THE ARCADE LABORATORY: AN ENVIRONMENT TO HELP TEACH ALGORITHMS

CAGNAT JM, GUERAUD V, PEYRIN JP.
Laboratoire de Génie Informatique
IMAG-Campus BP 53X
38041 Grenoble Cedex
France

We propose different ways to use the computer to help teach algorithms. Our objective however is not to develop a complete and autonomous computer assisted course, such that the student would be given a set of floppy disks and would come back a few months later only for his exams! We want rather to provide students and teachers with software tools that **enhance** and **complement** teaching practices as they exist in a traditional environment: formal lectures; exercises in small groups; individual computer assignments.

This paper presents elements that guided us during the elaboration of this **laboratory:** basic ideas, dreams and pedagogical choices. We describe the laboratory as it is today and we try to analyze its potential for students, teachers and software authors. We conclude on possible further developments.

1. The Arcade laboratory as we dreamt it

Some disciplines (e.g., Chemistry, Physics) use specially equipped "laboratories" where students come to practice a number of activities. These laboratories are places for experimentation and research, where students study in more details concepts introduced in the classroom. Heavy equipements are permanently set up in specific rooms: electrostatics, hydraulics,...

Why not develop our own "**laboratory**" to experiment with algorithms?

The design of this laboratory has been inspired by other well known models:

The **City of Science and Industry** of La Villette (Paris, France) is a cultural place for entertainment, information and learning. Numerous themes are illustrated by activities distributed in huge rooms opened to the public. The visitor defines freely his own itinerary in his quest for a better understanding of selected topics. Active participation in various manipulations and experiments is privileged.

Quite different are the **amusement arcades**, where electronic games are at the disposal of players who go from one game to the next as they like. Freedom of circulation, ease of access to the machines and games variety are attractive elements for the player, who may exercise his reasoning and observation capabilities, as well as his reflex.

Finally there are the popular **adventure games**. If we put aside their anecdotal aspects, we observe that they use an interesting approach: a problem must be solved; the player has to find solution elements, select and combine particular ones in order to progress (possibly) towards the solution. There is no formal guidance, but a trial and error approach requiring reasoning by deduction and by analogy.

We wanted our laboratory to be a **synthesis** of these models: offer an attractive interface, involve the user in numerous and various activities that he may select and organize freely.

2. Pedagogical choices

We have chosen to define small, easy to use components aimed at very specific topics, since we considered that a balance had to be found between the richness of an environment and its convenience. Each tool is based on our practical experiences in teaching algorithms, programming and other computer science topics. We also propose innovative or unusual activities.

Our goal is more to assist the student in his learning task, than to teach him particular topics. We put the emphasis on the student's activity, not on the teacher's.

We wanted our students to have a great freedom to choose and organize their itinerary. We also wanted them to participate very actively, often playfully. Freedom and participation are characteristics often associated, each one depending stongly on the other. In the context of teaching to adults, they are necessary because they prepare students to assume professional responsibilities while respecting their personalities.

Freedom and ease of use are facilitated by presentations and operating modes **homogeneity** in the Laboratory. The student does not have to learn new conventions, interfaces or expression modes for each different activity. Help screens provide operating informations, suggested exercises,...

To promote a strong student participation, three aspects are privileged: manipulations, games and active observations.

Through **manipulation**, an object can be demystified, mastered and appropriated. This is made possible by the great proximity of the object and the freedom to act on it. The manipulated object is thus taken apart, controlled, known. The student attempts, through hypotheses he formulates, to understand the object structure, mechanisms and uses. Manipulation has a non negligible role to play when adults must be trained to apprehend concrete realities, to make decisions and to produce concrete results. It can take various aspects and be integrated in activities such as observations, games, ...

To place a student in a **game** situation is a way to involve him. Play is an important factor to increase motivation which is an essential condition for learning; the practice of a game requires the application of concepts and techniques and necessitates a very strong implication from the student. A well chosen and well defined game thus opens new learning possibilities. Business games and flight simulators have become classical training tools.

The teacher, with his traditional chalk and blackboard, is poorly equipped to present dynamic phenomena. We want to give students the opportunity to observe a program execution as they would observe a spectacle. But we especially want him to be able to act on the visualization parameters, so that he becomes involved in an **active observation**: choice of data, execution speed, point of view, etc. Complementary exercises ask to simulate execution, to reproduce a particular execution, ...

Play, manipulation and observation concepts have common points: they all evoke freedom of action, concentration, approaches based on thought and curiosity.

3. The laboratory as it is today.

The Arcade laboratory has been developed on Apple Macintosh. Its software components have been written in Pascal and integrated with HyperCard.

3.1 The laboratory structure

On the first screen appears a **village** where each building has a name that informs on the topics covered in this building. A simple mouse click on a building lets enter it.

The buildings:

+ A special "Information" building provides a first-time user with explanations on the main conventions used.

+ Every other (standard) building contains a single room. All rooms look the same. In each room appear the names of various proposed activities (along with icons whose symbolism becomes more meaningful as activities are practiced).

+ The "languages" building is devoted to traditional programming practice. The student can access files containing the texts of several algorithms studied in other buildings. He may thus read their code, observe their normal or step by step execution, produce modified versions, ...

+ A "library" building gives access to various on-line manuals describing the Arcade project and its various components.

A typical activity:

The selection of an activity leads to a presentation screen, similar for all activities. It always contains:

+ A brief description of the activity, sufficient to get an idea of its purpose. It is thus possible to explore quickly a room to see what resources it contains. It also gives the regular user a confirmation that this activity is effectively the one he is interested in.

+ An icon that is used to start the activity. At this point, the particular operating mode of the supporting software becomes operative. When the user terminates the activity, he returns to this same presentation screen. (A special icon is used to return to the containing room and another one to return to the village screen.)

+ Other icons that give access to help screens, suggested exercises, a users' mail, ...

3.2. The laboratory content

The following presentation describes the various buildings independently of any particular pedagogical progression.

Specifications Building

From the first day of his programming lectures, the student hears about "specifications"; however, in his daily practice, he works only on small algorithms, and he has no real opportunity to understand the necessity of specifications, nor the underlying difficulties. To provide such an opportunity, we designed a **role-playing game**, to be used by three players connected through a computer network.

The pedagogical objective is to illustrate the importance of specifications in the analysis of a problem to be solved by computer. For this, the game involves the student in an experience that puts in evidence the necessary methodical behavior of an analyst-programmer going through the various steps from a problem statement to a possible solution.

The specific game objective proposed to the student is to create one page of comic strip; the programming activity is thus transposed in a different domain, carefully selected for its analogy with the programming domain.

Here is a brief description of each role: the **customer** defines the synopsis and the main actors (analogy with the requirement specifications definition); the **scenarist** builds a scenario from the synopsis and defines the page layout (analogy with the analysis process); the **artist** produces the comic strip page from the scenario and the page layout; he may use a predefined drawings library (analogy with the programming activity).

Sorting building

This building offers a diversity of approaches on this theme, rich by its great variety of algorithms:

- An **animated movie** centered on the selection sort and defined with a double pedagogical objective: to teach and to entertain. The fantasy used is the reorganization of a train in a railway-station; this activity is formulated by a musical dialogue between a student-analyst and a teacher-programmer.

Several activities are proposed: observation of the process as it could take place in real life; analysis of the various components of this process; coding in a particular programming language; observation of the program execution (optionally, step by step).

The student may evaluate his understanding by completing a film strip built from scenes of the film.

- A set of tools that can be used to **sort objects** represented by cubes (with invisible values): move a cube to a special position; exchange two cubes; find the cube with the smallest value; etc. The student's goal is to place the cubes in increasing order of their values. For this, he may manipulate freely and (hopefully) rediscover by himself classical algorithms, or he may try to reproduce exactly algorithms introduced in the classroom. Most non recursive classical sorting algorithms can be effectively reproduced: selection sort, insertion sort, bubble sort, Shell sort, merge sort, etc.

- A **visualization of execution** of the main internal sorting algorithms: selection sort, bubble sort, insertion sort, Shell sort, Heapsort, Quicksort,... The student selects an algorithm, defines data characteristics (random, ordered, inversely ordered, or equal values) and the number of elements, or even inputs his own values. He chooses the execution speed. He may at any time start again, or select another set of data or another algorithm. Each sort is executed on a graphical representation: data elements are shown as sticks with lengths proportional to their values. The student may also observe histograms that compare the various algorithms, according to their execution speed and their number of comparisons and assignments.

Recursivity building

The student is offered here a concrete approach of recursive mechanisms, often poorly understood in the classroom. There is mainly:

- Two classical games (**Baguenaudier, 8 Queens**) that let practice recursive mental mechanisms. Each game is simulated and its size is adjustable (number of rings, number of squares); one can either play under control of the computer, or observe a game played by the computer itself.

- A work on **geometrical drawings** where one must discover the recurrent definition of each pattern (and thus of the recursive program that draws the pattern). The proposed patterns are of varying

difficulties: squares, stars, snow flake, triangles, Peano, Serpinski and Hilbert curves. For each pattern, the following activities are proposed: a structured reading of the algorithm; an observation of the program execution (the student may select the execution speed, the recursivity depth and the observation mode: normal or level by level); an assistance to understanding the recursive description (comparison or superposition of consecutive levels, highlight of a particular level, construction rules); a reinforcement exercise (the student must select, one by one, the drawing components in the order defined by the algorithm: a game aspect is introduced by a scoring system).

Graph and Tree building.

This building provides:

- Tools to manipulate **binary trees**. They allow work on the main algorithms: traversal, searching and updating. The student may increase the level of difficulty by choosing to work on a visible tree with node values displayed, on a visible tree with no apparent node values or on an invisible tree with information on the current node only.

- A visualization of classical algorithms on **graphs** (depth-first and breadth-first searches, topological sort,...). Graphs can be easily created and modified. Standard or internal (data structure) representations of a graph are available and the student may go from one to the other during execution.

Other buildings are being developed: table exploration; pattern matching,...

4. The richness of our laboratory: first conclusions.

We have experimented our laboratory on various publics: secondary schools teachers and students; college, undergraduate and graduate (computer science) students. We shall now present some conclusions about he advantages it may offer to students, teachers and even software authors.

Students:

The attractive presentation of the laboratory and its ease of use induce the student to **explore**.

The structure of the laboratory and the facility with which components can be located and activated provide an **easy** and **fast** access to topics of interest.

The student may use his total freedom of movement to create his **own** itinerary, so as to satisfy his own motivations. Learning depends on the attitude of the learner: his interests at a particular time, his curiosity. Some students prefer to wait until a theme has been introduced in the classroom before experimenting with the related software components; others, on the contrary, will spend a lot of energy on exploration of topics not yet introduced by the teacher. We allow both approaches.

The important number of offered **topics** and their diversity let the student choose according to his motivations and thus place him in proper learning situations.

The number and the variety of proposed activities on a given theme offer a variety of **approaches**, of points of view. This is always a richness for the learning process and facilitates a mastering of the subject; a child, for instance, learns to read through different approaches: reading, talking and listening. These approaches are complementary, they all contribute something by themselves, and they can also be considered as pieces of a jigsaw puzzle. This idea is well illustrated by the variety of activities proposed in the laboratory on the sorting theme.

Each topic can be approached at a variety of **levels of difficulty.** Some activities are at a beginner's level, others require a finer understanding of concepts. The same activity can be practiced at different levels, depending on prior knowledge of the subject. Consider for instance the visualization of sorting algorithms: if a student has no particular knowledge of these algorithms, he can observe their global behavior, note their differing speeds, and then may want (or not) to study these algorithms in details; if he has an already good knowledge of the subject, he may verify that the observed behavior confirms the known characteristics of each method. The suggested exercises can also be of varying difficulty.

The freedom of activity offered by each component facilitates an **active** approach.

The notes, suggested exercises, operating manual and users' mail that accompany each component are there to **help** in case of difficulty or hesitation.

Teachers:

As shown by experiments made by various colleagues, there are different ways for the teacher to take advantage of the activities proposed in our laboratory :

- use the presentation of some concepts as an **idea** to improve his own pedagogy (for instance, the

presentation of the execution of a recursive algorithm as a series of independant views showing the work done at each level).

- **illustrate** directly his lecture by using one of the sotfware component. Encouraging experiments have been done in this direction. The visualization of sorting algorithms, for instance, was used, in a presentation to secondary school students, to illustrate easily the existence of a variety of algorithms and their dynamic behavior.

- define **new kinds** of practical programming assignments. For instance, he may use the Geometric Drawings component to ask students to observe the way a pattern is drawn and write a program that produces the pattern in exactly the same order, to observe the drawing elements belonging to each separate recursivity level and to modify a standard recursive program so that it produces a similar kind of level by level output.

- **guide** the students at specific times toward particular components so that they may have a first approach of the concepts, reinforce their understanding of a domain partially presented in the classroom, observe different approaches, complementary or possibly opposed, of a single topic.

- consider the laboratory as a self-service tool that can be **referenced** to as one would make a reference to a textbook or to a training situation.

Software authors:

The laboratory provides a nice-looking, confortable and pleasing environment where each software component thus appears in a **good light**.

The existence of other components, particularly on the same theme, gives an **implicit message** to the student: each particular component has no ambition to offer a complete, absolute answer to all problems related to this theme. It represents only one approach which, considered in this context, can be better appreciated.

The software components which contribute to the understanding of a single theme benefit together from this association. It is possible to define a component that aims only at one **particular aspect** of a problem, knowing that other components will take in charge other aspects. The author is thus not obliged to cover all aspects of a question.

5. Conclusions

Teaching practices and the Arcade laboratory

Freed from nearly impossible tasks that are much more easily done through the activities offered in the laboratory, the teacher may find more time to concentrate again on his primary pedagogical role.

The content of the laboratory is not based on any particular pedagogical progression. It may thus be used by all students, even if their teachers follow different pedagogical approaches.

Tool development

We now work on extracting common points from the diversity of our products: program structures, visualization and interaction modes, development techniques. A first set of components has already been isolated. Our goal is to provide further tools that will help software authors to develop more easily new products in the same Arcade spirit.

Evolution toward other Computer Science specialties

The actual content of the Arcade laboratory is related to algorithms only because of the specific set of proposed activities.

The laboratory could easily cover other themes: operating systems, data bases, artificial intelligence, ... The laboratory structure can be extended without in depth modification. We only have to add new buildings, and possibly regroup them in districts. The main task remains however to imagine activities and develop the corresponding software components for the new computer science specialties.

Such an extension is now being considered. Interested teachers, who regularly teach these other domains, will of course be asked to contribute. They usually know well what parts of their lectures present specific difficulties. However most of the time they do not really imagine what kinds of tools would help them or their students. We have had numerous opportunities to observe that, when they assist to a demonstration of our laboratory, they suddenly begin to express their ideas as a number of very precise activities that could easily be integrated in our Arcade laboratory.

Bibliography

Brown M., Sedgewick R.: "Technics for Algorithm Animation", IEEE Software, Vol 2 n°1, January 1985

Guéraud V., Peyrin J-P. "Un jeu de rôles pour l'enseignement de la programmation" Colloque sur la Didactique de l'Informatique, Paris, Sept. 88

Guéraud V. "Un jeu de rôles dans le laboratoire Arcade: une autre façon d'enseigner la programmation" Thèse de Doctorat de l'Université Joseph Fourier - Grenoble I, Février 1989

Guéraud V. "A Role-Playing Game to learn (programming) methodology" CATS'90, International Conference on Computer Aided Training in Science and Technology, Barcelone, July 1990

Liem I. "Visualisation de l'exécution des programmes pour l'enseignement de la programmation" Thèse de Doctorat de l'Université Joseph Fourier - Grenoble I, Sept. 1989

Peyrin J-P., Liem I. "Visualisation of program execution: pedagogical reflexion and courseware realisation" CATS'90, International Conference on Computer Aided Training in Science and Technology, Barcelone, July 1990

Peyrin J-P., Cagnat J-M., Guéraud V., Liem I., Painvin S. "Un laboratoire pour l'enseignement de la programmation" Colloque sur l'évolution de l'outil informatique à l'université, Poitiers, Sept. 1988

****

SENIOR SEMINAR-- continued from page 36

A system that centralizes scheduling time and room assignments for internal and external events at Beaver College. The system also generates a monthly calendar of events.

2) Employee absence tracking system

User -- Smithkline Beecham
Clinical Laboratory

The system maintains employee absence records and distributes awards for employees with good attendance records.

3) MIDI database of Musical Effects
(Musical Instrument Digital Interface)

User -- Jampyr (a local musical group)

The system integrates band equipment for the creation of electronic music. Such things as effect settings, event timers, synthesizer patches and sequences are stored.

4) Inventory and control system for Electrical Equipment

User -- Prudential Insurance Co.

Manages electrical hardware inventory.

5) Network PC management System

User -- SEI (local Micro Computer firm)

A system to keep track of the configuration of SEI's computer system installations for the technical support department.

6) Bid analysis system for bids on electrical work.

User -- Somerset County, NJ.

The system accepts all relevant information concerning bids on electrical work needed by Somerset County NJ and given the criteria established in state law, selects makes recommendations on the assignment of contracts.

7) Record keeping system for students

User -- Cosmetology Department, EMCO-AVTS
(a local technical school)

The system maintains records on the progress of students in their areas of study and prints reports for the Pennsylvania state board of Cosmetology.

8) Circulation Control System

User -- Atwood Library, Beaver College

A circulation control system that maintains records of all circulating books.

9) Library Catalog system

User -- Applied Technology Library,
UNISYS Corporation

An automated catalog for a technical documents library.

10) Athletic Department Scheduling System

User -- Athletic Department, Beaver College

A system that maintains schedule for coaches, teams and team rosters for all athletic teams at Beaver College.

11) Customer service system

User -- Atlas Exterminating Co.

The system contains information of active customers. Store type of treatment, warranties in effect and times of scheduled appointments.