believe that the main elements for an effective learning experience are *activity* and *discovery* (Carrol, 2000). Moreover, these activities are meaningful to the student when they are linked to his/her personal interests, needs, and objectives. This happens because people want to learn in a realistic environment; they want to be able to use what they know in order to critically evaluate the new knowledge and skills they encounter. These themes are not new. They were widely developed by psychologists as Jerome Bruner, John Dewey, and Jean Piaget. Although these themes were well known in the 1970s, they were not discussed in the outline and in the structuring of proposals for the educational use of computers (Carrol, 2000).

ercises that could be used by intermediate level students of chemistry. The first chapter of the learning material, titled *Supplying our Water Needs*, starts with a fictional approach. A newspaper reports on a "fish kill" environmental problem in the region of Riverwood; many fish have died and residents are preoccupied with the possible pollution of the water. In this chapter, as well as in the classes, the student must solve this problem, employing several concepts and methods of chemistry. One of the computer activities uses the Lake Study software (Whisnant, 1984). This software has a simulation that allows the student to change an aquarium temperature, as well as the quality and concentration of different chemical compounds in the water, checking the fish killed in the aquarium in relation to the conditions that were chosen. Thus, the problems that are initially approached in a fictional context are used as a methodological and conceptual reference to the development of the small investigative projects. For example, after studying the characteristics of the water supply in the fictional region of Riverwood, the students are challenged to investigate this theme in the context of the region where they live. However, this learning material was produced in English and no similar Portuguese material exists. Thus, it is difficult to use it in the Brazilian educational system.

When using digital technology, several tools can be incorporated into this type of design, such as notepad, hypertext, network access, and discussion lists. The hypertext is often seen not as a simple technique, but as a type of epistemological metaphor to the interactivity. However, once again, one has to be careful with exaggerations, especially because hypertext systems frequently do not achieve their full potential, in large part, due to the poverty of their hierarchic project and their interrelations (Otter & Johnson, 2000).

Worldwide, there are some products or projects that have the majority of the characteristics presented in this paper (Calza & Meade, 1998; Christian, 2000; Sun & Chou 1996; Zimmerman, Thomas, Gan & Murillo-Sanchez, 1999). However, many of these products and projects deal with formal content and are developed for higher education. There are also some interesting commercial products, such as those from Falcon Software (http://www.falconsoftware.com) or the Logal High School Science Explorer Series from Riverdeep (http://www.riverdeep.net). Although they are pedagogically useful, they may cost many thousands of dollars, making them too expensive for use by developing countries. Moreover, the use of fee-based software goes against the policy of free-software, which was adopted when choosing the technologies for the pedagogical project implementation.

The next part of this paper will briefly describe the characteristics of the educational software called *Carbopolis,* which is implemented in Java under

the concept of free software—a concept developed from the ideas presented in this paper. To acquire this software in Portuguese and Spanish, as well as its source code, please visit: www.iq.ufrgs.br/aeq/carbop.htm.

THE *CARBOPOLIS* DESIGN

Carbopolis is a computer program that deals with environmental pollution. It was developed for students and professors with different education levels. The program uses problem solving and play to deal with chemical and environmental concepts related to air pollution and acid rain.

The topics addressed in the program are based on real environmental problems, for example, those that involve the thermo-electrical power plant of Candiota in the State of Rio Grande do Sul in Brazil (Fiedler, Martins, & Solari, 1990).

The problem presented in *Carbopolis* deals with a decrease in farming and cattle production in a region next to a thermo-electrical power plant. In order to solve the problem, the student must determine the damages and their origin, as well as suggest a solution that reduces or removes them. The students are supposed to find out that the problem was caused by acid rain, which was caused by the burning of coal in the thermo-electrical power plant of the region (Eichler & Del Pino, 2000).

When using the program, some tools at the student's disposal make her/him aware of what is going on in the region. The student can, for instance, refer to the declarations of several characters, such as some farmers of the region, a power station public relations employee, a ranger, a miner, and even the mayor of the city. We can also find some tools for the sampling and analysis of rainwater and air quality, as well as a library for research. Besides text, the library also includes pictures, for example, the picture of the biogeochemical cycles involved.

During the solution of the proposed problem, the student can formulate hypotheses about the cause of the problem and suggest a solution; for example, she/he can install anti-pollutant equipment. If the student guesses that the environmental problem is related to the increase in concentration of sulphur dioxide in the air, she/he can use the air pollution control system. In other words, the student can use available equipment to rid the air of sulfur and can then collect and analyze samples to detect whether or not the rainwater and air quality improves.

Finally, the text in *Carbopolis* uses a hypertextual format. In other words, the information related to the understanding and solution of the pro-

posed problem is actively connected (i.e., linked). This makes it possible to explore such information instantly, according the order requested by the user. Thus, the way the chemical and environmental concepts are dealt with is neither linear nor scalar. From the possible connections, the student's curiosity and needs will establish the path she/he will use to read, understand, and solve the problem.

THE CHOICE OF THE *CARBOPOLIS* COMPUTATIONAL IMPLEMENTATION

The National and Municipal Policies for Educational Computing

In Brazil, two programs have guided the political-pedagogic works related to educational computing. From the *Programa Nacional de Informática na Educação (ProInfo)* (http://proinfo.gov.br) of the Brazilian Department of Education, courses in educational computing are offered to teachers in the school network and computer laboratories are installed in public schools throughout Brazil. In its turn, the *Programa Sociedade da Informação (Information Society Program – SocInfo)* (http://www.socinfo.org.br) of the Brazilian Ministry of Science and Technology develops a multi-annual plan. This plan has, as a goal, the public appropriation of microcomputing development in order to meet the special needs of education,

The Green Book of the *Information Society Program* (Takahashi, 2000) establishes a summary of the possible applications of information and communication technologies and has, as a goal, the implementation of appropriate programs. Relative to education, the authors of the Green Book understand that these technologies can greatly contribute to the efficiency of educational programs; therefore, they can affect a larger number of communities and regions. Furthermore, they understand that educators' pedagogical and technological abilities need to be in parallel with the development of learning materials written in Portuguese. Related to this development, they have recommended some policies. Examples include: a) identification and provision of free software in order to create areas of interest, as well as other specific uses, in didactic activities in all levels of all fields; and b) creation and provision of free learning materials directed to the information and communication technologies, as well as their impact on society.

The free software policy is also important in the educational computing field in Porto Alegre, the capital city of a southern state of Brazil. In this city, there is a project called *Informática na Educação: uma rede para inclusão digital*. Information on this project can be found at: http://www.pro-

cempa.com.br. Through this project all the schools in the city's educational network were computerized; these computers include the GNU-Linux operating system, StarOffice, e-mail, and free access to the Internet. The computerized environments were established as part of the pedagogical policy of the city and they are used by different disciplines in order to develop their subjects.

Thus, the technologies of the *Carbopolis* software implementation were chosen with national and municipal public policies as a basis.

Some Technologies in the Implementation of Pedagogical Projects for Science Education

There are several types of technologies that can be used in the implementation of a pedagogical project such as *Carbopolis*. One of these is *remote exploratoriums* suggested by Ambach, Corrina, & Repenning (1995). With a browser the user can access the exploratoriums, which are, according to these authors, computerized simulations of the experiences found in modern interactive science museums. In such exploratoriums, the user can interact with the simulation by testing the validity of the model presented, by modifying parameters, or by including new variables, but always preserving the integrity of the original exploratorium. However, these remote exploratoriums were developed using a technology called Agentsheets (http://agentsheets.com), which is not free.

However, in 1995 the authors cited above claimed that their learning environments used as many Internet resources as possible. Yet, in the educational computing field, the progress in technology and the adding of other media that are relatively more efficient became evident within few years. For example, Warner, Catterall, Gregory & Lipson (2000), describe the Simscience, which is a project that, through interactive educational modules, tries to show to the general public, not just the importance of but also the objectives of, research that uses supercomputers to perform large scale simulations. The modules, which illustrate the simple concepts underlying the main topic, deal with the topics of their research: crack growth, noise, fluid flow, and membranes. Each one of these themes takes different approaches, and they might include some simulations using Java or video clips.

Nowadays, it seems that the current literature encourages the development of a group of network-based generic simulation programs, which offer maximum interfaces to the user and which are independent of platforms (Veith, Kobza, & Koelling, 1999). Thus, the most promising technologies

would be based on virtual machines, meta-languages, and open Internet standards (Christian, 2000).

That is the reason the Java language has been used in the creation of educational software. Besides the examples listed so far, it is proper to mention the use of this technology in the development of software that seeks to: 1) teach some Physics content that usually is not part of the traditional higher-education curriculum, such as chaotic systems, fractals, and liquid crystals (Tobochnik, 1999); and 2) teach process simulation (Miller, Seila, & Xiang, 2000). From the approaches of the authors mentioned above, one can notice two tendencies toward the use of network simulations: in higher education and in the development of academic projects. It means that the inclusion of this type of technology in secondary education has not been done nor put into plan yet.

The Java Technology

In the past few years, Java has been considered one of the most significant advances in computer software technology. Such an advance is as important as the advance of HTML technology was for the success of static page editing in the Internet.

This new technology has caused so many changes that Java has been called the second Internet revolution (Ritchey, 1996), especially because it was created to fill a need in the real world even before its first program had been run. Such needs can be simplified into a list of commercial requirements, according to one of the creators (Naughton, 1996).

In other words, this technology, in its origin, should be: simple and powerful, safe, object-oriented, robust, interactive, neutral in relation to the architecture, easy to learn and interpreted, and be high-performance. Thus, Java is considered an alternative (Naughton, 1996) as far as an evolution of the C++ language (Ritchey, 1996).

The Java Language was carefully adapted for its environment: high-performance, distributed computing in heterogeneous systems (which practically translate into what the Internet is). Furthermore, Java represents a dynamically extensible system, which can incorporate files from computer hard discs, files stored in local network computers, or files in systems located anywhere in the world through the use of the Internet (Ritchey, 1996). Java programming is object-oriented. Objects represent things, either real or abstract, that exist in the real world and are represented by one or more classes in Java. Classes are groups of code that model the behavior of the objects

within the software. It means that programmers can spend less time in code execution, directing their efforts toward the development of functionality (Ritchey, 1996). In this way, a well-planned class hierarchy is the basis for the reuse of code that requires less time for development and testing. The encapsulation of data and code makes more rapid implementation possible without having to understand the interfaces of the entire system. (Naughton, 1996). Because of these characteristics, Java has been considered "an explosion of interactive content" through the Internet (Naughton, 1996). The technology makes possible a certain way of thinking that is totally new to distributed computing. According to Ritchey (1996), until now the information presented within the Internet's pages was completely static; everything had previously been developed. The users were essentially consulting electronic versions of printed documents. However, Ritchey understands that computers exist to provide *dynamic interaction*, in which the users are given *immediate return* and the programs perform actions by themselves. Thus, he points out that the Java language makes such functionality possible, allowing the execution of code that can be distributed via the Internet using a robust, safe, and high-performance environment.

Yet, infrastructure is another problem. Recently, Cates (2001) pointed out that even in the United States, the potential of the pedagogical project still requires technological progress, such as the increase of broadband technology. Related to this, there could be a temporary solution, which is the development of an off-line version of the program. It would be on the Internet, but would be downloaded to the user's computer. The connection to the Internet would occur only at the beginning and at the end of the session when the exchange of information with the system's central database was necessary, or even when some communications routine with other users was activated.

This type of off-line system does not make the ease of updating and expansion, which are important characteristics of Web programs, impracticable (Starr, 1997). Whenever a user runs the program, messages are exchanged with the server in the beginning and/or at the end of the session. Thus, when there is something new related to the program, the path of communication will itself be used to tell the user about any updating and expansions that have occurred. Moreover, we understand that the many possibilities for updating and expanding computing systems, especially those systems oriented to education, are maximized when the concept of free software is incorporated into them.

The Concept of Free Software

A program is considered free software (Stallman, 1998) when a specific user may: 1) execute the program for any purpose; 2) modify the program and adapt it to his/her needs (to be able to do this it is necessary to have the source code) 3) redistribute copies for free or not; and 4) distribute modified versions of the program, so that the community can benefit from program improvements.

The Java development environment that was used in the *Carbopolis* software implementation (http://www.java.com) is available free from Sun Microsystems. According to one of its creators (Naughton, 1996), the notion of a language that creates programs for all computers at the same time is a clear threat to all systems manufacturers who depend on the interfaces in order to guarantee their place in the marketplace.

In addition, Stallman (2000) points out the idea that the social system of payware (a type of system that does not allow users to modify nor share the software) is not just antisocial, but also not ethical. Since the author is aware of the impact of such an opinion, he asks: "What more could be said about a system based on dividing the public and keeping users abandoned?"

Finally, it is important to emphasize, according to Christian (2000), that the present challenge in using Java technology is to develop representations of scientific interests in a significant pedagogically manner (Christian 2000).

THE PROPOSED DESIGN COMPUTATIONAL IMPLEMENTATION

The Graphical Interface

The graphical interface and the user's events manipulation were made with an interface system called Swing. This system integrates the JFC (Java Foundation Classes), which is totally based on Java (JavaBeans); it does not make calls to the native code of the operating system. It strengthens the proposal of independent platforms made by the creators of the language.

The login screen in the program, which was created with this interface, can be seen in Figure 1. In this screen there is a dialog box where the user can select his/her name from a list that has been dynamically loaded from the database; the user can then type his/her password. If the user is not registered, he/she can type a new name and a new password. The password is saved in the database, as is information about use of the software.

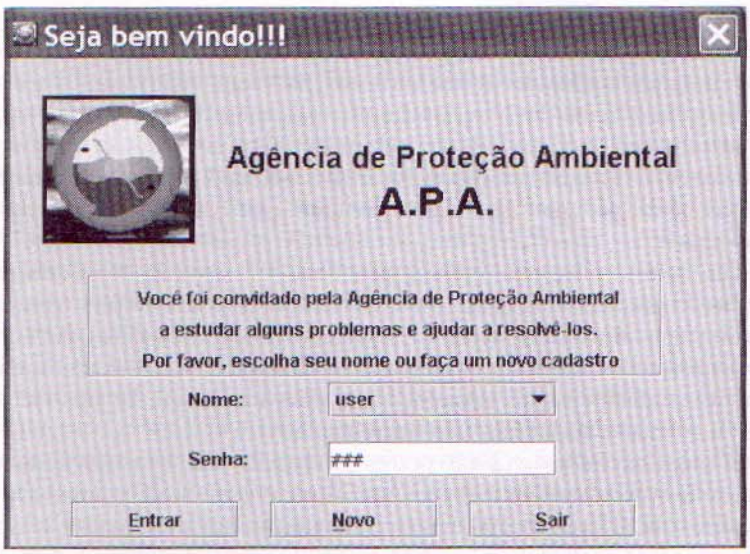

Figure 1. The login screen of the ***Carbopolis*** software, where the user can register or type her/his name.

The following section of the code shows how users' names are loaded into the database from the combo box in the software input window.

Listing 1

```
public final synchronized void loadComboItems()
{
    Thread runner = new Thread(new Runnable(){
    public void run() {
        DataBase dataBase = new DataBase();
        dataBase.connect();
        if (!dataBase.isConnected()) {
            JOptionPane.showMessageDialog(getParent
            Frame(), "Não foi possível estabelecer
            uma conexão com o banco de dados.");
            return;
        }
        Vector row = dataBase.select("SELECT NOME
        FROM USUARIO ORDER BY NOME");
        if (row == null || row.isEmpty()) {
            dataBase.disconnect();
            return;
```

```
            }
            else {
                for (int i = 0; i < row.size(); i++) {
                    String name = (String)row.
elementAt(i);
                    addName(name);
            }
        }
        dataBase.disconnect();
        }});
        runner.start();
            }
```

After the user's identification has been verified, the main screen of the program is displayed. It can be seen in Figure 2.

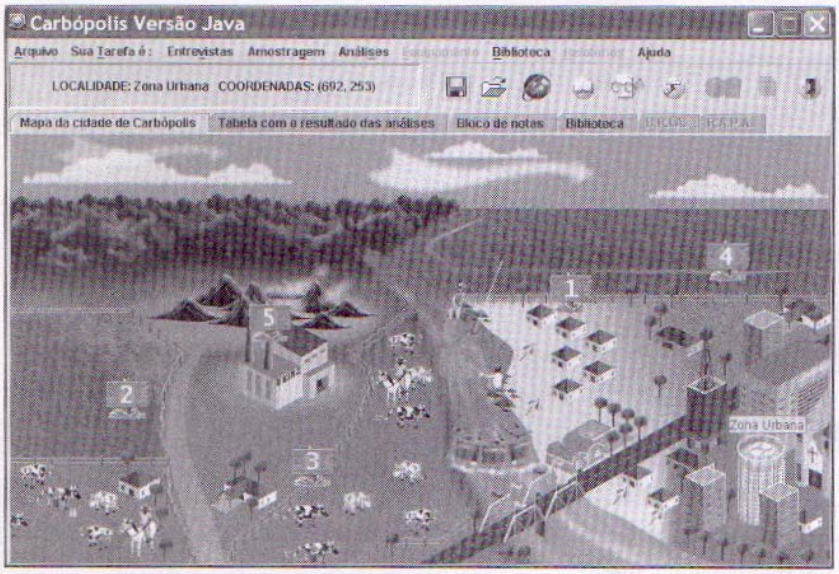

Figure 2. The main screen of *Carbopolis* software

The Scenario Mapping

The whole application is based on information associated with the scenario that illustrates and defines the problem presented in *Carbopolis*. It was necessary to map the pictures so that the program could keep control of the

actions made by the users with the mouse (for example, interviews and samplers) in the different parts of the scenario. The mapping was made by using the Cartesian-coordinate system, assuming a horizontal X-axis and a vertical Y-axis.

Then, it was decided to develop an abstract class called "AreaMapeada." This class, which must be inherited by all the classes which represent the regions, makes all the classes follow a certain procedure.

Figure 3 shows different regions depicted in the scenario. In Figure 3(a), areas marked with letters refer to the interviews with people who live in the region. The interviews are made with: a) the owner of the farm called *Primor;* b) the ranger; c) the thermo-electrical power plant public relations employee; d) the owner of the barn called *Soya;* e) the owner of the farm called *Vacaria;* f) the mayor of *Carbopolis;* and g) the coal miner. In Figure 3(b)), the areas marked with numbers refer to the air pollutant concentrations in different areas of the scenario.

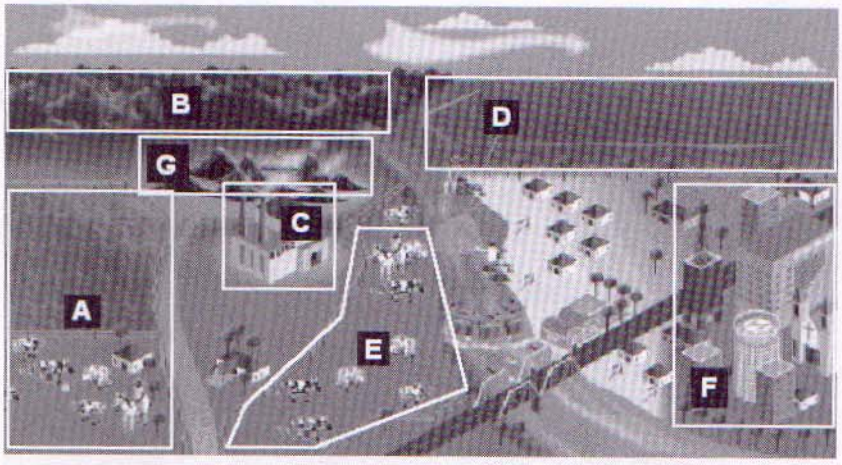

Figure 3(a). Regions of the map where interviews with people who live in the region can be made.

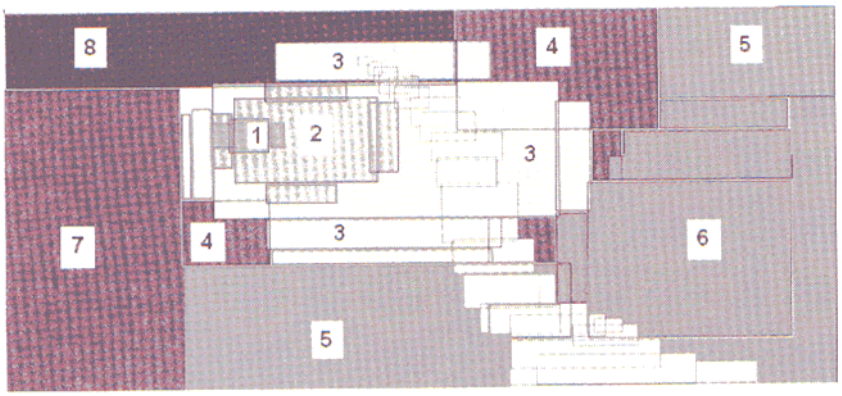

Figure 3(b). Masks of the equi-pollutant curves, indicating the different air pollutant concentrations whose values are in Table 1.

The air pollutant concentrations are collected in different equi-pollutant regions. The air pollutants are sulphur dioxide (SO_2), nitrogen oxides (NO_x), particulate matter (PM), carbon monoxide (CO), hydrocarbons (HC), and pH. Their values, shown in Table 1, were adapted from a case of environmental pollution by a coal electric power station (Fiedler, Martins, & Solari, 1990). According to the international standards for air pollutant concentrations, these values are higher than normal (Caselli, 1992; Serrano, Rodriguez, & van der Goes, 1993).

Table 1

Air pollutant concentrations in the different equi-pollutant curves

	Air Pollutants (values in mgAir Pollutants (values in mg/m^3)					
CURVES	SO_2	NO_x	PM	CO	HC	pH
1	1,83	0,16	0,10	4,36	0,01	3,80
2	1,70	0,14	0,10	3,54	0,01	4,00
3	1,61	0,12	0,08	2,87	0,01	4,10
4	1,44	0,07	0,08	1,43	0,01	4,40
5	1,40	0,06	0,07	0,76	0,01	4,70
6	1,17	0,26	0,10	5,76	0,02	5,00
7	0,09	0,05	0,30	0,45	0,01	5,60
8	0,06	0,04	0,01	0,14	0,01	5,40

The different equi-pollutant regions are described by several polygons as Figure 3(b) shows. These polygons are represented by X and Y coordinates, returning *true* when in the mapping area and *false* when out of this area.

Interviews

The interviews are presented in dialog boxes that include a picture of the character (.gif) and a text area that is filled with the content from a text file that has been dynamically loaded. Figure 4 shows one of these dialog boxes.

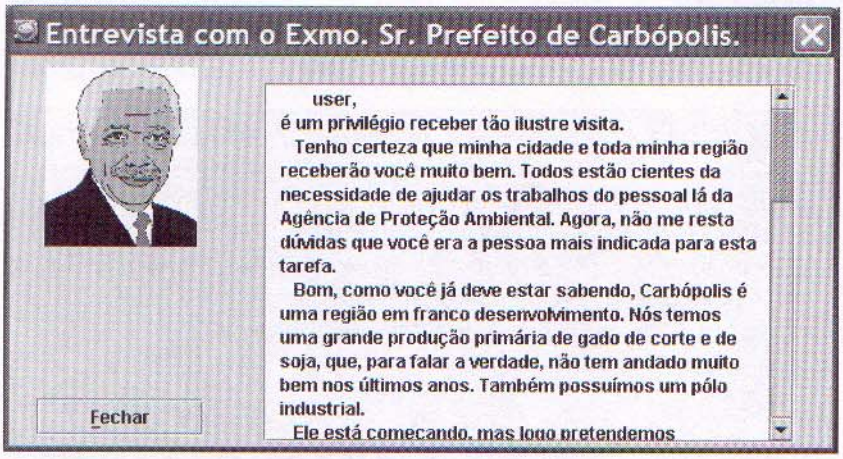

Figure 4. A dialog box showing the declarations of the Carbopolis mayor

In order to implement the interviews, a class called "Mensagem" was created. It works for the interviews and for the messages of the Agency called "Agência de Protenção Ambiental" (the user works for this agency that is asking that user to solve the problem that has afflicted the people who work with agriculture and cattle in *Carbopolis*). Two *strings* that work as constructor parameters are given to this class. The first represents the name of the text file and the second represents the name of the image file. From this, the classes that implement the interviews were created. Hence, all the basic interfaces were created.

When the user clicks on a region of the map, a *popup* menu appears and the user can select the option "Fazer Entrevista." Then the interview is

shown. A part of the reference code related to the usage of the mouse in some regions of the map is shown below.

Listing 2

```
    class MouseControl implements MouseListener {

    JComponent componente;
    JPopupMenu popup;
    PopupControl cPopup;
    public MouseControl(JComponent componente,
    JPopupMenu popup,          PopupControl cPopup)
    {
    this.componente = componente;
    this.popup = popup;
    this.cPopup = cPopup;
}
public void mousePressed(MouseEvent e) {}
public void mouseReleased(MouseEvent e) {
    int auxX, auxY;
    int x = e.getX();
    int y = e.getY();
    auxX=x;
    auxY=y;
    if (e.isPopupTrigger())
    {
       popup.show(componente, auxX, auxY);
    }
    auxX=x;
    auxY=y;
    cPopup.informaCoordenadas(auxX, auxY);
}
public void mouseClicked(MouseEvent e) {}
public void mouseEntered(MouseEvent e) {}
public void mouseExited(MouseEvent e)  {}
}
```

Data Storage

Java Database

The JDBC (Java Database Connectivity) is an API (Application Program Interface) that is defined by Sun Microsystems. It makes it possible to

establish communications between a Java application and a database server. In other words, the JDBC allows the establishment of a connection with the database, sending SQL (Structured Query Language) instructions and processing the results received.

A driver manager and API are available from Sun because of the need for maintaining compatibility between Java and several DBMS (Database Management Systems). The database manufacturer must develop drivers for his/her specific database managers, implementing JDBC interfaces according to certain specifications.

Drivers, of which there are four types, are the main part of the JDBC. Each is based on different technologies; they provide groups of unique characteristics that are recommended for the development of specific kinds of applications.

Type 1 – JDBC-ODBC: It works like a bridge between JDBC and an ODBC driver. The installation of the ODBC driver is necessary in each client server in which the application is installed.

Type 2 – Native-API partly Java technology-enabled driver: It converts JDBC calls into API incoming calls to the database client. The native methods must be in the machine in which the application is installed.

Type 3 – Pure Java Driver for Database Middleware: The API of the JDBC access is a middleware application, which translates JDBC calls and sends them to the database. The advantage of this driver is that it can be used either in the Internet or in Intranets. This is the most flexible kind, since the driver server layer can be implemented to access several DBMS. This simplifies the migration among databases.

Type 4 – Direct-to-Database Pure Java Driver: This driver converts the JDBC calls directly into the DBMS native protocol. With this kind of driver, the client application can access a server application in a remote and native way without any intermediary action. This is the simplest driver to install and configure. It is only necessary copy a *.jar* file or a group of classes into a directory in the *classpath*.

The choice of the DBMS

Due to the need to maintain a simple application for the user, the developers have opted for a DBMS that uses a Type 4 driver because this type does not require the user to install or configure the driver.

In addition, it was considered necessary to provide a no-cost database system to the user and to the developer. Thus, MySQL (www.mysql.com) was chosen. MySQL is a powerful and free database manager that is available in several platforms and is easy to install. These were important criteria in selecting a database manager suitable to *Carbopolis*.

Using the database in the Carbopolis software

The entire user's information, as well as his/her actions while the program is running, is saved in the database. Thus, a class called "DataBase" was created. This class implements the majority of the database manipulation functions using methods such as *select(), insert(), update(), delete()*, etc.

Log files were created in order to keep a copy of the user's actions while the program is running. These files were created in a directory called "user/UserName/UserName.txt." *UserName* is the name the user enters when the application starts. Consequently, this name will be the user's name as registered in the database. The class called *RamdomAccessFile,* from *java.io,* which is provided in the Java Development Kit (JDK), was used to save the file mentioned above. .

The implementation of the save routine was made in a class called "LogFile." This class uses a method called "*saveAction().*" This method is given, as a parameter, a string that saves the action of the current user in the *log* file mentioned above. Thus, using these two classes, the database and *log* file functionalities were implemented.

A part of the code that describes this is shown below.

Listing 3

```
Method saveAction():
        public static void saveAction(final
    String userAction, final String userName)
    {
        Thread runner = new Thread(new Runna-
    ble() {
        public void run()
        {
            try {
                LogFile log    = new
    LogFile(userName);
                String path    = log.getLogPath()
                + "/" + log.getUserName() +
                ".log";
                Locale locale = Locale.getDe-
                fault();
                SimpleDateFormat formatter =
                new SimpleDateFormat("dd.MM.yyyy
                ':' H:mm:ss zzzz", locale);
                String date = formatter.
```

```
                format(new Date());
                RandomAccessFile random = new
                RandomAccessFile(path, "rw");
                random.seek(random.length());
                random.writeBytes("\n" + userAc-
                tion + " [" + date + "]");
                random.close();
            } catch (Exception e) {
e.printStackTrace(); }
        }
        });
    }
```

When the user clicks on the map and he/she has an interview, for example, with the miner of the city *Carbopolis,* the call by the method "*saveAction()*" returns:

```
log.saveAction ("Fez entrevista com mineiro.");
```

Each text, interview, or request made to the program's *help* is registered in the *log*, which is saved in the directory of each registered user. As can be seen in the code shown above, the *log* also collects information about date and time of each user-action. Later, the *log* files can be used to evaluate a particular user's usage of the program or of the paths followed to solve the proposed problem. It indicates the line of reasoning that user followed.

Samplers

Another functionality added to the program was the addition and removal of *samplers* identified by numbers. Figure 2 presents some samplers on the *Carbopolis* map.

The samplers are added and removed through a PopupMenu, which is activated with a click of the right mouse button. When the user puts a sampler on a region of the map, a small picture (.gif) is placed exactly at the position (Point P= (x,y)) of the mouse click. Thus, when the user saves the changes, the data of the samplers that were added are saved in a database table. In the same way, when the user removes the sampler, the picture is removed from the map and from the database.

Part of the code, indicating how the samplers are added to the map, the selection of the sampler number, and the verification of the maximum number of samplers that it is possible to install in the program, are shown below.

Listing 4

```
        public void addAmostrador(Amostrador amtIn-
stance) {

        int x = amtInstance.getX();
        int y = amtInstance.getY();
        int auxX=x;
        int auxY=y;
        int a = amtInstance.getArea();

        addAmostrador(auxX, auxY, a);
    }

    private void addAmostrador(int x, int y, int
a) {

        Amostrador amtGlobalInstance = new
Amostrador(x, y, a);
        amtAux.clear();
        int proximoId =  0;
        if ((amostradores.size() + 1) <= 20) {
            for (int i = 0; i < amostradores.size();
i++) {
                Amostrador amtInnerInstance = (Amos-
                trador) amostradores.elementAt(i);
                String ident = String.
                valueOf(amtInnerInstance.getID());
                amtAux.add(ident);
            }
            if (amtGlobalInstance.equals(amtInnerIns
tance))
            return;
            for (int i = 0; i <= amostradores.
size(); i++) {
                if (amtAux.indexOf(String.
valueOf(i+1)) == -1)
                {
                    proximoId = i+1;
                    break;
                }

            amtGlobalInstance.setID(proximoId);

        amostradores.addElement(amtGlobalInstance);
            insertAmostrador(amtGlobalInstance);
```

```
                updateGraphicallyAmostrator();
            } else {
                        JOptionPane.
                        showMessageDialog(this, "Você
                        pode colocar no máximo " +
                        (Amostrador.MAX - 1) + "
                        amostradores");
                        return;
            }
    }
```

The bookmark and the notepad

A bookmark and a notepad were also developed. The notepad was implemented by using a dialog box, which has a *JTextArea*, allowing the user to take notes and save them while the program is running. This notepad, which is activated when the user clicks on the icon in the toolbar or on its respective tab control *(JTabbedPane)*, has almost all the functions of the Windows Notepad.

The bookmark was developed by the use of a dialog box and an object called *List*. The Internet addresses (URLs) are added and removed from the object *List* through objects called *JButtons*. The program that was utilized to visualize Internet pages, which has been called *Browser*, is activated when the user double-clicks on the item that represents the desired URL in the List. In other words, it is called the current browser of the operating system.

The bookmark and the notepad contents are also stored in the database.

Browser as a class within Carbopolis

The HTML technology was used to implement the hypertext of the *Carbopolis* library. This hypertext is poly-hierarchic and very extensive. It includes 56 topics and more than 340 active connections. The view of the HTML files used by the program is made from a class called *Browser*. This class extends JScrollPane and it implements a *HyperLinkListener*. This class makes it possible to show the HTML text and implements the bookmark hypertext according to the code below.

Listing 5

```
    public Browser(CarbopolisApp frame, String
initialPage) {
    try {
            File f = new File (HELP_PATH + initial-
```

```
Page + ".htm");
            String s = "file:" + f.getAbsolutePath();
            URL url = new URL(s);
            addPage(url);
        } catch (MalformedURLException e) {
            System.out.println("Malformed URL: " + e);
        }

        html = new JEditorPane(s);
        html.setEditable(false);
        html.addHyperlinkListener(this);

        JViewport vp = getViewport();
        vp.add(html);
        } catch (IOException e) {
            System.out.println("IOException: " + e);

        }

}
```

There is a class called *UIBrowser*, which extends a JFrame. It makes the logical browser implementation. This class has some specific functionality in order to deal with the HTML text that was shown. In the *Carbopolis* software, the text opened by the program is shown in a special tab control (JTabbedPane) and not in an extra window, as is common. Hence, the functionalities implemented are: to return, to advance, and to reload the page that was shown and the main index. All the functionalities are managed by buttons.

The method of the UIBrowser class, which shows the HTML text, is shown below.

Listing 6

```
    public UIBrowser(CarbopolisApp frame, boolean
inFrame, String index){
    super("Biblioteca");
    super.setIconImage(getToolkit().
    getImage("imagens/gif/simbapa.gif"));
    browser = new Browser(frame, index);
    setContentPane(createLayout());
    setSize(600, 400);

    Dimension dim = Toolkit.getDefaultToolkit().
getScreenSize();
```

```
        int w = getSize().width;
        int h = getSize().height;
        int x = (dim.width - w)  / 2;
        int y = (dim.height - h) / 2;

        setBounds(x, 0, w, h);
        setVisible(true);

    }
```

The part of the code that opens the HTML text in a tab control of the program is presented below.

Listing 7

```
    getParentFrame().tabbedPane.setComponentAt(3,
(bBrowser=new  UIBrowser(getParentFrame(),  false,
"amos1",false)).createLayout());
getParentFrame().tabbedPane.setSelectedIndex(3);
```

Installing Carbopolis

In order to run a Java program, it is necessary to at least install the JRE (Java Runtime Environment), i.e., the group of essential classes to run any Java application in a specific operating system. Furthermore, the correct configuration of the classpath is also necessary. Only then can the application run through the prompt command of the operating system.

For example, if a high school professor wants to use *Carbopolis* in the school laboratory he/she should, *a priori,* download the program, and install and configure the JRE. Next, he/she should download and install MySQL. Only then can the professor download and install *Carbopolis* and run it by the prompt command. Thus, a certain degree of technical knowledge of science and computers is necessary. Hence, *Carbopolis* goals are: free software, no-fee software (professional technical training needs time and investment), and user-friendly software.

Thus, to achieve these goals, it was necessary to search for an installer that was freeware and that simplified the installation and the use of *Carbopolis. InstallAnywhere Now!* (www.zerog.com) satisfied these criteria. This installer allows the creation of another installer for all platforms, a shortcut to run the program (just click), and an uninstaller for the program. It frees the user from more complex procedures. In order to run *Carbopolis* it would only be necessary to follow these steps: 1) install and run MySQL; 2) download *Carbopolis* for the operating system that was chosen; 3) click on the

installer and follow the installation instruction; 4) start the DBMS and click on the *Carbopolis* icon.

EVALUATION

During the implementation of the *Java* version of the *Carbopolis* software, two evaluation activities of the pedagogical project of the software were developed through the use of a demo version, which was implemented in *Delphi*. These were: (1) the contexts of usability in *Carbopolis* and (2) the possibility of conceptual learning.

One of the activities involved professors of different disciplines and from different schools (Guterres, Eichler, & Del Pino, 2003). In this sense, four courses to publicize the software to 30 professors in public and private schools were carried out. Later, professors who had used the software in any aspect of their work, either in school or at a university, were interviewed. The intent of these interviews was to obtain information about how the professors introduced the software in their pedagogical proposal, i.e., checking what Squires and McDougall (1994) call *software usability*. In addition, opinions and suggestions on improvement to the software were identified.

The professors who were interviewed see computing as a tool that can support and diversify their work in the classroom. The fact that *Carbopolis* is free software was praised, since one of the criteria in searching for educational software on the Internet is that the software should be free. The interdisciplinary features were also mentioned, since these features make it possible to use the software in projects involving professors from several different fields (e.g., Biology and Geography); this is a requirement in many elementary and junior schools. Regarding the context of usability, it was found that the *Carbopolis* demo version was employed in teacher training courses, in teaching programs for youth and adults, in elementary, junior, and high schools, in technical schools, and at universities (in courses such as Geology, Environmental Engineering, and Biology). The main critiques were related to the difficulty in using the software with first grade students, as well as to the amount and complexity of the information in the hypertext of the software.

Another stage of the research about the usability of *Carbopolis* involved the observation of two didactical transpositions, which were made by the professors. In this stage, the introductory planning, the use of the activity, and the evaluation of the results of these activities, were observed. The transposition, carried out by a professor in a municipal public school,

involved the use of a small part of the software: the collection and analysis of air and rainwater samples. According to that teacher, the activity was intended to motivate the students to think about relevant issues regarding environmental problems. It not only stimulated their interest in Land Sciences but also motivated their desire to stay in school. In that school a large number of students had given up their studies; the teacher believes that educational computing can stimulate students to remain in the school. From a conceptual point of view, this activity was intended to reinforce the ideas of oxides, acid, and base reactions, which they had already been taught. When the students finished the activity, they were surprised with the way the contents appeared in the software: *"I thought it was different because here* (in the school) *you see things like this: "You do this, you do that... There* (in the software) *I had an objective for using it! (...) You knew the reason why you were doing that!"*

The didactical transposition made by two private school professors was much more complex, since it joined high school disciplines of chemistry, biology, history, and geography in a 40-hour extra-class project, in which 28 hours were spent in a computer laboratory, and 12 hours in fieldwork. The work was developed in 14 two-hour meetings in the computer laboratory, where the students analysed the problems related to the atmospheric pollution and tried to find solutions. After completing the activity in the computer laboratory, a meeting with the students was held in order to prepare them for the fieldwork. In their fieldwork they visited the Department of Air Quality Management, the Municipal Bureau of Environment Protection in Porto Alegre (RS, Brazil), the Laboratory of Analysis of Water Quality, the Municipal Department of Water and Sanitation, coal mines in Arroio dos Ratos (RS, Brazil), and a thermo-electrical power plant in Butiá (RS, Brazil).

During the fieldwork activities, it was observed that the students were able to integrate the knowledge they had developed in the use of the software, correlating the information about coal characteristics and electrical conversion processes with the environmental standards regulated by the law and observed by the environmental agencies. The professors said they were quite satisfied with the results and that after the visits, "*Carbopolis became real and not virtual anymore.*"

As mentioned in the beginning of this section, the second context of the *Carbopolis* software was related to conceptual learning. This evaluation is part of a Master's thesis, which intends to show the cognitive conduct manifested by exemplar subjects during an environmental analysis of a computer-simulation problem about atmospheric pollution. (Eichler, Del Pino, & Fagundes, 2004)

Regarding the conceptual domain considered in this research, it is possible to arrive at some conclusions. In the research, some thoughts pointing to notions of the different systems related to environmental topics (for example, judicial, economical, physical-chemical, and ecological systems) were expressed, although any subject manifested the integration of all those systems. Thus, it could be said that, in relation to the environmental issues there are subjects with different profiles. There are those who are guided by economics and those who are guided by the law. Others follow the description and explanations of the natural science models. There are also those who campaign and understand that the publicizing of information leads to a change in attitudes. So, the profiles are many and quite different.

Finally, it was possible to observe that the *Carbopolis* software allowed subjects with different profiles, who used different paths, to rebuild the causal nexus of the simulated problem and then solve it.

CONCLUSIONS

This article had two main goals. The first one was to describe some characteristics of the *Carbopolis* software pedagogical project, and the second to present some programming solutions from the computational implementation of this project.

As for the first goal, some pedagogical considerations were made that related to the development of the computerized learning material project for scientific education. The design of scenarios was presented as a way of integrating didactic strategies of simulation, of problem solving, and of play. Finally, we have tried to show this strategic union of the *Carbopolis* software within the project.

Related to the second goal, some potential for using the Java technology in the computational implementation of pedagogical projects with such characteristics was briefly described. We have tried to show the use of the *Carbopolis* computational implementation under the concept of free software, due to the national and municipal policies relating to the educational computing field. For this, some developed programming solutions were presented. These solutions involved the graphical interface, scenario mapping, data storage, bookmarking, the notepad, a browser specific to the HTML hypertext, and the software installation.

The *Carbopolis* software is the first product of a project for the development of a virtual learning environment for science education (Eichler, Gonçalves, Silva, Junges, & Del Pino, 2003). The means of electric energy

production and its environmental and social impacts are used as generator themes in teaching and learning activities that are anticipated for use in several products, which will be developed as part of this project. Hence, the programming solutions, as well as the Java classes project presented here, will be used later in the computational implementation of other educational software that have similar pedagogical projects.

Aknowledgement

The authors would like to thank the Research Support Foundation of the Rio Grande do Sul State (FAPERGS) for its financial support, The National Council for Scientific and Technological Development (CNPq) for the scholarships that it granted, and the Data Processing Company of Porto Alegre City (PROCEMPA) for providing the trainees who implemented the software.

References

Ambach, J., Corrina, P. & Repenning, A. (1995). Remote exploratoriums: combining network media and design environments. *Computers & Education, 24*(3), 163-176.

American Chemical Society (1993). *Chemistry in the community*. Duduque – Iowa: Kendall/Hunt.

Boutinet, J.P. (1990). *Anthropologie du project*. Paris: PUF.

Calza, R.E. & Meade, J.T. (1998) The GenTechnique project: developing an open environment for learning molecular genetics. *Computer & Education, 30* (1/2), 117-123.

Carroll, J.M. (2000). Five reasons for scenario-based design. *Interacting with Computers, 13*, 43-60.

Caselli, M. (1992). *La Contaminación Atmosférica*. México: Siglo Vientiuno.

Cates, W.M. (2001). Introduction to special issue. *Educational Technology, 41* (1), 5-6.

Christian, W. (2000). Java programming and Internet technologies for undergraduate education. *Computer Physics Communications, 127*, 16–22.

Duveen, J. & Solomon, J. (1994). The great evolution trial: use of role-play in the classroom. *Journal of Research in Science Teaching, 31* (5), 575-582.

Eichler, M.L. & Del Pino, J.C. (2000). Carbopolis, um software para educação química. *Química Nova na Escola, 11*, 10-12.

Eichler, M.L., Del Pino, J.C., & Fagundes, L.C. (2004). Development of cognitive conducts during a computer simulated environmental analysis. *Chemistry Education: Research and Practice, 5* (2), 157-174.

Eichler, M.L., Gonçalves, M.R., Silva, F.O.M., Junges, F., & Del Pino, J.C. (2003). Virtual learning environments in Brazil. *Educational Technology, 43* (6), 58-60.

El-Tigi, M. & Branch, R.M. (1997). Designing for interaction, learner control, and feedback during web-based learning. *Educational Technology, 37* (3), 23-29.

Farynaiarz, J.V. & Lockwood, L.G. (1992). Effectiveness of microcomputer simulations in stimulating environmental problem solving by community college students. *Journal of Research in Science Education, 29* (5), 453-470.

Fiedler, H., Martins, A.F., & Solari, J.A. (1990). Meio ambiente e complexos carboelétricos: o caso Candiota. *Ciência Hoje, 12* (68), 38-45.

Freire, P. (1996). *Pedagogia da autonomia: saberes necessários à prática educativa*. Paz e Terra, São Paulo.

Guterres, J.O., Eichler, M.L., & Del Pino, J.C. (2003). Compreensões de professores sobre o software educativo Carbópolis e sua utilização em diferentes realidades de escola. *Revista Brasileira de Informática na Educação, 11* (2), 86-99.

Laurillard, D. (1992). Learning through collaborative computer simulations. *British Journal of Educational Technology, 23* (3), 164-171.

Lin, X.., Bransford, J.D., Hmelo, C.E., Kantor, R.J., Hickey, D.T., Secules, T., Petrosino, A.J., & Goldman, S.R. (1995). Instructional design and development of learning communities: an invitation to a dialogue. *Educational Technology, 35* (5), 53-63.

McLaren, P. (2000). *Che Guevara, Paulo Freire, and the pedagogy of revolution*. Lanham (MD): Rowman & Littlefield Publishers.

Miller, J.A., Seila, A.F., & Xiang, X. (2000). The JSIM web-based simulation environment. *Future Generation Computer Systems, 17*, 119–133.

Naughton, P. (1996). *The Java Handbook*. New York: McGraw-Hill.

Otter, M. & Johnson, H. (2000). Lost in hyperspace: metrics and mental models, *Interacting with Computers, 13,* 1-40.

Piaget, J. (1970). *Epistemologie genetique*. Paris: PUF.

Piaget, J. (1974a). *La prise de conscience*. Paris: PUF.

Piaget, J. (1974b). *Réussir et comprendre*. Paris: PUF.

Rieber, L.P. & Matzko, M.J. (2001). Serious design for serious play in physics. *Educational Technology, 41* (1), 14-24.

Rieber, L.P., Smith, M., Al-Ghafry, S., Strickland, B., Chu, G., & Spahi, F. (1996). The role of meaning in interpreting graphical and textual feedback during a computer-based simulation. *Computers & Education, 27* (1), 45-58.

Ritchey, T. (1996). *Programming with Java! Beta 2.0*. Indianapolis: Macmillan Computer Publishing.

Serrano, O.R., Rodriguez, G.P., & van der Goes, T.F. (1993). *Contaminación Atmosférica y Enfermedad Respiratória*. México: Fondo de Cultura Económica.

Silverman, B.G. (1995). Computer supported collaborative learning, *Computers & Education, 25* (3), 81-91.

Squires, D. & McDougall, A. (1994). *Choosing and using educational software: a teacher's guide*. London: The Falmer Press.

Stallman, R. (1998). The GNU Project. [Documento digital em: http://www.gnu.org/gnu/thegnuproject.html].

Starr, R.M. (1997). Delivering instruction on the World Wide Web: overview and basic design principles. *Educational Technology, 37* (3), 7-14.

Sun, C. & Chou, C. (1996). Experiencing CORAL: design and implementation of distant cooperative learning. *IEEE Transactions on Education, 39* (3), 357-366.

Takahashi, T. (Org.) (2000). Sociedade da Informação no Brasil - Livro Verde. Brasília: Ministério da Ciência e Tecnologia.

Tobochnik, J. (1999). Teaching students to write computer simulations in Java. *Computer Physics Communications, 121–122,* 562–568.

Veith, T.L., Kobza, J.E., & Koelling, C.P. (1999). Netsim: Java-based simulation for the World Wide Web. *Computers & Operations Research, 26,* 607-621

Warner, S., Catterall, S., Gregory, E., & Lipson, E. (2000). SimScience: Interactive educational modules based on large simulations. *Computer Physics Communications, 127,* 1–5.

Whisnant, D.M. (1984). Scientific exploration with a microcomputer: simulations for nonscientists. *Journal of Chemical Education, 61* (7), 627-629.

Whisnant, D.M. (1992). A role-playing exercise using a computer simulation. *Journal of Chemical Education, 69* (1), 42-43.

Zimmerman, R.D., Thomas, R.J., Gan, D., & Murillo-Sanchez, C. (1999). A Web-based platform for experimental investigation of electric power auctions. *Decision Support Systems, 24,* 193–205.

Notes

This paper has been translated from Portuguese by Susana de Azeredo.