The use of Games to teach Programming Algorithms

Laura Felice
lfelice@exa.unicen.edu.ar

Martín Fernandez
ofernand@exa.unicen.edu.ar
Departamento de Computación y Sistemas. Facultad de Ciencias Exactas. UNCPBA
Tandil. 7000. Buenos Aires. Argentina

Abstract

Using real world problems as examples to solve with different design techniques of algorithm emphasizes how the learned concepts of design (Backtracking, Divide and Conquer, Greedy and Dynamic Programming) help to create good algorithms.
When we analyze the contributions of a typical course of analysis and design of algorithms, we can observe the students trying to improve their aptitudes in the programming areas. However, the jobs we propose for our courses, could give students with a more enriching experience by providing them a way to work with problems (more precisely, games) building a "real" framework for users and thereby creating the necessary tools.
This paper introduces the author's experiences teaching this course with a system development project as a means to evaluate the applications of the introduced concepts and techniques on assignments and projects.

Keywords: algorithm design techniques, programming teaching, games, complexity, C++ language.

1. INTRODUCTION

The Computer Science Department of Universidad Nacional del Centro de la Provincia de Buenos Aires, specifically the computer science curriculum, offers an extensive exposure to the discipline through two introductory courses of analysis and design of algorithms, "Analysis and Design of Algortithms I" and "Analysis and Design of Algorithms II", which could be termed ADA1 and ADA2. We teach these courses at the second year of the career, and their home page can be found at <http://www.exa.unicen.edu.ar/catedras/aydalgor> and <http://www.exa.unicen.edu.ar/aydalgo2>.

ADA1 starts with an imperative approach of programming, referencing the traditional bibliography like (Aho, et.al., 1985; Aho, et. al., 1995, Baase 1993; Cormen 1990; Horowitz 1997; and Sedgewick 1999), while the second part of the course adopts an object-oriented approach.

In two previous courses of programming ("Introduction to Programming I" and "Introduction to Programming II"), the students have covered the basics of creating programs using imperative programming with Pascal language.

Mainly, ADA1 course provides students with the fundamental concepts of analysis and design of algorithms like complexity, giving emphasis on design techniques of algorithms (Backtracking, Divide and Conquer, Dynamic Programming and Greedy strategies). A knowledge of design will help students to create good algorithms. The number of basic design strategies teached is limited, so the students can develop a good understanding of each techniques.

Also, we introduce the concepts of abstract data types as a basic concept for object-orientation. The object classes that are involved in a problem are identified and specified in an algebraic style. Thus, students learn how to be precise and mathematically rigorous in the type declarations. The specification describes object classes in an abstract way, free from most implementation details. The object class specifications are constructed from previous existing ones by applying mechanisms provided by the algebraic language: generalization, specialization, parameterization, and instantiation. The language used is NEREUS (Favre, 2006).

ADA2 course deals with Graph Theory, Searching problems, NP-C problems, and the basic strategies and algorithms to their resolution, always working with the complexity concepts and linked with abstract data types.

By the other hand, working with a large-scale project has the advantages of doing typical textbook exercises. Based on our own observation, students that have made these assignments have had a better understanding of the techniques and complexity concepts than those students who have not developed this kind of projects. An additional benefit of making projects is that students feel a real sense of accomplishment in completing them. They use of the benefits of using modular design, incremental implementation and complexity analysis, so their work could grow not only in the functionality of the system but also in reusability.

The games development can be a good topic for teaching these techniques. It gives students a real project where they can apply the techniques they are studying. Further, they have the added benefit and the challenge of developing a "real system".

The main goals of this kind of practice include:

- To explore and define best practices in design techniques of algorithms and apply those from traditional games,
- To integrate a theoretical and empirical reasoning using a methodological approach of development not only for the resolution of the game but also for the whole design of the project.

Thus, students get an opportunity to apply these strategies on assignments and projects.

This paper is organized as follow: in section 2 we explain why we propose games development to teach the core concepts of ADA1 and ADA2. In section 3, we present a case study that covers the learning concepts of ADA1 and ADA2. This work has obtained an award at EST 2005 simposium (Fernandez, 2005) as an innovative way to resolve the navigation of video-games. Section 4 deals with the languages and tools used in the courses and to projects development. Finally, conclusions are made in Section 5.

2. WHY GAMES?

Concluding ADA1 course, the first issue with which students are confronted when writing programs is the *problem*. Typically, students are confronted with "real-life'' problems, and they provide a program for the problem.

However, real-life problems are nebulous and the first thing they have to do is to try to understand the problem separating necessary from unnecessary details. So, they obtain their own abstract view, or model, of the problem. This process of modeling is called abstraction of the design.

Designing proper, efficient, and implementable algorithms for real-world problems is a tricky task, because the successful algorithm designer needs access to the main aspects of knowledge: the techniques. Good algorithm designers understand several fundamental algorithm design techniques including backtracking, divide and conquer, greedy, dynamic programming, heuristics and data structures.

We have found that projects based on game development are a good way to introduce the core concepts of ADA1 and ADA2.

Many physical systems can be represented by rules and relationships. Games are also systems based on simple rules and relationships. Many of them can be applied to different situations, so the opportunity of reuse is present many times. For example, in games where players role play strategic situations, like GO game, 1914 wargame or TEG game, a similar intuitive understanding can be developed using heuristics algorithms. The TEG game is an Argentine risk-based board game created during the 1970s. The name is an acronym of Táctica y Estrategia de la Guerra, Spanish for War Tactics and Strategy. Similarly, in games like Scrable game, Rubik Cube resolution, Calculum, etc., a similar reasoning to construct a backtracking scheme can be developed. It's true that these algorithms can potentially take exponential time to run. This can be avoided using heuristics, but in the worst case it might not find a solution. It is important to remark that all of these games have been developed by our students building real interfaces and obtaining a very good behavior of their executions.

However, games are only an instructional resource for us, and teachers can adapt the requirements to meet the course objectives and to evaluate the students. Another advantage of this kind of project is that games have the capability to change the scale, and developers (i.e. students) can use them as opportunities for players to share and critique strategies, then they appreciate this way of learning and they are motivated to get quality results.

Teachers provide the techniques so that the game design could be good, and offer a broad range of conditions to be truly interesting. In general, when we investigate game development through several web sites, we can observe that no matter how good an algorithm is, it has a limited regime of applicability. Sometimes, our students must frequently design a number of algorithms and switch from one to another as conditions change because the odds are that a specific algorithm will work best under a narrow range of conditions. Our intentions as regards game development is that the resulting prototypes have design information, original and creative work accompanying a methodological rationale. Several students have had the experience with dialogue among the game developer communities, and the interest has grown among themselves. In the last year, two works developed by students of ADA1 and ADA2 courses were presented and published at a traditional student workshop (Fernandez, 2005) and (Ridao, 2005). The first gets an award one of the best in the algorithms area.

3. CASE STUDY. AN EXPERIENCE

One of the topics of ADA2 course was searching problems. During the course, the student had been assigned a project were he implemented the artificial intelligence (AI) of the computer in the game of Othello using the alpha-beta variant of the min-max search algorithm. To be accomplished the student had done little investigation and evaluation of the two algorithms and the possible heuristics. The positive experience while doing this first project motivated the student to do a final project of similar characteristics instead of taking the traditional written final exam of the course. When the student approached the teacher, he proposed to continue to study advanced heuristic search algorithms in the context of autonomous navigation. The student wanted to know which search algorithms were used in applications like robot navigation or the AI in real time strategy games. It was clear to the student that the basic algorithms he had learned weren't suitable for the real time requirements of these kind of applications, and he was eager to learn how the search concept was adapted for these problems.

The new project started with the development of a video game like application that would allowed to test the search algorithms. The game would consist on a maze with obstacles that could be pushed and destroyed in real time and where players could be controlled by the user and the computer.

At the same time, he started the search of information on Internet sites about robot navigation and pathfinding in video games, as well as in a few books with information on advanced search that were available. In the short run the student was able to confirm that there was a whole another class of search algorithms called "on-line search algorithms" or "real-time search algorithms", in contrast to the "off-line search algorithms" that he had studied during the courses. Through bibliographical references he came to know about the work of Richard E. Korf, a researcher who has many publications on the search topic and developed one the most popular real time search algorithms: the Learning Real Time A* (LRTA*) (Korf, 1990) and (Korf, 2000). As most the publications weren't available on-line, the student asked professor Korf for some papers, who in turn, very kindly, sent copies of them. With those papers the student had all the information on real time search to finish his research, together with a detailed description of two representative real time search algorithms: Real Time A* (RTA*) and LRTA*.

Once the application was finished, the student added the AI for the computer. Three modules were implemented with different navigation algorithms. The algorithms were the traditional A* (which implements off-line search), and the RTA* and LRTA*. The interactive environment allowed to see how the real time nature of the search algorithms adapted to the modifications in the maze, and how the learning variant of the RTA* actually improved its performance as the iterations passed. This experience allowed the student to compare the off-line and on-line search paradigms in a very visual and interesting way.

The application was developed in C++ standard. The main idea is that the application runs in many platforms, so the used libraries are multi- platform. The proved platforms were Linux, that was the main development platform and all of Windows versions (9x and NT).

4. LANGUAGES AND TOOLS

In ADA1 and ADA2 courses, we intent to use the right mix of languages and language features so, the solution to a problem is much easier to describe and implement, with better results. C++ remains an essential tool for project developments not because anybody thinks it's the best possible language, but because it's a single, portable language that works better than any alternative in each of several areas. Students select the appropriate tools according to the projects. In the most cases, they use visual frameworks.

For many uses, C++ is not the ideal language. Some more experienced students prefer Tcl/Tk for writing a user interface for example. In our courses, C++ is used because it works well when the ideal language is (for whatever reason) not available, and because it interfaces easily with the libraries and the other languages the students use. Students rarely develop a big program written all in one language, or without using libraries, so easy integration with other languages and libraries was a key design goal. C++ was designed with libraries always in mind, and its most useful features are those that help to write portable, efficient, easy-to-use libraries.

We use a methodology linked with formal specifications of abstract data type and Nereus is the used language for the early phase of a project development. Nereus is an algebraic language where the basic unit of specification is the class. Classes may declare types, operations, and axioms that are formulae of first-order logic. They are structured by three different kinds of relations: importing, inheritance, and subtyping. Figure 1 shows the syntaxis of a Nereus class. The keypoint of this way of specifying abstract data types into a rigorous approach was published in (Favre et. al., 2000).

```
CLASS className [<parameterList>]
IMPORTS <importList>
BASIC
CONSTRUCTORS<constructorList>
EFFECTIVE
TYPES <typeList>
FUNCTIONS
<FUNCTIONSList>
AXIOMS <varList>
<axiomList>
END-CLASS
```

Figure 1. NEREUS basic specification syntax's

5. CONCLUSIONS

It is widely known that the study of algorithms is not an end in itself. We intend to teach the topics in a framework that emphasizes factors of good design and quality such as correctness, extensibility, reusability, efficiency, maintainability, etc. Also, the ability to use a systematic, precise, and mathematically rigorous approach to design efficient algorithms is the keypoint in both of courses.

The experiment of algorithms teaching with games has been very satisfactory and has helped us reach some interesting conclusions in relation to the effectiveness of this approach. This kind of works enriches the learning not only of the student but also of the teachers who have to investigate and have to plan an appropriate framework to develop the projects and to make a creative job. We consider that it could be reproduced in any Computer Science study program and we think it is an innovative approach that is interesting to be applied. The necessary prerequisites can be provided by introductory courses on programming, computer science, and mathematics.

6. REFERENCES

Aho, A., Hopcroft, J; Ullman, J. (1983) Data Structure and Algorithms. J.Addisson-Wesley

Aho, A., Ullman, J. (1995) Foundations of Computer Science. C Edition. Computer Science Press.

Baase, Sara (1993) Computer Algorithms. Introduction to Design and Analysis. Second Edition.

Cormen, T; Lierserson, C; Rivest, R. (1990) Introduction to Algorithms. Ed. The MIT Press.

Favre, L; Felice, L; Martinez, L; Pereira, C. (2000) "On Teaching a Data Structures and Algorithms Course through a Rigorous Approach" In Proceedings of 'ISECON 2000: Information Systems Education' - Philadelphia 9-12 Noviembre 2000.

Favre, L; (2006) "A Rigorours Framework for Model-Driven Development" In Advanced Topics in Database Research Series. Vol 5. Chapter 1. IGP (Idea Group Publishing). Keng Siau Ed. USA. Pp: 1-27.

Fernandez, Martin; (2005) "Algoritmos de búsqueda heurística en tiempo real. Aplicación a la navegación en los juegos de video." EST 2005 (Concurso de Trabajos Estudiantiles): 34 JAIIO. Jornadas Argentinas de Informática e Investigación Operativa. Rosario. Argentina.

Horowitz, E;Sahni, S; Rajasekaran, S.(1997) Computer Algorithms/C++.

Korf, Richard (2000) " Real - Time Heuristic Search" In "Artificial Intelligence", 42.

Ridao, I; Vidal, S; (2005) " Algoritmos de Resolucion para el cubo de Rubik". EST 2005 (Concurso de Trabajos Estudiantiles): 34 JAIIO. Jornadas Argentinas de Informática e Investigación Operativa. Rosario. Argentina.

Sedgewick, R. (1999) Algorithms in C++. Addison-Wesley.