DELYS: A novel microworld-based educational software for teaching Computer Science subjects

Vassilios Dagdilelis[1], Georgios Evangelidis[2*], Maya Satratzemi[3], Charalabos Zagouras[4]

[1] *Department of Educational and Social Policy, University of Macedonia, 156, Egnatia str., P.O.Box 1591, 54006, Thessaloniki, Greece,* dagdil@uom.gr

[2]*Department of Applied Informatics, University of Macedonia, 156, Egnatia str., P.O.Box 1591, 54006, Thessaloniki, Greece,* gevan@uom.gr

[3]*Department of Applied Informatics, University of Macedonia, 156, Egnatia str., P.O.Box 1591, 54006, Thessaloniki, Greece,* maya@uom.gr

[4] *Department of Mathematics, University of Patras, Campus (Aristotle str.), Greece,* zagouras@math.upatras.gr

Abstract

The Greek ministry of education with the support of the European Union has begun an attempt to incorporate information and communication technology in the normal, everyday activities of secondary education. Part of this effort is the development of pilot educational software. Our project proposal was DELYS, an educational software environment that aids the teaching of computer science in secondary education. It was one of the approved proposals and, incidentally, the only one funded in the field of computer science. DELYS consists of four "exploratory microworlds" which deal with the following subjects: (i) the various components of a computer and how they operate as a whole, (ii) the booting process in a computer, (iii) the way data is represented numbers are processed, and (iv) a programming environment. The material is mainly presented through the use of simulations and animations and it is accompanied by text and/or video. In this paper, we present the design rationale of the system and its description. In addition, we present initial evaluation results of DELYS with data collected from students and teachers. It must be stated that these first results of DELYS are most encouraging.

[*] Corresponding author

Keywords:

Applications in subject areas, interactive learning environment, multimedia/hypermedia systems, programming and programming languages, secondary education

1. Introduction

In the past few years, extensive efforts have been made at an international level to incorporate information and communications technology (ICT) in education. This is attributed both to the technological advancement of computing systems (better GUIs, development of computer networks, increased computational power, etc.) and the latest findings from extensive research and studies conducted regarding the use of computing systems in the learning process.

Recently, Greece has attempted to incorporate ICT into secondary education with the aim of creating a critical mass of school communities that will embody ICT in their everyday educational activities. The findings and experience gained from this initial phase will enable conclusions to be drawn up as to how ICT can be fully incorporated into all Greek secondary schools. In the next phase primary schools are to join the program. In the pilot phase 300 schools were enlisted in the program and 1000 more are to follow suit.

The project to incorporate ICT into schools is titled *Odysseia*[1] and is co-funded by the European Union. It includes many actions, one of which, called *Nafsica*[2], aims in the development of pilot educational software to support the teaching process of the various subjects of secondary education. Within this context we developed an

[1] Computer Technology Institute. *Project Odysseia*: http://odysseia.cti.gr/English/ODYSSEIANEW/about.html
[2] Computer Technology Institute. *Project Nafsica*: http://odysseia.cti.gr/English/ODYSSEIANEW/e22_naysika/e22_naysika.htm

environment for the introduction of computer science, called DELYS[3]. DELYS is a collection of complementary microworlds, each one concentrating on a specific issue of computer science. DELYS uses extensive animation techniques and has a high degree of interactivity with the user.

Our basic motivation for creating an educational environment to support the teaching of computer science was the fact that students have difficulty in understanding basic theoretical and practical principles about both the hardware and the software of a computing system.

The hardware refers to the operation of a machine, which the teacher is called on to present in the form of static shapes (drawn) on the blackboard and verbal descriptions. This is a source of problems for the students, since many times they are expected to follow the functioning of a machine or the execution of an algorithm through verbal descriptions and simplified diagrams. Added to this, is the students' lack of understanding of the various functions of a computer since it is a machine they are not familiar with.

Even though many students come into contact with computers outside of schools they usually have only an empirical or procedural knowledge of their functioning. This procedural knowledge allows them to use a computer, however, students are ignorant of the basic principles of its operation.

Yet another problem related to the teaching of concepts that have to do with the hardware of a computer is that it is impossible to perform an experiment that actually demonstrates the operation of a computer. Lastly, it must be mentioned that the various chapters in schoolbooks that refer to computer hardware, usually, lack suitable exercises and experiments because of their very nature.

[3] University of Macedonia. *Project DELYS*: http://macedonia.uom.gr/~delys

Another major problem area is the teaching of computer software since:

- it constitutes a specialized environment for solving problems, and
- the solutions required are of a general nature, which high school students are unaccustomed to finding, since they are usually taught to apply given formulas rather than find general solutions.

In the following sections we first describe the general principles that led to the design of DELYS (Sections 2 and 3). Then, we present its functional and didactic features (Section 4) followed by an evaluation of the software (Section 5) and in the final section of the paper we present the conclusions.

2. Software development framework defined by project ODYSSEIA

DELYS was implemented in the context of project *Odysseia* which defines very strict guidelines concerning the characteristics of the software to be implemented. These characteristics refer as much to technical and user interface features as to didactic and pedagogical principles of the software projects to be developed. We briefly present the design principles our software had to adhere to:

Incorporation of the software in school activities: The software should be appropriate for usage in the general educational content as well as in the everyday school activities. It should not be regarded as yet another pilot project to be used in special only situations.

Pedagogical and Didactic principles: The software should encourage teachers and students to become more actively involved in the educational process than they are at present. Modern pedagogical and didactic theories (Piaget, 1974; Brousseau, 1984 ; Balacheff, 1988; Balacheff, 1991) suggest that students cannot learn in isolation; they learn through interaction with their environment, a process that leads to a reconstruction of their knowledge. The teacher, the classmates, and the computer are all parts of this learning environment. The software should not be a simple e-book or tutorial but it should favor the active participation of students. The software should be designed in such a way as to be used within the framework of educational scenarios. These scenarios should be based on the exploratory characteristics of the activities undertaken by students. Wherever possible, the software should have an interdisciplinary nature and combine didactic goals from various subjects in a complementary way.

Supportive educational material: In the school environment the learning process takes place within a framework of didactic situations especially designed by the teacher, where new knowledge (propositions, concepts, methods) arises from solving new problems. Consequently, the software should not been designed for self-learning – although this is not prohibited – rather it should be designed for use within a teaching framework. Therefore the software should be accompanied by a teacher's book that does not describe the functioning characteristics of the software but didactic situations that the software can be used in.

Collaboration with other applications: The software should allow for copying and pasting to and from commonly used applications (text and image editors, word-processors, spreadsheets, etc.). It should also exploit the use of networking technologies encouraging a CMC type of collaborative learning among students.

3. Design rational of DELYS

As has already been stated, computer science is a difficult subject to teach. For this reason, we decided to design and implement a special environment, called DELYS, which can become a supplementary aid to the teacher of computer science in secondary and tertiary education. It is not intended to become the sole means for teaching. With DELYS we tried to create an environment which would support teaching in the best possible way, in other words we designed software which would not only be based exclusively on the state-of-the-art technology, but also on strong didactic analysis. DELYS is an educational environment which has been created taking into consideration the well-known learning difficulties of students.

More precisely, DELYS can:

- Promote collaboration among the students.
- Act as a source of information.
- Act as a virtual laboratory where students can experiment and discover knowledge by themselves, without harming expensive hardware equipment; and.
- Be used as a tool for teaching concepts and operations that are inherently difficult and time-consuming to teach – especially to novice students – using traditional means of teaching (blackboard and verbal descriptions).

Based on the above mentioned didactical principles we designed DELYS as a framework of interactive microworlds. Each microworld was designed and implemented having in mind certain peculiarities that characterize courses of the computer science curriculum and each microworld allows students to explore, seek out information, and eventually solve problems related to the understanding of the way modern computing systems operate. In addition to the high degree of interactivity, the designed microworlds offer two important features: adaptation to the knowledge level of the user and use of animations.

Our microworlds are adaptable to the knowledge level of the student. For example, in the programming microworld, students can choose among a novice level with the programming language X (a Pascal-like language) or move on to an advanced level that includes an assembly language and, for example, observe the contents of the machine registers during the execution of an X program. Thus, the design of our software allows for the teaching of programming principles at various levels of sophistication and complexity depending on the level of students' knowledge.

The implemented microworlds make extensive use of animation techniques, which, in recent years, have been used extensively in educational software (Stasko, Domingue, Brown & Price, 1997). It is true that various studies on the value of animation in the learning process report mixed results; some consider it a positive feature while others a negative one. Despite that, Park & Hopkins (1993) propose certain usages of animated graphics that can contribute to the understanding of a subject and therefore support the learning process. DELYS has incorporated animations mainly as illustrators, i.e., as dynamic visualizations used as an effective aid to represent the structural and functional relations among components in a domain knowledge and as device models for forming mental images, representing system structures and functions that are not directly observable (Park & Hopkins, 1993). As an example, usual programming environments do not offer the option of visualizing the execution of a program. As a result, the process of execution remains hidden and students tend to believe that executing a program has to do with data input and data output. This "black box" approach (du Boulay, O'Shea & Monk, 1989) conceals the semantics of the programming language.

4. Functional Characteristics of the System

We have already mentioned that we base the design of the microworlds on a pedagogical analysis of the comprehension difficulties experienced by students. Each microworld helps students overcome specific difficulties. The implemented microworlds (see Figure 1) are the following:

- "Components of a computer". Students can explore the various components of a modern computer.
- "Booting a computer". This microworld presents the boot process of a computing system using interactive animation.
- "Representing Data & Processing Numbers". This microworld includes various components covering binary representation of integers, addition of binary numbers, ASCII character representation, data storage to and retrieval from the hard disk, and basic logic circuits.
- "The Programming Environment". The didactic objective here is to offer students an elementary programming environment for a simple Pascal-like language and clarify the phases of compilation and program execution that usually constitute a "black box" in professional programming environments.

In addition to the above-mentioned microworlds, DELYS implements a virtual laboratory and a virtual classroom.

- “Virtual Laboratory”. Students can assemble a virtual computer from scratch, find out how peripheral devices and internal components of a computer are attached and/or connected, and use a Virtual Scale to better understand the decimal and binary number representation systems.

- "Virtual Classroom". This environment is accessible from all other microworlds and can help students test their knowledge by answering various questions. The system can provide the correct answers. Another use of this environment is for administering tests consisting of built-in questions or additional questions provided by teachers. For this purpose, we provide a tool that teachers can use to prepare and administer the tests of the Virtual Classroom over the classroom Intranet.

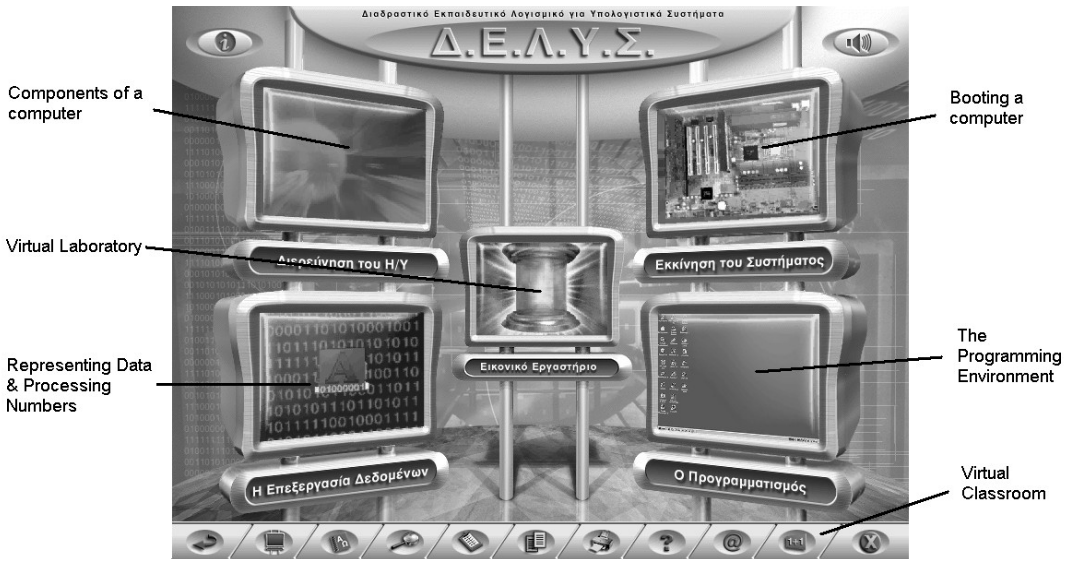


Figure 1. The Main Menu of DELYS

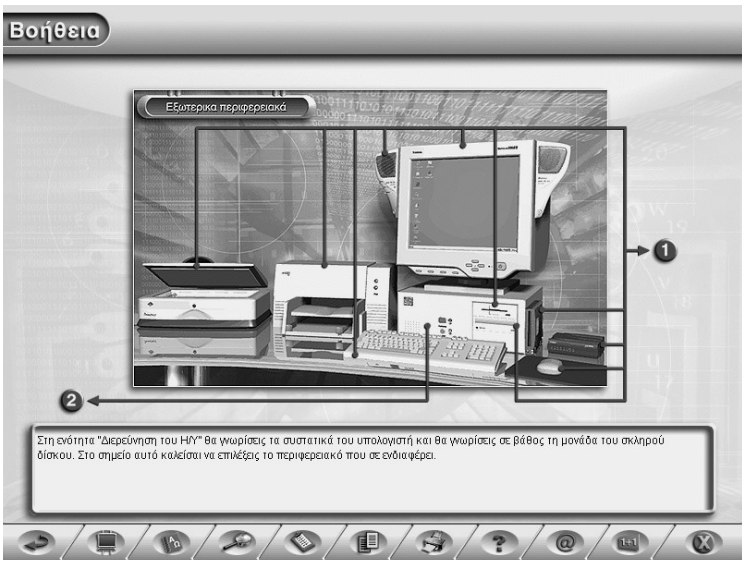
The educational material included in DELYS consists of simulations and animations (Stasko, Domingue, Brown & Price, 1997) and is accompanied by text and/or video. DELYS offers the student multiple representations for a given study subject using simultaneously text, figures, simulations, experimental setups, and video. As regards simulations, the student can control the parameters and execute the simulations step-by-step. Using different input parameters the student can explore the possibility of obtaining different outputs. Our software makes heavy use of visualization techniques in order to explain the various functions/procedures of a computing system or the various algorithms (or programs) and help students comprehend concepts or functions that are very difficult to comprehend by observing a computer or by using other traditional teaching means.

4.1 The interface

The interface used is uniform across all the implemented microworlds (Alessi & Trollip, 2001). The main environment uses a full-screen menu to guide students to the 4 microworlds (Figure 1). A frame menu located at the bottom of the screen is common to all microworlds and offers access to a series of common additional functions that enhance the usability of the system:

- Selective navigation in all environments: Students can navigate the system according to their needs (return to the previous environment, exit the system, connect to the WWW or the class intranet, etc.).
- Dictionary of terms: A useful add-on that is accessible at all times.
- Notes: Students can keep electronic notes during the various didactic activities and use them at a future time, for example, while preparing a project.
- Copy/Paste: certain objects from the software, like text and figures, can be copied and pasted to other applications, for example, a word-processor.
- Printing: Students can print text or figures.
- Network connection: Students can connect to a network (the local intranet or the internet) in order to send or receive messages, participate in electronic discussions, and collaborate with other students.
- Help: A versatile context-sensitive help system is provided.

Figure 2 shows a help-screen where the movement of the mouse cursor over each numbered area displays the corresponding help message.



Στη ενότητα "Διερεύνηση του Η/Υ" θα γνωρίσεις τα συστατικά του υπολογιστή και θα γνωρίσεις σε βάθος τη μονάδα του σκληρού δίσκου. Στο σημείο αυτό καλείσαι να επιλέξεις το περιφερειακό που σε ενδιαφέρει.

Figure 2. Help system

The graphical user interface was designed according to the state-of-the-art requirements for GUI design (Alessi & Trollip, 2001; Shneiderman, 1998). Information is presented in a compact and consistent manner: specific actions produce specific outcomes. User-friendliness was not used as an end in itself but was used in order to maximize its didactic effectiveness.

- Depending on the needs of each environment, the most appropriate interface is used in order for the various didactic goals to be achieved. The various visible and active icons function depending on the operations executed in each specific working environment. In this way, students do not have to memorize their functionality and can use them to accelerate the execution of certain system functions.
- Where deemed appropriate, the text is accompanied by verbal narration (Clark & Paivio, 1991). In some cases distinctive sounds are used to emphasize the responses of the system.
- Where also deemed necessary, animation is used to better explain a naturally dynamic concept, i.e., something that evolves in space and time.

- In all environments students do not remain passive, but actively participate in all the proposed activities by interacting through DELYS.
- The software is accompanied by a set of manuals for the teacher and the student that include teaching scenarios for the best possible utilization of the pedagogical capacity of the software.

DELYS has been implemented on the Microsoft Windows platform using Macromedia Director and the compiler construction tools LEX and YACC (Aho, Sethi, & Ullman, 1988).

4.2 Description of the Microworlds

4.2.1 Microworld 1: The Components of a computer

This microworld presents the hardware of a personal computer. It is a graphical interactive navigation of the basic internal and external components (or peripherals) of a computer. Students can explore the technical characteristics of each component, the way it is connected or attached to the system, and its role in the functioning of the system. This is done by having students use drag and drop techniques to internally connect (or externally attach) various components in a virtual computer (see Figure 3).

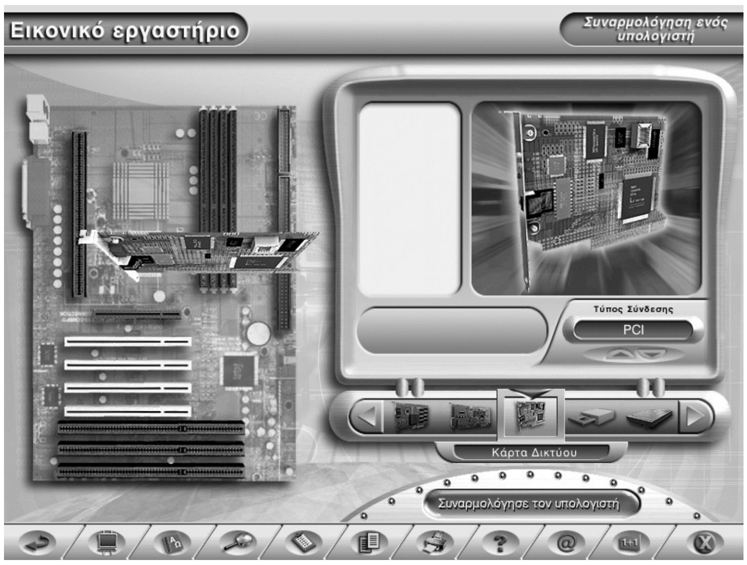


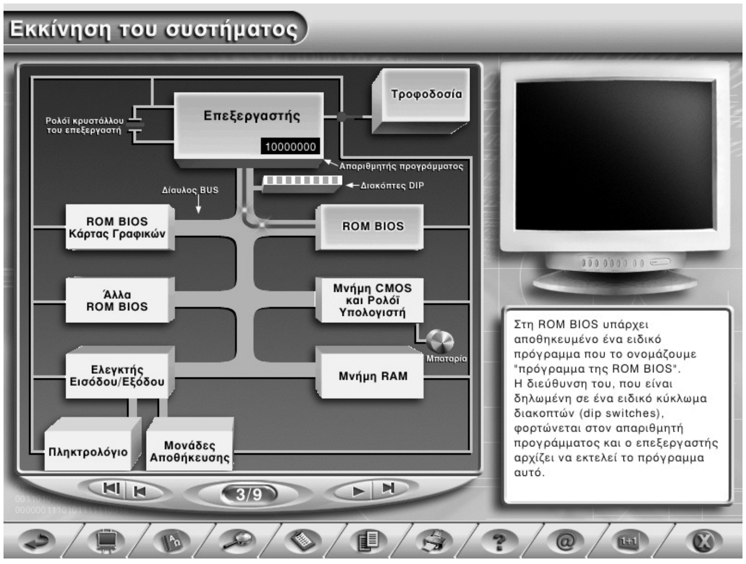
Figure 3. Assembling a computer

Students become acquainted with the operation of a computer and its basic components. Although many of the students may be familiar with computers, i.e., they may own a personal computer, they do not necessarily have a solid understanding of the way a computer and its various components relate and operate as a whole. As is often the case, their approach is empirical, i.e., they have mastered a series of practical workarounds (some *theoremes-in-action* (Vergnaud, 1981) that allow them to cope with certain problems they may encounter when using a computer and has a procedural character: students often can complete specific tasks but have a false idea of the whole function of the system.

It is important to mention that DELYS can easily be customized so that it replaces certain components of a computer with newer, more modern ones, in order to always expose students to the most current computer technology in use.

4.2.2 Microworld 2: Booting a computer

This microworld explains the boot process. This is accomplished by presenting the specific components involved, explaining the role of each component in the boot process, and displaying appropriate messages in a virtual computer screen in order to associate what is seen with what is actually happening inside a computer. Students are presented with a graphical representation of the computer internals (i.e., motherboard with power source, CPU, DIP switches, bus, ROM BIOS, RAM, graphics card, other cards with their own BIOS, keyboard, and secondary storage). We use animation (Park & Hopkins 1993; Stasko et al, 1997) to describe the interaction between the various components involved in the phases of the boot process. In this microworld we have included verbal narration in order for students to be able to follow all the details of the boot process. Students can replay each phase or navigate through the sequence of phases that comprise the boot process (see Figure 4).



Στη ROM BIOS υπάρχει αποθηκευμένο ένα ειδικό πρόγραμμα που το ονομάζουμε "πρόγραμμα της ROM BIOS". Η διεύθυνση του, που είναι δηλωμένη σε ένα ειδικό κύκλωμα διακοπτών (dip switches), φορτώνεται στον απαριθμητή προγράμματος και ο επεξεργαστής αρχίζει να εκτελεί το πρόγραμμα αυτό.

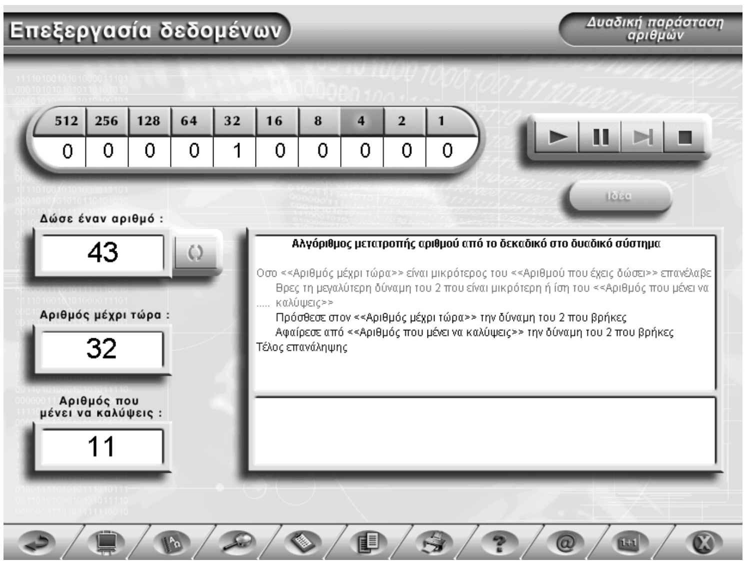
Figure 4. The Boot Process

The goal of this microworld is not to offer detailed technical information about the boot process itself, but to help students comprehend the principles of how a computer operates and in this way gain control of the system behavior (messages that appear on the screen, beeping sounds, etc). In addition, students may be able to identify the basic types of malfunctioning and/or failures that can occur when a computer is turned on.

4.2.3 Microworld 3: Representing data and processing numbers

The transformation of character strings to ASCII code strings and numbers to a series of binary digits and vice versa, and the performance of operations on numbers are typical examples of processes including both a technical and a mathematical dimension. We tried to split the above processes into their component parts and create models that simulate the theoretical aspects involved. In particular, in this microworld a suite of activities helps students to:

- Understand the way characters are encoded in ASCII code and experiment by moving back and forth arbitrary characters and their binary representations.
- Understand an algorithm by which numbers represented on a decimal basis are transformed by the computer into their binary representation. There are three modes of execution: (a) watch mode, where students observe the software perform and explain each step of the algorithm, (b) step-by-step mode, where the algorithm is executed at the students' pace, and (c) direct manipulation mode, where students are free to transform any number into its binary representation by setting on and off its bits and asking for a hint when they find it difficult to proceed. For that purpose we have implemented a backtracking algorithm. Students are not forced to follow the correct algorithm; at any time they can undo the wrong decisions and, by using the provided hints, solve the problem (see Figure 5).



Αλγόριθμος μετατροπής αριθμού από το δεκαδικό στο δυαδικό σύστημα

```
Οσο <<Αριθμός μέχρι τώρα>> είναι μικρότερος του <<Αριθμού που έχεις δώσει>> επανέλαβε
    Βρες τη μεγαλύτερη δύναμη του 2 που είναι μικρότερη ή ίση του <<Αριθμός που μένει να
..... καλύψεις>>
    Πρόσθεσε στον <<Αριθμός μέχρι τώρα>> την δύναμη του 2 που βρήκες
    Αφαίρεσε από <<Αριθμός που μένει να καλύψεις>> την δύναμη του 2 που βρήκες
Τέλος επανάληψης
```

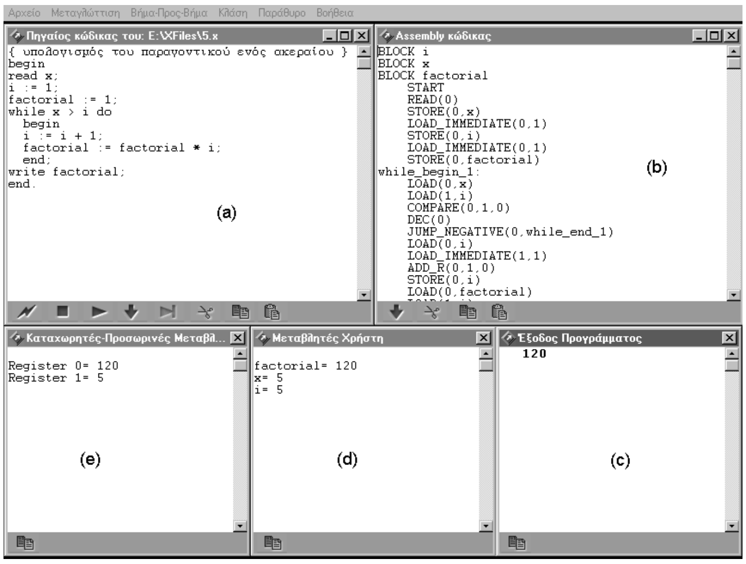
Figure 5. Transforming a decimal number to its binary representation

- Understand the operation of addition of two binary numbers. Again, students can choose any of the above-mentioned modes. In the direct manipulation mode students add the binary digits one by one taking into consideration the carried over digit. The software tests the correctness of each step by allowing students to proceed.
- Understand how the computer hardware implements the addition of binary digits. First, students see how the AND, OR, and NOT logical circuits work, and then, see how these circuits can be used to implement a half-adder and a full-adder. In all cases students can experiment by choosing the input digits.

4.2.4 Microworld 4 - The programming environment: X-Compiler

We have implemented a programming environment, called X-Compiler, where students can write programs in a simple Pascal-like language (see Figure 6). The language supports the *assignment*, *if ... then*, *while ... do*, *read*, *write* and *compound* statements. Identifier variables and numbers are integers and all arithmetic expressions evaluate to integers.

The programming environment consists of five windows: (a) source code, (b) assembly code, (c) program output, (d) program variables watch, and (e) registers and compiler generated temporary variables watch. We provide two modes of operation: (i) novice user mode, where only windows (a) and (c) are active, and (ii) advanced user mode, where all windows are active. Students can activate any window they want.


Αρχείο Μεταγλώττιση Βήμα-Προς-Βήμα Κλάση Παράθυρο Βοήθεια
Πηγαίος κώδικας του: E:\XFiles\5.x
Assembly κώδικας
(a)
(b)
Καταχωρητές-Προσωρινές Μεταβλ...
Μεταβλητές Χρήστη
Έξοδος Προγράμματος
(e)
(d)
(c)

```
{ υπολογισμός του παραγοντικού ενός ακεραίου }
begin
read x;
i := 1;
factorial := 1;
while x > i do
  begin
  i := i + 1;
  factorial := factorial * i;
  end;
write factorial;
end.
```

```
BLOCK i
BLOCK x
BLOCK factorial
    START
    READ(0)
    STORE(0,x)
    LOAD_IMMEDIATE(0,1)
    STORE(0,i)
    LOAD_IMMEDIATE(0,1)
    STORE(0,factorial)
while_begin_1:
    LOAD(0,x)
    LOAD(1,i)
    COMPARE(0,1,0)
    DEC(0)
    JUMP_NEGATIVE(0,while_end_1)
    LOAD(0,i)
    LOAD_IMMEDIATE(1,1)
    ADD_R(0,1,0)
    STORE(0,i)
    LOAD(0,factorial)
```

```
Register 0= 120
Register 1= 5
```

```
factorial= 120
x= 5
i= 5
```

```
   120
```

Figure 6. The Programming Environment (advanced user mode)

The assembly language used is a pseudo-assembly that runs on a virtual machine with two registers and includes the basic LOAD, STORE, COMPARE, JUMP, ADD, etc., instructions needed to implement the source language. Students must write and compile a program. If there are errors the software reports them and suggests possible solutions. Once there is a compiled program, students can either execute it or step through source or assembly code statement by statement, watching the association between source and assembly statements and the intermediate values of program variables, machine registers, and temporary variables (if any).

The didactic objective of X-Compiler is to offer students an elementary programming environment with simple high level and pseudo-assembly languages and clarify the phases of compilation and program execution that usually constitute a "black box" in professional programming environments (du Boulay et al, 1989). Brusilovski, Calabrese, Hvorecky, Kouchnirenko & Miller (1997) point to three factors related to the programming environments most often used in teaching programming and which cause significant difficulties to novice students:

- General-purpose languages come with integrated programming environments that are too complex and sophisticated for the novice programmer. Thus, learning the fundamentals of programming and comprehending the basic syntax of a programming language is a hard task and a time consuming process.
- Programming environments are intended for professional programmers. Syntax editors, compilers, and debuggers cause bewilderment to a student programmer.
- Programming environments do not usually offer the option of visualizing the execution of a program. As a result, the process of execution remains hidden and students tend to believe that executing a program has to do with data input and data output. This “black box” approach (du Boulay et al, 1989) conceals the semantics of the programming language.

X-Compiler offers interesting didactic features. Users get detailed feedback on the errors encountered during compilation, and are always aware of everything that happens to the internals of the mental machine during program execution (by seeing the correspondence between source and assembly code, the intermediate values of the machine registers, the system generated temporary variables, their own variables, and the contents of the output window). Moreover, users can alter the produced assembly code and then execute it.

We provide teachers and students with the appropriate manuals that contain a series of educational activities on the use of X-Compiler. We have designed the included activities based on the findings of the research community and our teaching experience on the difficulties encountered by students that are novice programmers.

For example, the following case of “cognitive transfer” could be a potential source of difficulty for novice programmers trying to solve problems in a traditional programming environment. Some students may believe that the following code computes the area of a rectangle:

```
area := base * height;
read(base);
read(height);
write(area);
```

They will be surprised to realize that area is not computed correctly. In the X-Compiler programming environment they can see why the above program is not correct by observing the intermediate values of their variables.

Now, consider the code fragment below that swaps the values of variables A and B.

```
TEMP:=A
A:=B;
B:=TEMP;
```

The teacher can observe that one can get the same effect without using the extra variable TEMP, as shown in the following code:

```
A:=A+B;
B:=A-B;
A:=A-B;
```

Students can examine the intermediate values of the variables and understand why this solution is correct. The teacher could then show that this solution is slower (because it uses more assembly instructions than the previous solution) and also it does not always work correctly (when we have integer addition underflow or overflow).

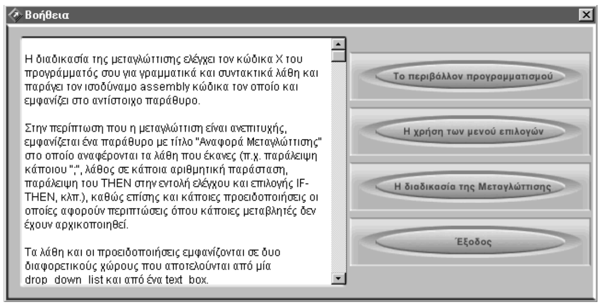
These two examples demonstrate the didactic capabilities of our programming environment. Students can not only examine whether their programs produce the correct output, but also discover quickly and easily the syntactic and semantic errors they make.

Special attention was paid with the help system accompanying the programming environment. Apart the difficulties novice programmers encounter, mentioned above, the international research literature reports additional problems related to the comprehensibility of the error messages produced during the detection of run-time errors (Freund & Roberts, 1996).

For the programming environment X, two types of help systems exist:

- regular help files for the programming environment (Figure 7), and
- an editor sensitive to user double-clicks, i.e., users can double-click keywords, operators or delimiters to obtain detailed information about them (Figure 8).

During compilation, syntactic errors in the source code trigger a pop-up window that contains two drop-down lists, one for the detected errors and one for the warnings issued by the compiler. Users can choose the list element they desire to get an explanation of the type of the error or warning. At the same time the appropriate line of the source code is highlighted (Figure 9).

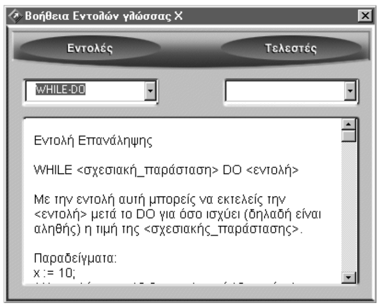

Βοήθεια
Το περιβάλλον προγραμματισμού
Η χρήση των μενού επιλογών
Η διαδικασία της Μεταγλώττισης
Έξοδος

Η διαδικασία της μεταγλώττισης ελέγχει τον κώδικα X του προγράμματός σου για γραμματικά και συντακτικά λάθη και παράγει τον ισοδύναμο assembly κώδικα τον οποίο και εμφανίζει στο αντίστοιχο παράθυρο.

Στην περίπτωση που η μεταγλώττιση είναι ανεπιτυχής, εμφανίζεται ένα παράθυρο με τίτλο "Αναφορά Μεταγλώττισης" στο οποίο αναφέρονται τα λάθη που έκανες (π.χ. παράλειψη κάποιου ";", λάθος σε κάποια αριθμητική παράσταση, παράλειψη του THEN στην εντολή ελέγχου και επιλογής IF-THEN, κλπ.), καθώς επίσης και κάποιες προειδοποιήσεις οι οποίες αφορούν περιπτώσεις όπου κάποιες μεταβλητές δεν έχουν αρχικοποιηθεί.

Τα λάθη και οι προειδοποιήσεις εμφανίζονται σε δυο διαφορετικούς χώρους που αποτελούνται από μία drop down list και από ένα text box.

Figure 7. The help system of X-Compiler



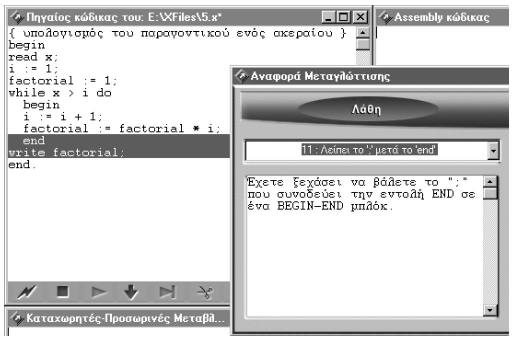
Εντολή Επανάληψης

WHILE <σχεσιακή_παράσταση> DO <εντολή>

Με την εντολή αυτή μπορείς να εκτελείς την <εντολή> μετά το DO για όσο ισχύει (δηλαδή είναι αληθής) η τιμή της <σχεσιακής_παράστασης>.

Παραδείγματα:
x := 10;

Figure 8. On-line help for X statements


Πηγαίος κώδικας του: E:\XFiles\5.x*
Assembly κώδικας
Αναφορά Μεταγλώττισης
Λάθη
Καταχωρητές-Προσωρινές Μεταβλ...

```
{ υπολογισμός του παραγοντικού ενός ακεραίου }
begin
read x;
i := 1;
factorial := 1;
while x > i do
  begin
  i := i + 1;
  factorial := factorial * i;
  end
write factorial;
end.
```

11 : Λείπει το ';' μετά το 'end'

Έχετε ξεχάσει να βάλετε το ";" που συνοδεύει την εντολή END σε ένα BEGIN-END μπλόκ.

Figure 9. Error detection and advice to the programmer

4.2.5 Virtual Classroom and Virtual Laboratory

In addition to the above-mentioned microworlds, we allow students to test the acquired knowledge by taking tests composed of true/false, multiple choice, fill-in the blank, and matching type questions. Students must answer each question and then can check for the correct answer. Using the provided administration tool, teachers can add more questions and prepare and administer exams through the classroom intranet using DELYS. The software automatically grades the answers and keeps statistical data for each student.

We also provide a virtual laboratory where students can build a full computer system from scratch. The computer initially consists of a case and a motherboard. Students can add various components and try to power-on the system. This is achievable only when all required components have been connected.

Finally, an extra virtual laboratory activity has been implemented to help students understand the relationship between the decimal and binary number representation systems. We use a pair of scales and the idea is to drag and drop decimal and binary weights respectively on each tray so that the scales balance (see Figure 10). The set of weights on the left tray corresponds to the decimal system, i.e. the system most commonly used and the one on the right corresponds to the binary system, i.e. the system used on computers.

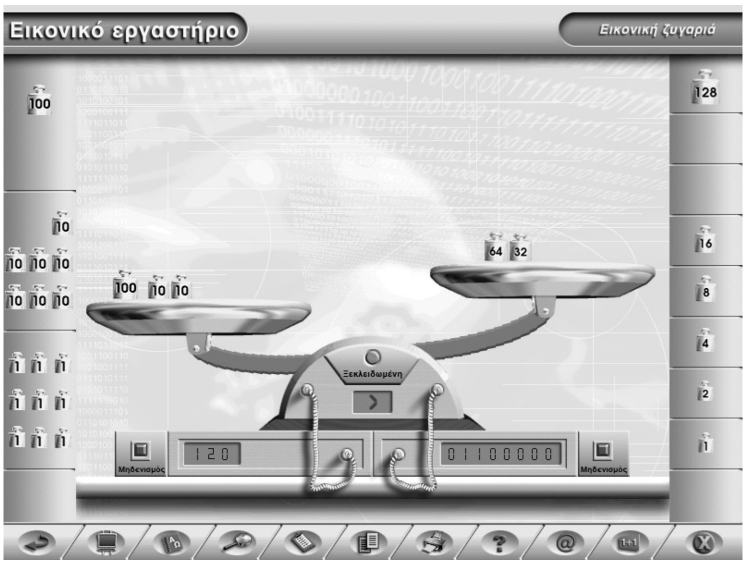


Figure 10. The Virtual Scale

We use multiple representations (weights and digits) for the decimal and binary numbers. Students can alter either representation and the software automatically updates the other one[4].

An attempt to generalize the properties of numbering systems is hindered by the familiarity students have with the decimal numbering system. The purpose of the scales is to familiarize students with the binary system via an alternative representation approach and also to explore some basic properties of numbering systems. This didactic goal can be achieved by using a variety of activities, which involve the placing and removal of weights and are accompanied by questions like the ones below:

- Is it always possible to balance a given decimal weight with weights from the right set of weights? In other words, for every integer weight expressed in decimal weights does there exist an equivalent binary weight representation? Is this representation unique? What about the other way around? In other words, are the two representation systems equivalent, i.e. is it possible to uniquely represent an integer in both systems?
- Would you be able to determine the required weights for a numbering system with a base of 3 or 5? Would we be able to devise any system we wish by choosing as a base an integer greater than 2?

The main aim of the above mentioned activities is to make students realize the fact that there exist certain invariable properties in numbering systems and to guide them in discovering and mastering them.

The teacher can propose, as an alternative, the use of notation as a language for describing and understanding numbering systems. As an example, the teacher requests

[4] The interested reader can visit http://macedonia.uom.gr/~delys and experiment with a web-based version of this component.

one student to leave the classroom for a while and asks the rest of the class to unanimously place a certain weight on the right tray, e.g., $(10111)_2 = (23)_{10}$ units. Next, the teacher proposes that one of the students writes on the blackboard the shortest possible message, so that, when the student who is removed from the classroom returns, immediately understands which weight has been placed on the scales. Naturally, the shortest message is to write the binary representation of 23. Should no student consider this, the teacher can of course propose this message.

Then, the first student is asked back into the classroom and the teacher asks him/her to guess which weight has been placed on the scales by examining the message on the blackboard. The student will of course give the correct answer. The teacher will, however, insists that something is “still not quite right” and requests a second student (one who believes that has understood the basic principles) to leave the classroom. Then, the teacher asks the rest of the class to place 111 units on the left tray. The student that re-enters the classroom and most probably presumes that the new message indicates that there are 7 units on the right tray. This confusion stems from the fact that notation 111 can represent two different numbers, and thus, students realize the importance of the proper notation: $(111)_2$ or $(111)_{10}$.

5. Evaluation of DELYS

DELYS was tested and evaluated in a number of high schools during the second semester of the academic year 1999-2000. The participating schoolteachers used the software to teach the subjects “Exploring the components of a computer”, “Assembling a computer”, “The Virtual Scale” and “Programming environment”. On average, students spent 6 hours using the software.

Students were from the fourth year of a technical high school that follow the computer programmer course and also students of a general high school who attend a class on computer science literacy for 3 hours a week.

In the beginning, a short demonstration introduced the software environment to the participating students. Then, the students were given various activities to help them explore the microworlds included in DELYS. Finally, the students worked on the decimal and binary number representation systems using the Virtual Scales and the programming environment, and were also given activities similar to the ones described in the previous section to practice on.

The students were presented with a series of open questions and problems (Brousseau, 1984) that urged them to use the software microworlds in order to formulate guesses, verify or reformulate them, improve their solutions, and prove the correctness of their assumptions.

The students were eager to use the software and willing to answer a questionnaire at the end of the class. They explored by themselves all the included microworlds - even students who were usually not interested in attending class.

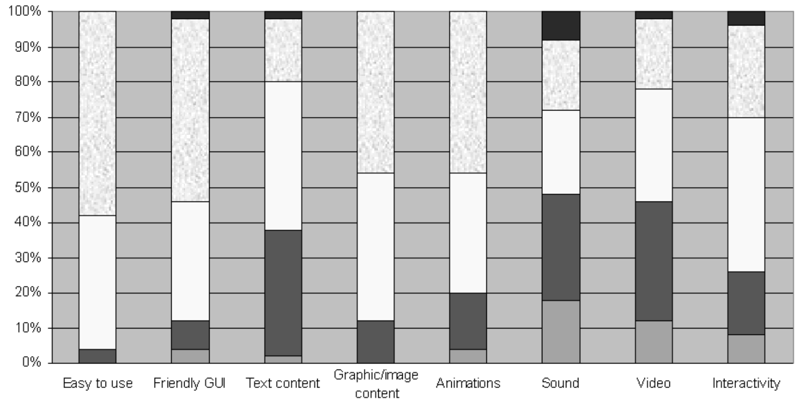
The answers to the questionnaires reveal that more than half of the students used multimedia software for the first time. Almost all the students used educational software for the first time and perhaps this is the reason why their answers to all the questions show that they are willing to use educational software and that they actually wish that educational software takes its place in the educational curriculum.

It must be mentioned that the students used beta versions of the software and their input greatly improved the final version of the software. In summary, we claim that the evaluation of DELYS helped all the parties involved (students, teachers and the developing team).

Table 1 and Figure 11 show the evaluation questionnaire and the corresponding answers of the students. (N=50).

	QUESTION	% yes	% no	% no answer
1	Is this the first time you have worked with multimedia software?	56	44	0
2	Is this the first time that you were taught a course using educational software?	84	16	0
3	Do you believe your learning experience was improved by using this software?	84	16	0
4	Do you think that the software helped you understand difficult concepts?	86	14	0
5	Did the complementary use of the software in the teaching process make the course more interesting to attend than before?	88	12	0
6	Would you like to have your teachers use educational software in addition to the blackboard approach?	94	6	0
7	Do you think that DELYS is a quality software?	86	10	4
8	Would you like to use DELYS at home while studying?	74	26	0
9	Would you like to see similar software accompanying your textbooks?	90	10	0

Table 1. Evaluation questionnaire and student answers



	Easy to use	Friendly GUI	Text content	Graphic/image content	Animations	Sound	Video	Interactivity
■ no answer	0	2	2	0	0	8	2	4
□ 4	58	52	18	46	46	20	20	26
□ 3	38	34	42	42	34	24	32	44
■ 2	4	8	36	12	16	30	34	18
■ 1	0	4	2	0	4	18	12	8

Figure 11. Evaluation of the multimedia content

DELYS was also tested on a group of High school teachers who were attending a six month training program[5]. These teachers in turn used DELYS at various schools. Their observations verify our findings: students were pleased to use DELYS and found it easier in certain situations to understand basic concepts related to the operation of computer systems.

Finally, we should report that the Hellenic Pedagogical Institute[6], a state organization acting under the supervision of the Greek Ministry of education, evaluated our software before using it in 300 schools for the academic year 2001-02. The Pedagogical Institute judged that our software offers important teaching aid for computer science related courses in the secondary education curriculum.

6. Synthesis and Conclusions

As already stated, the implementation of DELYS was based on a set of theoretical principles which guided its design. A basic working hypothesis was the fact that students already possess certain ideas and beliefs. Subsequently, both the educational environment created and the activities proposed are not based on the rejection of the students' erroneous behaviours and/or beliefs, but rather on the intrinsic inconsistencies that these behaviours and/or misconceptions will unavoidably lead to. For example, in the activity with the decimal/binary scales, if the student chooses the wrong weights then the software does not intervene. Instead, it lets the student reach an inevitable dead-end. Similarly, in the activity with the decimal and binary number representations, the built-in algorithm deals with the erroneous input and provides students with appropriate indication each time. There are many other proposed activities, such as the "area computation problem" in programming, which aim at

[5] Computer Technology Institute. *Project E42- Post-graduate instructor training*: http://odysseia.cti.gr/English/ODYSSEIANEW/e42/e42.htm

[6] Hellenic Pedagogical Institute: http://www.pi-schools.gr/

provoking inconsistencies among the possible student beliefs and the programming reality. Despite this fact, the realization of inconsistencies is not considered sufficient to reverse students' erroneous conceptions and make them adopt correct ones. The teacher should use these inconsistencies appropriately in such a way that students accept the new knowledge.

We consider the collaboration among students very important. That is why the design of our software favours collaboration in small teams (e.g., students sharing a computer), in larger groups (e.g., an entire traditional or virtual class). The realization of the proposed activities reinforces the collaboration among students by allowing them to share their individual reasoning with the rest of the team or class. Thus, on the one hand the networking capabilities of the software enable students to communicate with each other, and on the other, the proposed activities are based on the participation and collaboration of the entire class.

Our software focuses on the concepts involved in a computing system and not only on the hardware. The aim is to help students master those concepts and use them as tools in the problem solving process. On the other hand, teachers can control and fine-tune the parameters of the didactic scenarios by choosing an appropriate activity for each situation. In addition, our software emphasises on the inter-scientific nature of the involved concepts.

The proposed activities are geared towards introducing students to the very nature of the scientific process. The students are involved in problem solving procedures that require: (a) the quest for information, (b) the processing of data, (c) the construction of mental tools, and (d) the formulation, testing and verification of theories and hypotheses in order to solve the proposed problems. For example, in the numbering systems related activities, the students go from the exercise with the scales, to the conversion of algorithms from/to the decimal to/from the binary systems. They also generalize with the octal and hexadecimal systems; they explore the properties of the numbering systems; they perform arithmetic operations on binary representations of numbers; and they generalize the properties of arithmetic operations to any other numbering systems. All these activities are realized through a series of open questions and problems where students are asked to explore a "mathematical phenomenon", formulate conjectures, verify them, support and possibly amend their conceptions, test and rectify their solutions, and possibly prove the correctness of their proposals.

Contrary to the customary approach, we do not consider the use of the software itself as a prerequisite condition for the successful teaching of computer related courses. We believe that educational software becomes meaningful and useful only when it functions as a tool in a teaching situation. Contrary to the traditional *ex-cathaedra* teaching model the whole class participates actively in finding the solution to a given problem.

In this context, the educational software works as a source of organized information that facilitates the solution of a given problem and not merely provides "encyclopaedic" information. Educational software can also be used as a virtual laboratory where students can run experiments that they organize and guide themselves in their attempts to solve a problem. That is why the activities we propose have an experimental nature and correspond to open problems that need investigation.

Interaction among students in the proposed activities is a *sine qua non,* if a course is to be successful: students should be independent but not isolated. In this way, the acquisition of knowledge will have a social character and the diffusion of new knowledge will be an important component of each didactic situation. The proposed educational software also functions as a teaching aid, which supports the teaching of certain educational material that cannot be elaborated using conventional methods.

Since DELYS is a pilot software, it obviously does not fully cover all computer science subjects taught in secondary education. However, many interesting extensions from both, a didactic as well as research perspective, will be added to DELYS such as the incorporation of mechanisms that automatically record students' actions.

References

Aho, A.V., Sethi, R. & Ullman, J.D. (1988). *Compilers: principles, techniques, tools*. Addison-Wesley.

Alessi S., & Trollip S. (2001). *Multimedia for Learning Methods and Development*. (3rd ed.). Allyn and Bacon.

Balacheff N. (1988). *Une étude des processus de preuve en mathématiques, chez des élèves de Collège*, Thèse d'Etat, Grenoble, France

Balacheff N. (1991). Treatment of refutations: aspects of the complexity of a constructivist approach of mathematics learning. In: Von Glasersfeld E. (ed.) *Radical constructivism in Mathematics Education* (pp.89-110). Dordrecht : Kluwer Academic Publisher.

Brousseau G. (1997). *Theory of Didactical Situations in Mathematics*, Didactique des mathématiques, 1970-1990, Kluwer.

Brusilovski, P., Calabrese, E., Hvorecky, J., Kouchnirenko A. & Miller P. (1997). Mini-languages: a way to learn programming principles. *Education and Information Technologies*, 2(1), 65-83.

Clark J., & Paivio A. (1991). Dual coding theory and education. *Educational Psychology Review*, 3, 149-210.

du Boulay, B., O'Shea, T. & Monk, J., (1989). The Black Box inside the Glass Box: Presenting Computing Concepts to Novices. In E. Soloway & J. Sprohrer, *Studying the Novice Programmer*. Lawrence Erlbaum Associates, Hillsdale.

Freund, S. N. & Roberts, E. S. (1996). THETIS: An ANSI C programming environment designed for introductory use. *Proceedings of the SIGSCE '96*, 300-304.

Mayer R. (1997). Multimedia learning: Are we asking the right questions? *Educational Psychologist*, 32(1), 1-19.

Park O., & Hopkins R. (1993). Instructional conditions for using dynamic visual displays: a review. *Instructional Science*, 22, 1-24.

Piaget J. (1974). *Recherches sur la contradiction, Vol.2: les relations entre affirmations et négations*, Paris, PUF.

Shneiderman, B. (1998). *Designing the User Interface*. Addisson Wesley.

Stasko J., Domingue J., Brown M., & Price B. (Eds) (1997). *Software Visualization: Programming as a Multimedia Experience*. MIT Press.

Vergnaud G. (1981). Quelques orientations théoriques et méthodologiques des recherches francaises en didactique des mathématiques. *Actes du V Colloque du groupe Psychology of Mathematics Education*, Grenoble, France.