Third Misconceptions Seminar Proceedings (1993)

Paper Title: Designing Computer Exploratory Software for Science and Mathematics
Author: Teodoro, Vitor Duarte

Abstract: The aims of this paper are: 1) to characterize computer exploratory software; 2) to identify the roots of this kind of software; 3) to present a model to design computer exploratory environments for science and mathematics; 4) to discuss some of the basic issues of the model; and 5) to analyze some programs developed in the framework of the model. The model is based on findings in learning and in recent developments of computer graphic environments. It assumes that: 1) learning is a process of enculturation, a process of becoming familiar with ideas and representations; 2) exploratory software should be integrated with other resources; 3) exploratory software should allow direct manipulation of concrete-abstract objects and the exploration of multiple representations of a phenomenon.

Keywords: science education, mathematics education, computers, simulations, exploratory environments, learning
General School Subject:
Specific School Subject: science & math
Students: high school

Macintosh File Name: Teodoro - Computer Software
Release Date: 9-15-1994 I

Publisher: Misconceptions Trust
Publisher Location: Ithaca, NY
Volume Name: The Proceedings of the Third International Seminar on Misconceptions and Educational Strategies in Science and Mathematics
Publication Year: 1993
Conference Date: August 1-4, 1993
Contact Information (correct as of 12-23-2010):
Web: www.mlrg.org
Email: info@mlrg.org

A Correct Reference Format: Author, Paper Title in The Proceedings of the Third International Seminar on Misconceptions and Educational Strategies in Science and Mathematics, Misconceptions Trust: Ithaca, NY (1993).

Note Bene: This paper is part of a collection that pioneered the electronic distribution of conference proceedings. Academic livelihood depends

upon each person extending integrity beyond self-interest. If you pass this paper on to a colleague, please make sure you pass it on intact. A great deal of effort has been invested in bringing you this proceedings, on the part of the many authors and conference organizers. The original publication of this proceedings was supported by a grant from the National Science Foundation, and the transformation of this collection into a modern format was supported by the Novak-Golton Fund, which is administered by the Department of Education at Cornell University. If you have found this collection to be of value in your work, consider supporting our ability to support you by purchasing a subscription to the collection or joining the Meaningful Learning Research Group.

----- -----

Designing Computer Exploratory Software for Science and Mathematics

Vitor Duarte Teodoro
Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa, 2825
Monte de Caparica, Portugal

Abstract: The aims of this paper are: 1) to characterize computer exploratory software; 2) to identify the roots of this kind of software; 3) to present a model to design computer exploratory environments for science and mathematics; 4) to discuss some of the basic issues of the model; and 5) to analyze some programs developed in the framework of the model. The model is based on findings in learning and in recent developments of computer graphic environments. It assumes that: 1) learning is a process of enculturation, a process of becoming familiar with ideas and representations; 2) exploratory software should be integrated with other resources; 3) exploratory software should allow direct manipulation of concrete-abstract objects and the exploration of multiple representations of a phenomenon.

Keywords: science education; mathematics education; computers; simulations; exploratory environments; learning.

WHAT IS A COMPUTER EXPLORATORY ENVIRONMENT?

Some ideas are difficult to verbalize. The concept of "computer exploratory environment" seems to be one of those ideas. As with many other concepts, to build this one into the cognitive structure each of us needs to know and, more important, *to be familiar with* the use of computer exploratory environment. As one becomes more and more familiar with this kind of software environments, the concept becomes more precise and accurate.

Taylor (1980) suggested that all instructional uses of computers fall under three modes:

Table 1: Taylor´s (1980) modes of instructional uses of computers.

tutor	the student is taught knowledge by the computer
tool	the computer assists the student in the learning process but does not direct his/her efforts
tutee	the student teaches the computer

This classification has been used since then by many authors as a basis to classify the role of different educational software packages. The computer as *tutor* can be seen as a transposition of the classic role of the teacher to the computer. The computer as a *tool* is a transposition of the role of a pen or a calculator to the computer. The computer as *tutee* is a *new kind of educational environment* where teaching the computer is seen as a powerful aim that can allow students to get a deep awareness of how knowledge is built. Papert (1980) wrote:

«In my vision, the child programs the computer and, in doing so, both acquires a sense of mastery over a piece of the most modern and powerful technology and establishes an intimate contact with some of the deepest ideas from science, from mathematics, and from the art of intellectual model building.» (p. 5)

A *computer exploratory environment* combines two of Taylor's categories: *tool* and *tutee*. It is a tool because it can be used to help students think about one or more knowledge domains, doing tasks that could not be done without it or that would take more time than reasonable. It can be used in a *tutee* environment, because students can "teach" the computer how to do things (e.g.: build a mathematical model of an object that falls on the earth or on any other planet; move an angle that produces a graph; etc.) and then get feedback about the reasonability of what they have done.

In a computer exploratory environment there are three possible *kinds of objects*:

Table 2: Kinds of objects in a computer exploratory environment.

Type I: real objects	objects that *represent real objects* (e.g.: a planet; a car; a particle) with more or less perceptual fidelity
Type II: conceptual objects	objects that are "pure" *conceptual objects* (e.g.: a variable; a vector), that have no perceptual fidelity
Type III: relations between properties	objects that represent *relations between properties* of other objects (e.g.: a graph; an equation)

Type II objects of a computer exploratory environment acquire a new status, as compared with traditional modes of concept learning. Hebenstreit (1987) argues that they are a new genre of objects — *concrete-abstract* objects:

«L' "objet" sur lequel l'usager agit pendant une simulation est "concret" en ce sens qu'il réagit aux actions (par l'intermédiaire du clavier ou d'une souris) comme le ferait in objet réel, mais cet objet est cependant abstrait car si son comportement apparait sur l'écran de l'ordinateur, il ne peut cependant être vu ou touché comme le serait un objet concret.» (p. 1)

These *concrete-abstract objects* are *concrete* in the sense that they can be seen and manipulated as real — on the computer screen — and *abstract* in the sense they are physical and mathematical constructs.

One of the most important features of a computer exploratory environment is that it should allow the user to explore the relations between the different kinds of objects, in real time or with a different time scale, if necessary (faster or slower, depending on the nature of the problem under investigation). The exploration is done under the *full user's control*, not the computer's. This means that the computer does not behave as a video projector that presents a previously defined sequence of images but as a device that is fully manipulable by the user, who *must establish strategies* to

explore what s/he wants to explore, choose what, when, and how to visualize things.

Another important characteristic of exploratory software is the possibility of *linking multiple representations*. This is, perhaps, the characteristic that most teachers are interested in. Multiple representations can facilitate the process of creating meaning from representations if we assume that *meaning is created essentially when students relate different representations*. We only understand something if we can establish relations between different representations of phenomena.

The issue of representations is crucial in science and science education. Science can even be defined as a means for constructing and improving representations of the world (Rouse, 1987).

The Educational Technology Center (1988) points out that multiple representations is one of the most fruitful applications of computer technology, with particular relevance in science and mathematics, domains where knowledge has more than one mode of representation[1]. The Educational Technology Center presents two reasons for this:

«First, different representations of a complex idea (for example, a ratio, an algebraic function, or a concept such as heat) emphasize different aspects of the idea and afford different sorts of analyses. (…) Understanding the strengths and weaknesses (Bliss. et al., 1992) of various representations and the relationships among them helps mathematicians and scientists select and apply them efficiently in solving problems. Second, students differ in their ability to understand and use particular representations. (…) Thoughtfully designed computer software can present multiple, dynamically linked representations in ways that are impossible with static, inert media such as books and chalkboards.» (p. 10)

For example, if we want students to see, in a reference frame, how does the coordinates x and y of a moving object change with time when the object

[1] Thanks to computers and their powerful graphical capabilities, a new field of science has emerged in the last two decades: computer visualization or computer representation. One of the aims of this field is to create graphical representations of complex features of real phenomena or of complex mathematical constructs, such as fractals.

falls, exploratory software should enable students to see the following representations — *simultaneously* and in *real time* (Figure 1):

1 the object moving;

2 a stroboscopic representation of the motion;

3 the vertical and the horizontal components of the positional vector of the object;

4 the graphs of position in the y and in the x axis as a function of time.

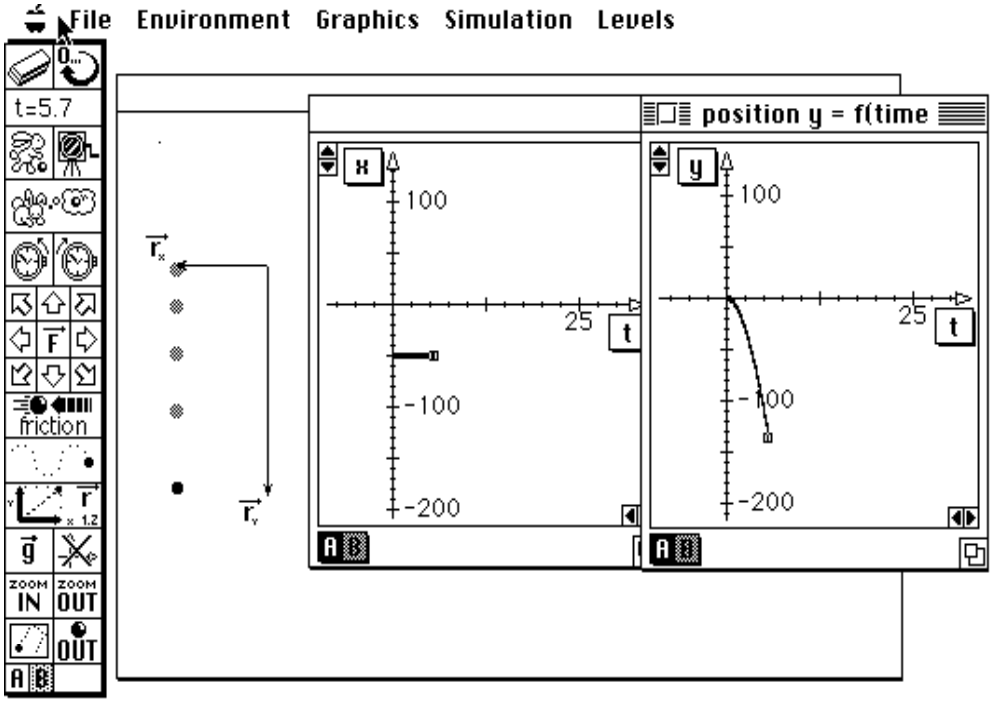

Figure 1: Multiple representations of a rectilinear vertical motion of a particle: stroboscopic representation, vertical and horizontal components of the positional vector, graph of position in the y and in the x axis as a function of time. Done with NEWTON (Teodoro, 1992).

Analyzing further the idea of multiple representations, exploratory software should allow students to *start from the graph* (for example, sketching the graph with a mouse) and *then obtaining the corresponding motion* (Figure 2). Another possibility of the software is that it should allow the student to write an equation and then obtain the motion of an object that behaves according to the equation and, simultaneously, obtain a graph (Figure 3). Linking equations of motion to graphs, graphs to motion in real

time, and motion to equations, can be a powerful process to derive meaning of each of the representations and of all of the representations of the same phenomenon.

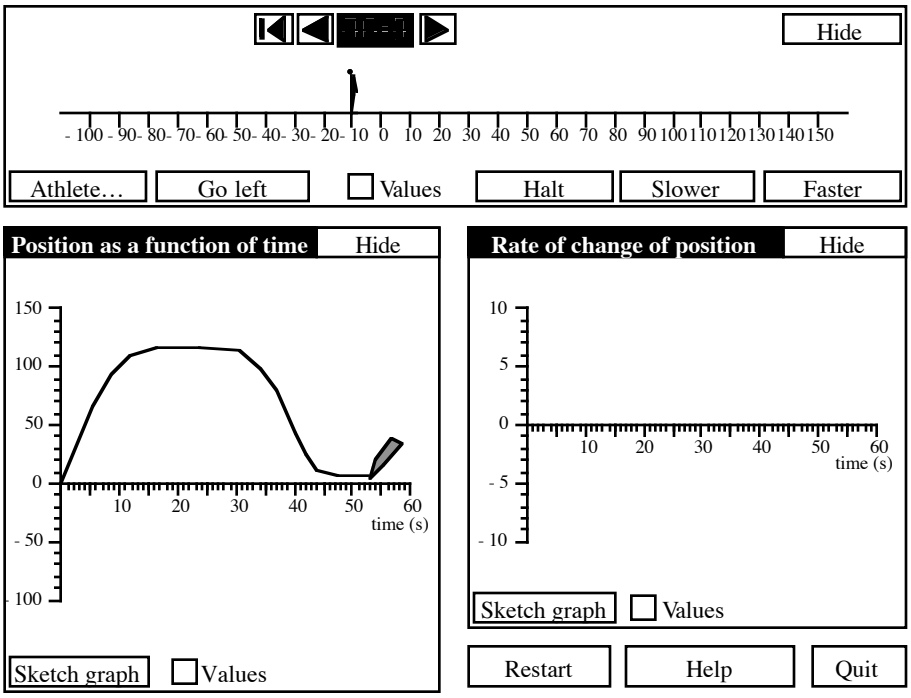

Figure 2: From graph to motion. The graph was sketched with the mouse. The motion of the athlete can be obtained from the graph of position, or from the graph of rate of change in position — graph on the right (done with CHANGE, Teodoro, 1992).

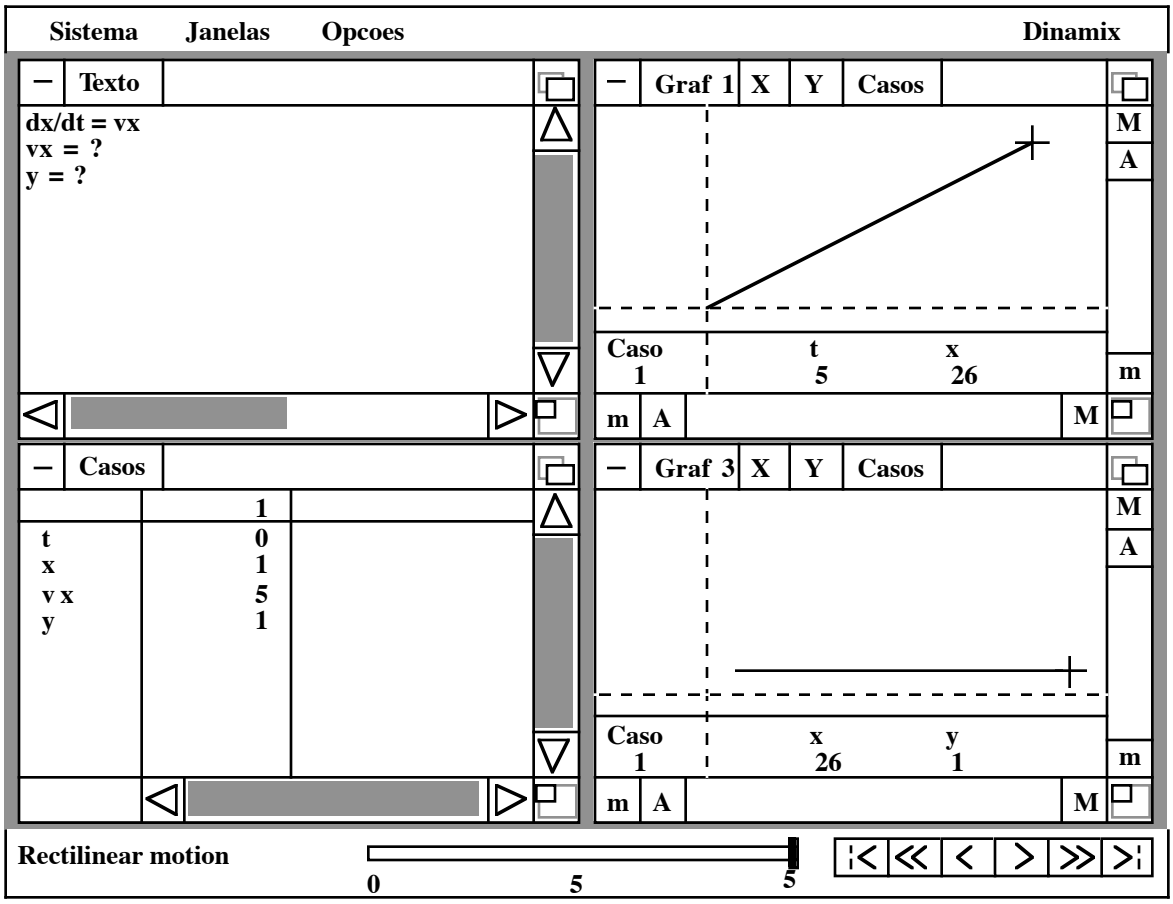

Figure 3: After writing an equation (« dx/dt=vx »), the user can move an object (a stroboscopic representation is shown down right) and see a graph of position as a function of time (top right). Done with DINAMIX (Lobo, 1991)

ROOTS OF EXPLORATORY SOFTWARE

Papert and LOGO

Since Papert wrote Mindstorms (1980) learning with computer-based exploratory environments in science and mathematics has become one of the predominant views of computers in education. Papert was the first to argue that a computer is a tool that students can use to *change the nature* of conceptual objects:

"Stated most simply, my conjecture is that the computer can concretize (and personalize) the formal. Seen in this light, it is not just another powerful educational tool. It is unique in providing us with the means for addressing what Piaget and many others see as the obstacle which is overcome in the

passage from child to adult thinking. I believe that it can allow us to shift the boundary separating concrete and formal. Knowledge that is accessible only through formal processes can now be approached concretely. And the real magic comes from the fact that this knowledge includes those elements one needs to become a formal thinker." (p. 21)

According to Papert, exploratory computer tools are seen as tools to overcome the boundary between lower and higher cognitive stages. This can be done because computers allow users to approach concretely what without the computer can only be approached in a formal way.

Papert's view is a compromise between the use of a computer as a new kind of tool in education and a more traditional view of the computer as an object to be programmed. I see this compromise as rooted in the personal history of Papert – a mathematician and a computer scientist who after some work with Piaget, became interested in how students learn.

Papert's work was based on a computational metaphor — *programming* — that is too elementary to allow the exploration of many fields. Programming, even in a high level language such as LOGO, uses primitives that are too "primitive" to allow meaningful exploration of most scientific ideas. With a programming language it is possible to explore most scientific ideas but the programming language itself behaves as a mediator that does not have the properties of the scientific ideas that we want to explore. For example, if we want to explore how velocity of an object changes with time under certain circumstances, we must have direct access to a representation of velocity, such as a vector. With LOGO or any other programming language, that can only be done with a big programming effort, because programming languages are not domain specific and only have general primitives and procedures.

Exploratory environments, unlike programming languages, are domain specific. What a user can do with an exploratory environment depends on the domain. On the one hand, this gives very powerful primitive actions, such as showing a velocity vector just by clicking the mouse; on the other hand, however, it narrows the range of the capabilities of the software. But this is usually not a real problem because of the domain specificity of each exploratory environment.

Constructivism

Papert's work and exploratory approaches to software are in accordance with a major shift on the dominant view of how humans learn. Decades of work by Piaget and collaborators, and many others (including critics of Piaget's work, such as e.g., Novak, 1977), have given clear evidence that learning is an active and constructive process. The *delivery paradigm* of education doesn't seem to be accepted by educators any longer: it is now widely recognized that the mental activity of the learner and his/her own experiences are the major factors that support learning.

How can we characterize a constructivist view of learning? Novak (1990) claims

"(…) that human beings all have an enormous capacity for meaning making and the use of language to construct and communicate meanings. I seek to conflate issues that deal with the nature of knowledge construction into the issues that deal with the psychology of meaning making. In both cases, I see human capacity for meaning making and the nature of that process as the 'bottom line'. What really counts, in my view, is how to empower human beings to optimize their phenomenal capacity for meaning making, including their awareness and confidence in processes that are involved. This capacity for meaning making is what I refer to as human constructivism." (p. 20)

Accordingly to Forman and Pufall (1988) constructivism embodies three properties: *epistemic conflict, self-reflection* and *self-regulation*:

1 *Epistemic conflict* involves two knowing systems: "These systems may originate in different individuals, and it may be that in early development we are more dependent on externally induced conflict than we are subsequently. Whatever the source of conflicting epistemic stances, if there is a resolution it is within the individual experiencing the conflict, that is, it is an individual construction. If the resolution is developmental, in the strict sense, it means constructing a new way of thinking about reality and is marked by logical necessity." (p. 236)

2 *Self-reflection* is construed as a response to conflict.

3 *Self-regulation* is the developmental restructuring of thought.

These authors point that the self-organizing properties of the knower allow him to abstract structure from action, "not necessarily with conscious processes" (Forman & Pufall, 1988, p. 236).

More recently, the educational community is recognizing that the constructivist view of learning must be combined with the recognition of the consequences of the fact that learning takes place in social environments, where the interaction between students, teachers and students, and all other social actors are crucial in the process of making sense, that is, the process of generating knowledge. As Brown, Collins and Duguid (1988, p. 7) wrote "learning is, we believe, a process of enculturation".

Under a constructivist view of learning, *abstracting structure from action* is not necessarily a conscious process. Then, in a certain way, learning can be viewed as a process by which *new knowledge is not necessarily new explicit knowledge* — we can know without *knowing*. This idea can be re-stated by defining *learning as a process of becoming familiar with knowledge*, not as constructing new explicit knowledge.

Learning most of the scientific and mathematical ideas at secondary school can then be seen as a process of becoming familiar with them. True understanding of an idea is most of the times a strong degree of familiarization with the idea. As Schank (1986, p. 5) pointed out, understanding «is not an all-or-none affair. People achieve degrees of understanding in different situations depending upon their level of familiarity with those situations».

Exploratory software must allow students to get a strong degree of familiarization with the basic ideas of the domain being explored. With exploratory software, students can see many situations, explore what happens in different conditions, discuss what happens if they change conditions, etc.; i.e., they can become more and more familiar with the ideas, the consequences of ideas and representations of the world. When they become more familiar with new ideas and new representations, they can establish more meaningful relations with ideas they already have. Exploratory software can be a major way to foster familiarity with new ideas.

Graphical and direct manipulation computer interfaces

We can also identify another important root of exploratory software: the graphical user interface, introduced by Apple in the early 80's and now used

in almost every computer environment such as Windows and OS/2. These graphic environments allow computer illiterate users to become almost immediately involved with computer software, by *exploring* icons, buttons, pull-down menus and other screen objects. When Taylor (1980) suggested that all uses of computers in education should be seen either as tutor, or tool or tutee, it was really difficult to see how the computer could be used as a tool — most computer tools were so difficult to use that nearly all potential users were intimidated by them (e.g., word processors and spreadsheets were still in their infancy). Now, with graphical interfaces, it is possible to have computer software that almost dispenses with manuals or specific instructions on how to use it.

Strongly related to graphical user interfaces is the concept of *direct manipulation*, proposed by Shneiderman (1983). This concept is based on the idea that most actions on the screen need not be mediated by any written language. Screen objects should have properties and reaction similar to those of real objects. For example, if we want some screen object to change its position on the screen, we only need to "hold" that object with the mouse and move it to another position. The concept of direct manipulation is crucial to the design of exploratory environments but it raises problems when it refers to "conceptual objects": if conceptual objects, such as a vector, are human constructs, what does it mean to directly manipulate something that does not exist as a real object?

This problem shows that exploratory software can be another source of misconceptions in science and mathematics. We must, then, be very careful about the use of direct manipulation in regard to conceptual objects.

A MODEL TO DESIGN EXPLORATORY SOFTWARE

In the above sections I tried to define exploratory software and identify its roots. I will now try to outline a model for the design of this kind of software. This model is based on reflections aroused by the development of a set of titles of exploratory software in the following domains: Newtonian mechanics, properties of chemical elements, semi-quantitative study of functions and rates of change, representation of geographical characteristics of a country, modeling with differential equations, trigonometry and properties of triangles, electrical fields, descriptive geometry, and heredity.

Figure 4 shows the structure of the model. It has two lines of approach, one of which is *methodological* and the other *theoretical*.

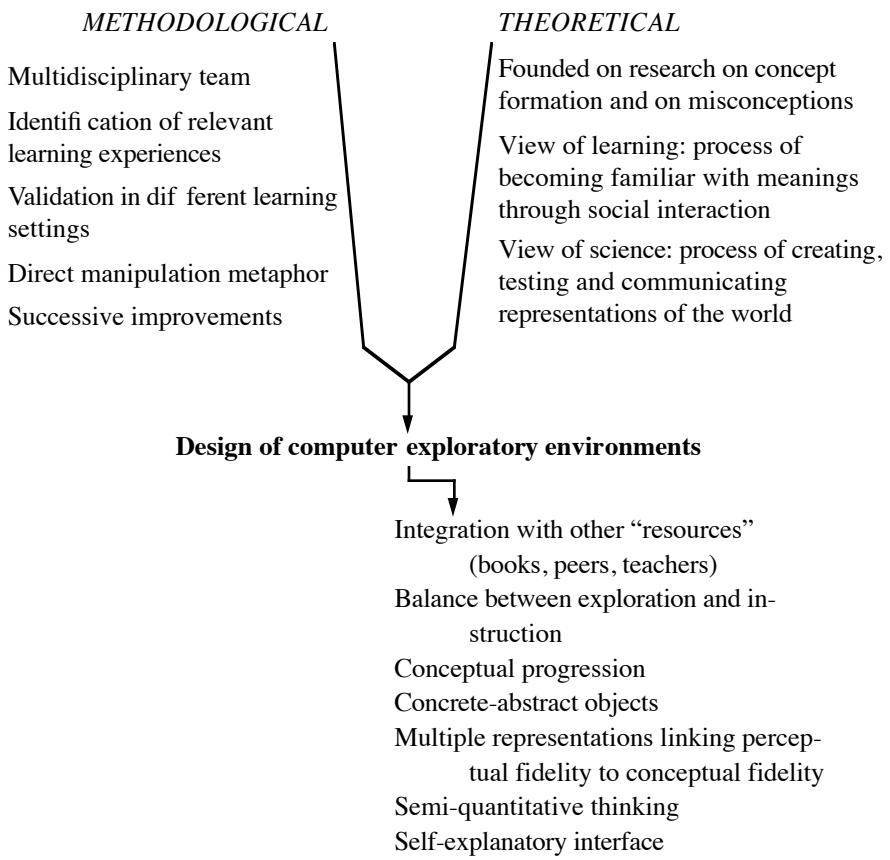

Figure 4: A model to guide the design of computer exploratory environments.

Along the theoretical line, we have three issues to consider. First, the design of exploratory software should be based on research on concept formation and on misconceptions. We have now an enormous body of literature about concept formation and misconceptions, specially in science and mathematics. This research can be taken as basic research to identify relevant learning experiences and sources of difficulties in concept formation. Second, we pose a specific view on learning. Such a view assumes that understanding, as a result of learning, is not a "metaconcept" but a much less ambitious concept. We understand when we are familiar with ideas and representations shared by members of a community. Understanding scientific ideas is, I assume, a process of enculturation. This

process is facilitated when learning occurs in the *zone of proximal development* [2] (Vygotsky, 1978). Third, we also pose a view on science as a process of creating, testing and communicating representations of the world.

Along the methodological line, we point out five issues. First, the development of exploratory software is a team project, involving different specialists: experienced teachers, software designers, programmers, graphic specialists, cognitive psychologists. Second, exploratory software should be designed after the identification of the most relevant learning experiences in a certain domain. A relevant experience is related to the process of concept formation, either because it gives "anchors" to subsume concepts or because it shows a conflictual view with naive thinking. Third, as learning takes place in many different settings, the software should be validated in the different settings where learning occurs. We shall not claim that exploratory software should be designed only for classrooms. With the increasing diffusion of computers, at home, in resource centers, in libraries, etc., students can have experiences with exploratory software in many different places outside classrooms and outside teacher control. Fourth, exploratory software should be based on graphical and direct manipulation interfaces, where the user controls his actions directly, not mediated by written language. Fifth, it is not possible to design good exploratory environments without successive improvements, based on ecological valid research. This research on the software developed should be carried out in real schools with real students and teachers, not in ideal settings.

As "output" of the model, we raise seven relevant issues:

1 Exploratory software by itself has very limited use. Exploratory software should be considered as a part of "learning packages" to foster "learning communities". I think that it is neither possible nor desirable to build exploratory software that is independent from other learning materials, such as books. *Exploratory software can be very powerful but learners can*

[2] Vigotsky (1978, p. 86) defines this concept as "the distance between the actual development level as determined by independent problem solving and the level of potential developlement as determined through problem solving under adult guidance or in collaboration with more capable peers".

only explore what they already know, not what they don't know. Well written materials, combined with good graphics, still have the most important characteristics to present lines of argument. Ideally, exploratory software should serve as a complement to books, allowing students to explore what they read, giving them the capabilities that no book has, unlike well-designed software.

As any other educational material, exploratory software is a *resource for learners*. Programs are like artist tools: tools can help artists but they don't produce art. Only artists do. But exploratory software has a unique characteristic: when well designed it fosters interactions between learners, in particular if students work in pairs or in small groups. Exploratory software can then help the formation of communities of learners that can explore, test and communicate ideas of science.

2 Balancing exploratory learning and direct instruction is a fundamental issue in the design of "learning packages" and in the creation of good learning environments. Research shows that exploratory learning is difficult (e.g.: Bliss et al. 1992, Njoo & Jong *in press*, Veenman et. al. *in press*). Teachers should always bear in mind that *learners cannot explore what they don't know already!* This statement can be seen as a corollary of Ausubel's famous principle: "The most important single factor influencing learning is what the learner already knows. Ascertain this and teach him accordingly" (Ausubel et al., 1978, p. iv).

The balance between exploratory learning and direct instruction must be managed by the teacher and should be one of his concerns. As Chi et. al. (1981) have shown that novice learners tend to be distracted by surface features of display presentations. Exploratory software can increase distraction because surface features of a domain are usually easily accessible.

3 One way of facilitating the balance between exploratory learning and instruction is developing software with increasing levels of complexity, such has NEWTON (Teodoro, 1992) — Figure 5. Progression of complexity is based on research on concept formation. In each phase of learning, each learner can have an environment which is as close as possible to his zone of

proximal development, so as to avoid one of the most crucial problems in educational software design, i.e., *information overload*[3].

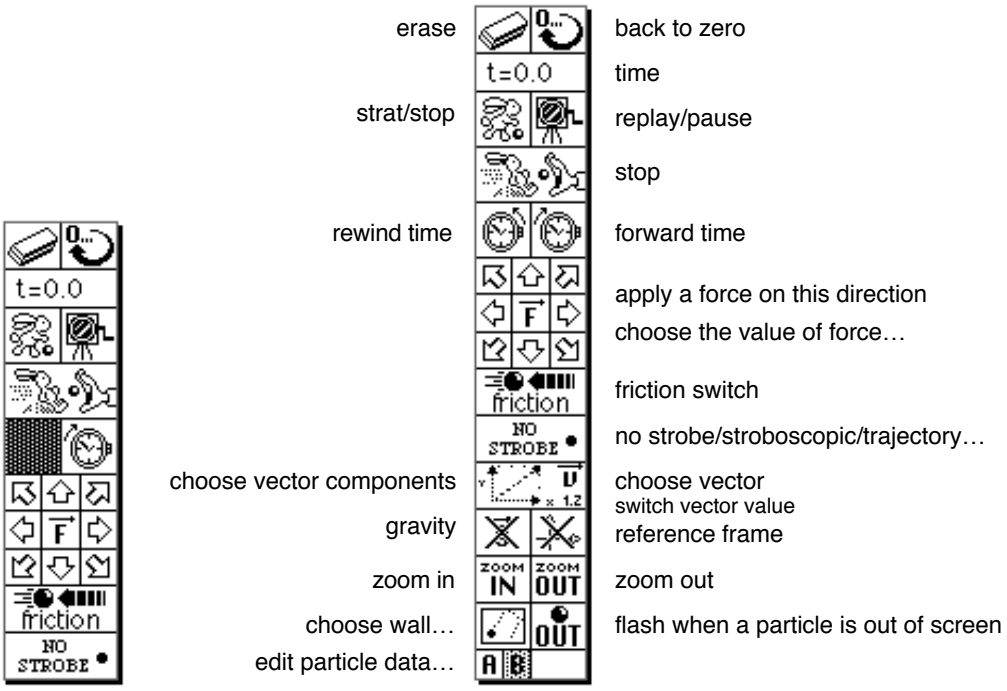

Figure 5: Control panel of the first level (left) and of the fifth level (right) of NEWTON. As the level increase, the complexity of the features of the software increase.

4 Direct manipulation of concrete-abstract objects is, no doubt, a subject that deserves more research. As argued above, this can be a powerful way to explore scientific constructs but it can also be a source of misconceptions or naive epistemologies about them, specially if we take into consideration that novice learners tend to be distracted by surface features.

5 *Multiple representations* is one of the most important features of exploratory software. This feature gives users the possibility to interact with different coordinated representations of a phenomenon. Multiple representations can easily lead to information overload in learners — that is

[3] Access to different levels is done with passwords. This prevents students from using capabilities of the software that are too complex for their conceptual level or for their level of progression in the study of the domain.

one of the reasons why exploratory software should have different levels of complexity.

Until some years ago, exploratory software only had limited capabilities of multiple representations: it was only possible to change conditions in one of the representations (e.g.: a parameter on an equation) and then see the change on the corresponding graph. But now it is possible to fully explore the capabilities of multiple representations. For example, it is possible to change a graph with a mouse and see the corresponding change in an equation. It is even possible to sketch graphs, with a mouse, and then obtain the corresponding phenomena (see Figure 2 and Figure 6).

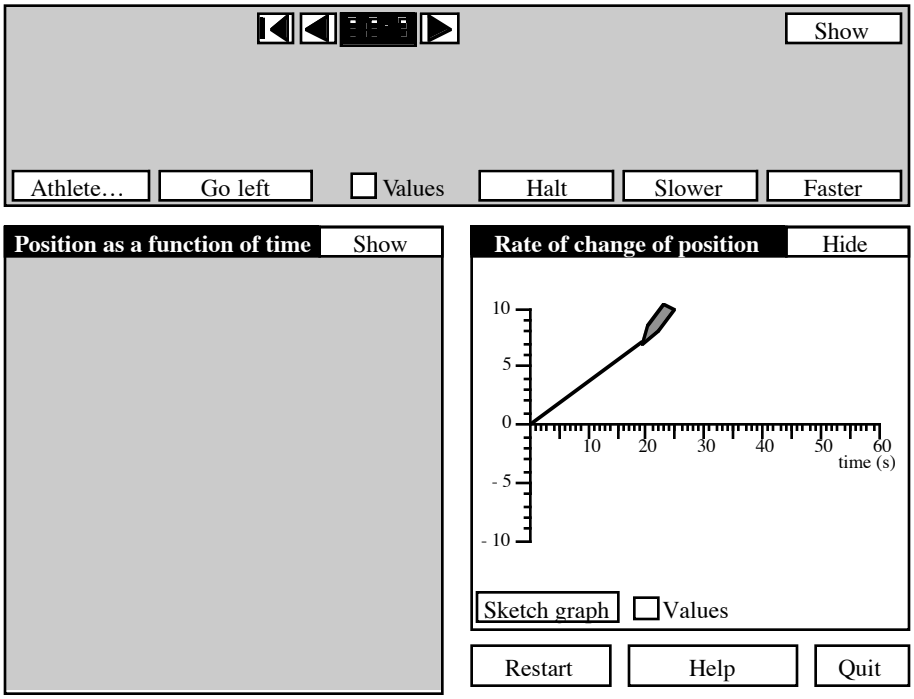

Figure 6: Linking multiple representations. The graph of the rate of change of position (velocity) was sketched with the mouse. The motion of the athlete and the graph of the position can then be obtained from the graph of the rate of change in position. Done with CHANGE (Teodoro et al. 1992).

One design hint that I found helpful is the use of a button that «hides» and «shows» the window with one of the representations. For example, Figure 6 shows a graph of the rate of change in position of an athlete done with the mouse. This graph doesn't have perceptual fidelity. The window

that has perceptual fidelity — the top window, where the athlete can "run" or "walk" — is hidden but it can easily be shown by pressing the button «Show». The same applies to the left window with the graph of the position as function of time. Using this button it is possible to foster discussions about the relations between the rate of change of a function and the value of the function, about the relations between the graphical representation of a function and the phenomena that it represents, etc.

6 Another important issue about the design and use of exploratory software is the relation between semi-quantitative learning and quantitative learning. Semi-quantitative learning is characterized as non-algorithmic learning: "I can discuss relations between variables, without the need of formulae". Quantitative learning is, on the contrary, algorithmic learning: "Relations between variables are expressed through symbolic equations". Some authors consider semi-quantitative learning as the most important issue on learning science and mathematics (e.g. Mestre, 1991). Teachers and researchers are well familiar with students who are almost perfect algorithmic problem-solvers but do not know anything about the meaning of the problem, about the plausibility of the solutions, about the validity of their claims, etc. Well designed exploratory software can bring semi-quantitative learning to a high level because students can concentrate on meanings not on rules or algorithms. For example, the situation presented in Figure 6 is usually discussed only with senior high school students and first year undergraduates, when they are introduced to calculus (derivatives and integration). But the computer program CHANGE can be used with young students (aged 12 or even less), who can then easily discuss powerful concepts, such as rate of change and function, without using complex formal mathematics. Older students can also use this program as a foundation to more formal approaches to the exploitation of those concepts.

7 Finally, exploratory software must have a self-explanatory interface. Students — as other software users — do not like to read computer manuals. If a student who has already been introduced to a scientific domain cannot use an exploratory piece of software about that domain, then the software cannot be widely used.

Text buttons, icons (if used with parsimony), context-dependent on-screen help, self-presented examples, etc., are now easily available for programmers on most computer languages from "resources workshops";

they are essential for the implementation of exploratory software which is easy to use.

A NOTE AS CONCLUSION

After all that has been written above one could expect exploratory software to have the potential to radically change science and mathematics education. This is not true: *the power of any computer environment is not on the computer: it is on the environment* — the cluster of systemic relations among learners, teachers, technology devices such as computers, books, etc.

Naive conceptions of educational change assume that educational change depends on the change of independent variables, such as method of teaching. But decades of educational thinking and practice show that there are no such variables as independent variables. All variables are mutually dependent when we think of learning environments. To think that computers alone (and exploratory software in particular) can change education is an expectation that can block out the most important steps that should be taken to transform schools into learning communities.[4]

REFERENCES

Ausubel, D. P., Novak, J. D., Hanesian, H. (1978) *Educational Psychology, A Cognitive View (2nd edition)*. NY: Holt, Rinehart and Winston.

Bliss, J. et. al. (1992) *Summary Report of the Tools for Exploratory Learning Programme*. London: King's College.

Brown, J. S., Collins, A., Duguid, P. (1988) *Situated Cognition and the Culture of Learning* (IRL Report No. IRL 88-0008). Palo Alto: Institute for Research on Learning.

Chi, M. T. H., Feltovitch, P. J., Glaser, R. (1981) Categorization and Representation of Physics Problems by Experts and Novices. *Cognitive Science* 5, 121-152.

[4] This note was inspired in the final talk given by Bob Glaser and in the talk given by Gavriel Salomon at the NATO-ASI Psychological and Educational Foundations of Technology-Based Learning Environments held at Kolimbari, Crete, July 1992.

Educational Technology Center (1988) Making Sense of the Future. Cambridge, Mass: Harvard Graduate School of Education.

Forman, G., Pufall, P. B. (1988) Constructivism in the Computer Age: A Reconstructive Epilogue. In G. Forman & P. B. Pufall (eds.) *Constructivism in the Computer Age*. Hillsdale, NJ: Erlbaum.

Hebenstreit, J (1987) *Simulation et Pédagogie, une Rencontre du Troisiéme Type*. Gyf Sur Yvette: Ecole Superieure d'Electricité (mimeo).

Lobo, M. S., Teodoro, V. D. (1991) *DINAMIX, um sistema de modelação dinâmica: Manual de Utilização*. Monte de Caparica: Universidade Nova de Lisboa (DINAMIX, a dynamic modelling system: User's Manual) .

Mestre, J. P. (1991) Learning and Instruction in Pre-College Physical Science. *Physics Today*, Sep. 56-62.

Njoo, M., Jong, Ton de (in press). Supporting Exploratory Learning by Offering Structural Overviews of Hypotheses. In Ton de Jong, Hans Spada, Doug Tawne (eds.), *The Use of Computer Models for Explication, Analysis and Experiential Learning*. N.Y.: Springer-Verlag.

Novak, J. D. (1977) An Alternative to Piagetian Psychology for Science and Mathematics Education. *Science Education*. 61, 453-477.

Novak, J. D. (1990) *Human Constructivism: A Unification of Psychological and Epistemological Phenomena in Meaning Making*. Paper presented at the Fourth North American Conference on Personal Construct Psychology, San Antonio, Texas, July 18-21.

Papert, S. (1980) Mindstorms: Children, Computers and Powerful Ideas. NY: Basic Books.

Reusser, K. (1992) Tutoring Systems and Pedagogical Theory: Representational Tools for Understanding, Planning, and Reflection in Problem Solving. In S. Lajoie & S. Derry (eds.) *Computers as Cognitive Tools*. Hillsdale, NJ: Erlbaum .

Rouse, J. (1987) *Knowledge and Power: Toward a Political Philosophy of Science*. Ithaca, NY: Cornell University Press.

Salomon, G. (in press) Differences and Patterns: Studying Computer Enhanced Learning Environments. In S. Vosniadou (ed.) *Psychological and Educational Foundations of Technology-Based Learning Environments*. Berlin: Springer-Verlag.

Salomon, G. (1991) Transcending the Quantitative/Qualitative Debate: the Analytical and Systemic Approaches to Educational Research. *Educational Researcher*. 20, 10-18.

Schank, R. C. (1986) *Explanation Patterns*. Hillsdale, NJ: Erlbaum.

Shneiderman, B. (1983) Direct Manipulation: A Step Beyhond Programming Languages. *IEEE Computer*. 16, 57-69.

Striley, J. (1988) Physics for the Rest of Us. *Educational Researcher* Aug-Sep., 7-19.

Taylor, R. P. (ed.) (1980) *The Computer in the School: Tutor, Tool, Tutte*. NY: Teacher's College Press.

Teodoro, V. D. (1992) Direct Manipulation of Physical Concepts in a Computerized Exploratory Laboratory. In E. de Corte, M. C. Linn, H. Mandl, L. Verschaffel (eds) *Computer-Based Learning Environments and Problem Solving*. Berlin: Springer.

Teodoro, V. D., Batalha, J. C. (1992) *VARIAÇÕES, Investigações sobre Gráficos e Variações: Manual de Utilização*. Monte de Caparica: Universidade Nova de Lisboa 1992. (CHANGES, Investigations about Graphs and Changes: User's Manual).

Veenman, Marcel V. J.; Elshout, Jan. J; Hoeks, J. (in press). Determinants of Learning in Simulation Environments across Domains. In Ton de Jong, Hans Spada, Doug Tawne (eds.), *The Use of Computer Models for Explication, Analysis and Experiential Learning*. Berlin:Springer.

Vigotsky, L. S. (1978) *Mind in Society*. Cambridge, Mass: Harvard University Press.