Teaching Experiences in Programming Using the Visual DaVinci Language

Raúl Champredonde
Laboratorio de Investigación y Desarrollo en Informática
Facultad de Informática - Universidad Nacional de La Plata

Analía Palacios
Instituto de Investigaciones Educativas
Facultad de Humanidades y Ciencias de la Educación - Universidad Nacional de La Plata

Virginia Ainchil
Laboratorio de Investigación y Desarrollo en Informática
Facultad de Informática - Universidad Nacional de La Plata

Abstract

The experience obtained teaching the subject Computer Programming using the Visual DaVinci language in a methodology oriented to the development of thinking skills in problem solving is presented.

This methodology was applied to a group of freshmen taking the Introductory Course for the Higher Course of Studies on Computer Science at the Faculty of Computer Science of the National University of La Plata, Argentina.

Introductory Course

The Introductory Course to the Higher Course of Studies on Computer Science at the Faculty of Computer Science of the National University of La Plata (La Plata, Province of Buenos Aires, Argentina), receives about 1,200 first-year students every year from different locations, not only from the province of Buenos Aires but from the whole country as well.

The course has a duration of sixty days, and it includes a final test which students do not need to pass if their attendance to classes is at least 80%. It has two basic objectives:

- To level students common knowledge, so that when the courses for the first year of the Course of Studies begin, all students are in similar learning conditions. This objective is related to the diversity of academic levels at the different high schools where the students come from, as well as to the different orientations of these high schools.
- To teach the basic programming and problem solving mechanisms, specially those suitable to be solved by means of computer programming.

During the Introductory Course, students re-enforce or acquire basic and fundamental concepts of logic, mathematics, computers and peripherals inner architecture, computers use, but, above all, there is an introduction to basic structured programming.

For this introduction to structured programming, a language specifically designed with this purpose and which is the product of a development and research line of the LIDI, is used [De Giusti, 1989]. Within this line of work, the language has evolved, and it gradually became the Visual DaVinci language and its extensions [Champredonde, 1997] [Champredonde, 1998].

Visual DaVinci

Visual DaVinci is a language whose aim is the programming of an abstract robot that moves around the streets of a square, and also abstract, city, which is 100 horizontal streets by 100 vertical avenues (figure 1). The robot walks from one corner to another by executing the instructions specified in the program.

During its run, the robot can pick up and leave flowers, and pick up and leave papers, and for this purpose it carries two bags, one for flowers and one for papers.

In addition to this, there might be obstacles at the corners that block the way. The programmer should take these obstacles into account and find alternative paths when necessary.

The language defines the primitive instructions that the robot executes and the necessary builders for modular and structured programming.

Programs development can be carried out visually or textually. In the first case, it is done by means of the creation of a visual diagram of the Nassi-Schneiderman type [Nassi, 1973], which adheres to the control flow paradigm [Glinert, 1990], as the one shown in figure 2. In the second case, it is done by means of a conventional textual code, such as the one in figure 3.

In both cases, all that is done in one of the representations is simultaneously and automatically reproduced in the other. That is, when an element is added to the visual diagram, it is automatically incorporated to the textual code, and vice versa.

Syntactic verification and the execution of the algorithms developed is carried out over the textual code in order to make language efficiency independent from its visual nature.

The edition of a program generates a dynamic hierarchy of objects that are able to compile themselves. The result of the compilation is another dynamic hierarchy of objects that are able to execute themselves. Within this hierarchy is the robot object, which is in charge of running the primitive instructions.

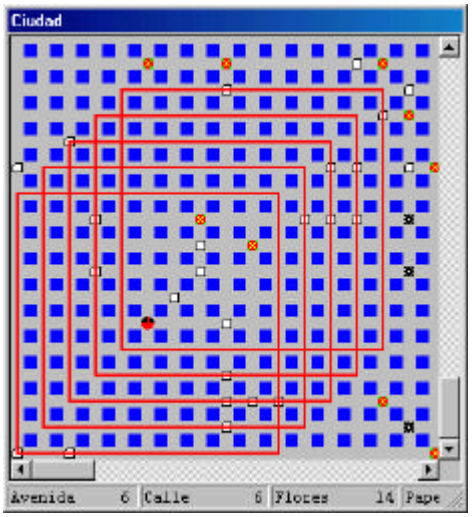

Figure 1: City

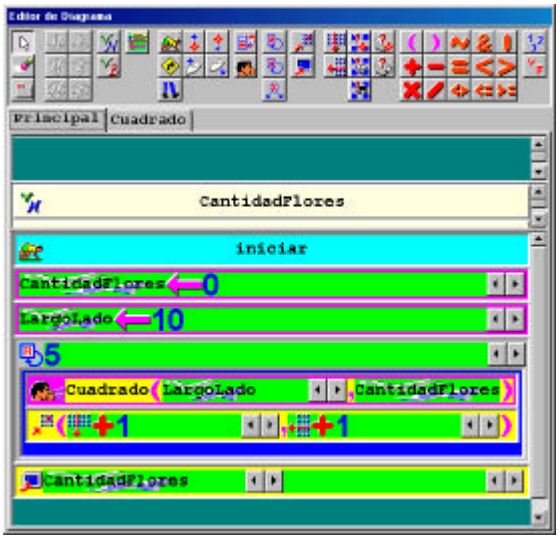

Figure 2: Visual diagram

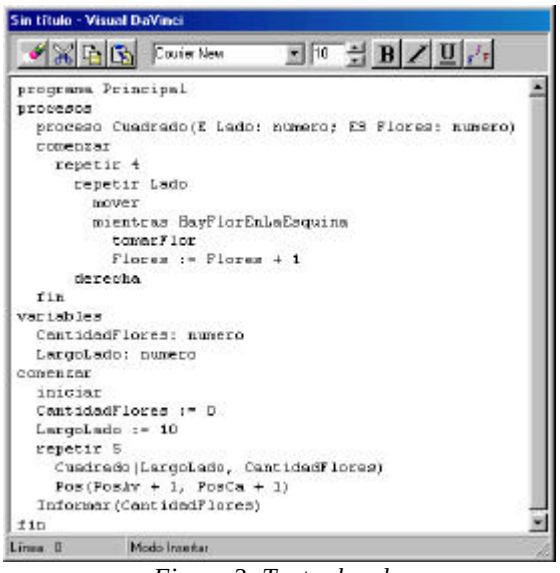

Figure 3: Textual code

Teaching Methodology

For the Introductory Course 2000, carried out during the months of February and March, a teaching methodology for programming has been designed, whose main purpose is the optimization of cognitive and meta-cognitive skills of our students.

Programming is a case of problem solving which requires the user to reach some kind of mental representation of the problem, convert it in a computable model, analyze solution alternatives, and decide on the most efficient execution sequence to perform a specific function of the program [De Giusti, 1998].

Programming suitable solutions stimulates mainly the development of skills for the **planning, supervision and evaluation** of the problems posed and of self performance. These skills are considered to be special intelligence ingredients, and every teaching effort is focused on them [Mayer, 1986] [Nickerson, 1987].

The two factors that set apart novice from expert programmers are: the structure of previous knowledge and the basic reasoning strategies applied during problem solving processes through programming. Therefore, the proposed teaching methodology is based on the **conceptual map** of experts, which reflects the mental structure of their knowledge components and the relations among them.

A series of increasing complexity practical activities was designed; each of these introduces new concepts taking the students previous knowledge as a starting point, and setting relations between previous and knew knowledge, so as to create in the students mental structures similar to those of experts.

Each practical work specifies its elements: the problem, the objectives, the ideas or concepts to use, and the task. The task contains, in turn, a list of inductive questions in order to guide the development of the main mental processes needed to define the nature of the problem, represent the information, select the steps and strategies for problem solving, and supervise the actions. An evaluation instrument is presented as a complement for students to make a report about the task in order to carry out any necessary adjustments.

Experience

The methodology here presented was applied to a group of 92 students from an approximate total of 1,200. From these 92 students, 47 were the test group, and 45 the blank group.

A diagnosis of students perceptions as regards their own ability to solve problems was carried out, and this diagnosis information was compared with students opinions, their assessments about the utility of practical tasks, and the performance level achieved by both groups by the end of the Introductory Course.

An evaluation mechanism was designed to evaluate students as regards problem comprehension, information representation, solving steps and strategies selection, and supervision. This mechanism was applied on conventional tests previously evaluated in the conventional way by professors of the course.

Conclusions

For the time being, it is not possible to draw definitive, precise conclusions as regards the teaching methodology proposed for programming. The main reason for this is that the results will only become evident during the course of the academic year, with the performance of the students that were part of the experience in the subject Computers Programming of the first year of the Course of Studies.

However, certain issues can be observed.

Considering the differences in motivation and stimulation between the test and blank groups, as well as the amount of non-solved tests handed in by both groups, a substantial improvement is suggested, at least as regards interest in understanding and learning, and therefore in the learning itself.

On the other hand, certain historical problems of practical classes became evident, since there was no suitable progression based on students previous knowledge, as well as in the evaluation of thinking skills, knowledge components, and acquired reasoning.

References

[Champredonde, 1997] "Design and Implementation of Visual DaVinci". Champredonde R., De Giusti A. III Congreso Argentino de Ciencias de la Computación. La Plata, Argentina. 1997

[Champredonde, 1998] "Visual DaVinci Extension for Real Robot Control". Champredonde R., Romero F., De Giusti A. IV Congreso Argentino de Ciencias de la Computación. Neuquén, Argentina. 1998

[De Giusti, 1989] "Abstract Machines in a First Course of Computer Science". De Giusti A., Lanzarini L., Madoz C. 11th International Symposium "Computer at the University". Zagreb, Yugoeslavia. 1989

[De Giusti, 1998] "Algoritmos, Datos y Programas". De Giusti A., Madoz C. y otros. Ed. Ciencias Exactas. 1998

[Glinert, 1990] "Visual Programming Environment: Paradigms and Systems". Glinert E. IEEE Computer Society Press. 1990

[Mayer, 1986] "Pensamiento, Resolución de Problemas y Cognición". Ed. Paidós. 1986

[Nassi, 1973] "Flowchart Techniques for Structured Programming". Nassi I., Schneiderman B. ACM SIGPLAN Notices, Volumen 8, Número 8. 1973

[Nickerson, 1987] "Enseñar a Pensar". Nickerson R., Perkins D., Smith E. Ed. Paidós. 1987