ED 396 683 IR 017 859

AUTHOR Guzdial, Mark
TITLE Approaches to Classroom-Based Computational Science.
PUB DATE 94
NOTE 10p.; In: Recreating the Revolution. Proceedings of the Annual National Educational Computing Conference (15th, Boston, Massachusetts, June 13-15, 1994); see IR 017 841.
PUB TYPE Reports - Evaluative/Feasibility (142) -- Speeches/Conference Papers (150)

EDRS PRICE MF01/PC01 Plus Postage.
DESCRIPTORS Class Activities; *Computer Assisted Instruction; *Computer Science; *Computer Simulation; Educational Technology; Higher Education; Models; *Science Education; Scientific Concepts; Student Projects
IDENTIFIERS *Computational Models

ABSTRACT

Computational science includes the use of computer-based modeling and simulation to define and test theories about scientific phenomena. The challenge for educators is to develop techniques for implementing computational science in the classroom. This paper reviews some previous work on the use of simulation alone (without modeling), modeling alone (without simulation), and computer-based modeling and simulation. In addition, it describes Emile, an environment which offers software-realized scaffolding to support student modeling activities. Four figures depict a physics explorer simulation of gravity; a student's simulation created in Emile; and Emile text program components. (Contains 23 references.) (AEF)

Paper (T4-202B)

Approaches to Classroom-Based Computational Science

Mark Guzdial
Georgia Institute of Technology
College of Computing
Atlanta, GA 30032-0280
(404) 853-9387
guzdial@cc.gatech.edu

Key words: computational science, modeling, simulation, physics, software-realized scaffolding

Abstract

Computational science includes the use of computer modeling and simulation to define and test theories about phenomena. It is an important new type of science with which students need to be familiar. This paper reviews approaches for providing computational science experiences in classrooms, describes research results, and discusses Emile, an environment which offers software-realized scaffolding to support student modeling activities.

Computer technology offers scientists today the opportunity of having a machine evaluate and present representations of their theories. The activity of creating computer-executable theories and evaluating the theories with powerful, interactive visualizations is called computational science [Denning, 1991]. Computational science is becoming as significant a branch of science as the empirical and theoretical traditions [Pagels, 1988]. The key components of computational science are:

- Modeling: Computers offer a metaphor for specificity in defining theories. Computer-based modeling (the process of defining a theory for simulation) requires that the scientist describe their theory in enough detail that the computer can execute it. Computer languages provide a notation for the specification of models.

- Simulation: Computers can execute a model (called simulating) and provide multiple representations of the process and final results (e.g., graphs, animations, numeric data). The combination of high-speed execution and various forms of visualization forms a synergy for highly interactive, exploratory analysis of theories.

Computational science is important for students of science for two reasons:

1. Computational science is a branch of science growing in importance. Understanding a significant new branch of science and the activities of real scientists is important for student literacy in science. We want students to understand science both to encourage their future participation in science as a career and to provide them with the means of understanding science as it appears in their lives.

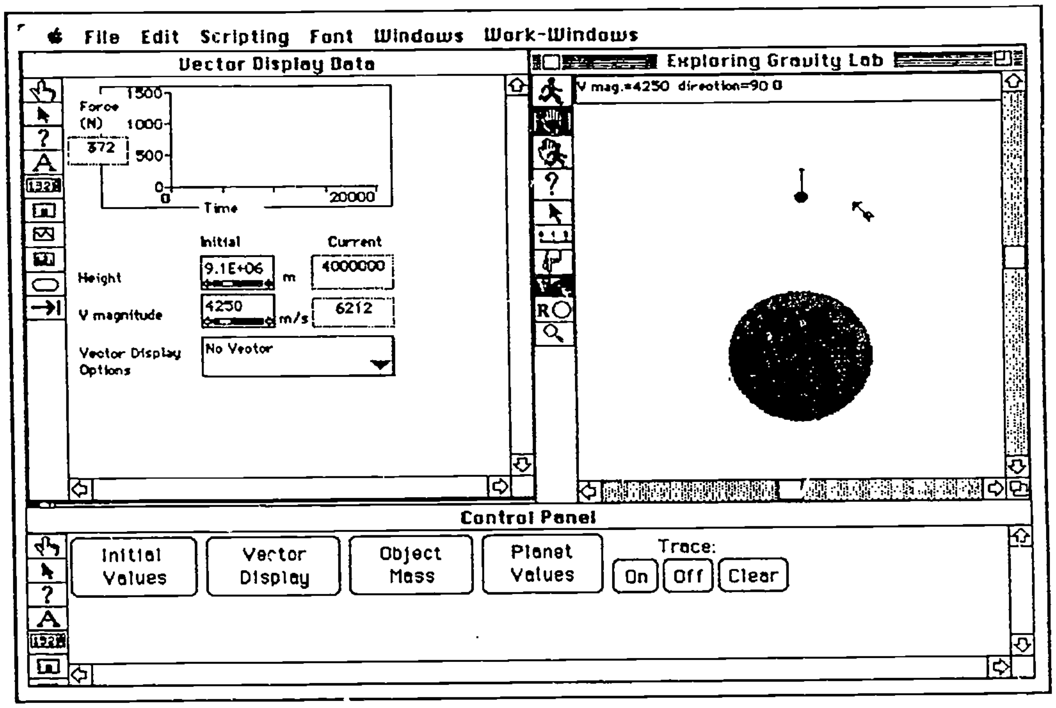
2. What works for scientists may work just as well for students. Scientists use modeling and simulation activities to aid them in defining theorics (modeling) and to enable them to gain insight into their theories (simulation). We might well expect students to gain similar benefits from computational science.

The challenge for educators is to develop techniques for implementing computational science in the classroom. This paper reviews some previous work on the use of simulations alone (without modeling), modeling alone (without simulations), and computer-based modeling and simulations. I also present some new work which uses scaffolding and an innovative classroom structure to support classroom-based computational science.

Simulations without Modeling in the Science Classroom

Richards and his colleagues [Richards, Barowy, & Levin, 1992] have taken the approach of emphasizing simulations instead of modeling in their science classrooms. Their approach has been motivated by a sense, from their experience, that traditional modeling is too complex for most students. Modeling requires certain skills that students may not have. For example, Hestenes has identified a problem that students have in seeing a scientific phenomenon in terms of components that can be modeled [Hestenes, 1987]. Richards et al. focus on having students explore simulations of expert-constructed models in an environment in which students use multiple representations and control various variables of the simulation in order to explore the model interactively (Figure 1). They feel that this approach can lead students to develop an understanding of models and to develop the skills necessary to undertake modeling activities on their own.

The problem that the simulations-alone approach faces is in getting the students to consider that the representations on the screen are more than a video game. Consider, for example, a popular video game such as Super Mario Brothers from Nintendo. When students are exploring this simulation of (at least nominally) physical phenomena, do they consider the underlying scientific model? For example, do they wonder about Mario's velocity and acceleration as he jumps up and falls back down? Do they consider the elasticity of collisions between Mario and the objects in his simulated world? For science learning to occur, there are two goals for a classroom emphasizing simulation:



Force (N) 372

	Initial		Current
Height	9.1E+06	m	4000000
V magnitude	4250	m/s	6212

Vector Display Options: No Vector

V mag.=4250 direction=90.0

Figure 1. A Physics Explorer simulation of gravity. Note the multiple representations (graphical, numeric, and vector) as well as the controls on the bottom of the screen.

1. The students must explore the simulation with a goal of understanding the underlying model. Without that, they can play the simulation as a video game without addressing any of the pertinent scientific concepts.

2. The students must relate the simulation's model to the real world. Without this attempt to transfer the knowledge, students might not address any of their existing preconceptions and change them to more theoretically correct conceptions [Champagne, Gunstone, & Klopfer, 1985].

As diSessa points out [diSessa, 1986], the educational issue is one of the setting for, and interpretation placed on, the simulation. Richards et al. avoided the student interpretation of the simulation as video game with focused in class discussions (1) which refer to the simulation as a representation of real world and (2) which direct attention to the details of the underlying model. Roschelle, for another example, solved the first problem but not the second in his research [Roschelle, 1991]. His students explored mechanics and kinematics in an abstract simulation called the Envisioning Machine. The Envisioning Machine used multiple representations of vectors and particle objects to describe the motion of objects in a Newtonian world, but without reference to Newtonian laws and without explicit labels such as "acceleration" or "velocity". To get students to reflect on the underlying model, he asked his students to force the particles to travel certain paths. Students using his simulation developed sophisticated underlying knowledge of physics (which he calls p-prims) but because this knowledge was not directly connected to the world of physics, the students did not relate the knowledge gained from his simulation to Newtonian physics. While they learned the Envisioning Machine, it was difficult to measure the physics knowledge they had gained.

Modeling in the Science Classroom

A second approach for classroom-based computational science is to ask students to create models, executable theories of scientific phenomena. The advantages of this approach are that it directly addresses the two concerns faced by the simulations-alone approach:

1. Because the students built the model, they describe the behavior of the simulation in terms of the underlying model. The literature on students' Logo programming demonstrates this phenomena, where students try to understand the behavior of their programs in terms of the procedures they entered [Harel, 1991; Papert, 1980].

2. Because students build the model to describe the real world, the hypothesis is that they will relate the simulation results to the real world and thus transfer what they learn. Literature on Logo programming suggests that students do not often use skills learned in a simulated world in the real world [Pea & Kurland, 1986], but some literature on scientific modeling [Halloun & Hestenes, 1987; Sherin, diSessa, & Hammer, 1992] suggests that students do learn science principles from modeling activities.

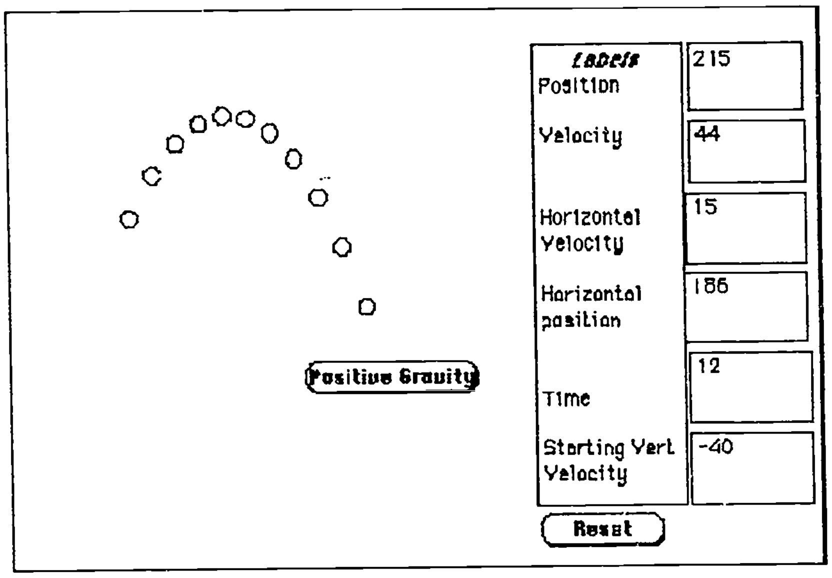
Hestenes and his colleagues and Sherin and the Boxer group at the University of California at Berkeley [Halloun & Hestenes, 1987; Sherin, et al., 1992] report that their students learn physics by developing models in class, as a joint activity, with the instructor and other students acting as the computer to evaluate and simulate the model. In these classrooms, students engage in modeling but without computer-based simulation. Hestenes had students use natural language and mathematics to define their models. Sherin et al. used the programming language Boxer as a notation for the model. Boxer provided a notation and a standard on which to base their model. However, the Boxer students did not actually implement their model, so they lost the potential advantages of the multiple forms of interactive representations available on the computer, which scientists use to gain insight into a theory.

Computer-based modeling (which would then lead naturally into computer-based simulations) has been difficult to implement in classrooms. For example, Tinker and his colleagues [Tinker, 1990] conducted a four week workshop for high school students to teach them modeling skills in two computer-based modeling environments (Stella [Mandinach, 1989] and Microsoft Excel). Tinker reports that traditional classrooms are not structured to implement computational science activities: It's difficult for students (Tinker's study used three weeks of preparatory instruction before starting modeling), it's difficult for teachers who have never had such experiences themselves, and traditional curricula are not conducive to modeling activities.

Emile: An Approach to Computer-Based Modeling

The challenge, then, is (1) to provide the advantages of modeling over a simulation-alone approach, (2) while avoiding the problems identified by Tinker, and (3) while gaining the learning-through-modeling reported by Hestenes and Sherin and their colleagues. My approach to this challenge is Emile, a programming environment running on the Apple Macintosh in which students construct kinematics simulations and multimedia demonstrations [Guzdial, 1993]. I evaluated Emile in a three week (Monday through Friday, three hours a day) workshop in which five high school students created three simulations and a multimedia demonstration of velocity, acceleration, and projectile motion. None of the high school students had any previous high school physics, none had any previous Macintosh experience, and three had never programmed before.

Figure 2 is a screenshot of a sample project developed by a student (without previous programming experience) using Emile. This project is a simulation of two-dimensional projectile motion. A user clicks down the mouse button with the arrow on the Positive Gravity object, drags the object somewhere on the screen, and releases the mouse button. The object then launches with the specified horizontal and initial vertical velocity, eventually falling back down in an accelerated motion as if under the influence of gravity. The student built this program in three days.



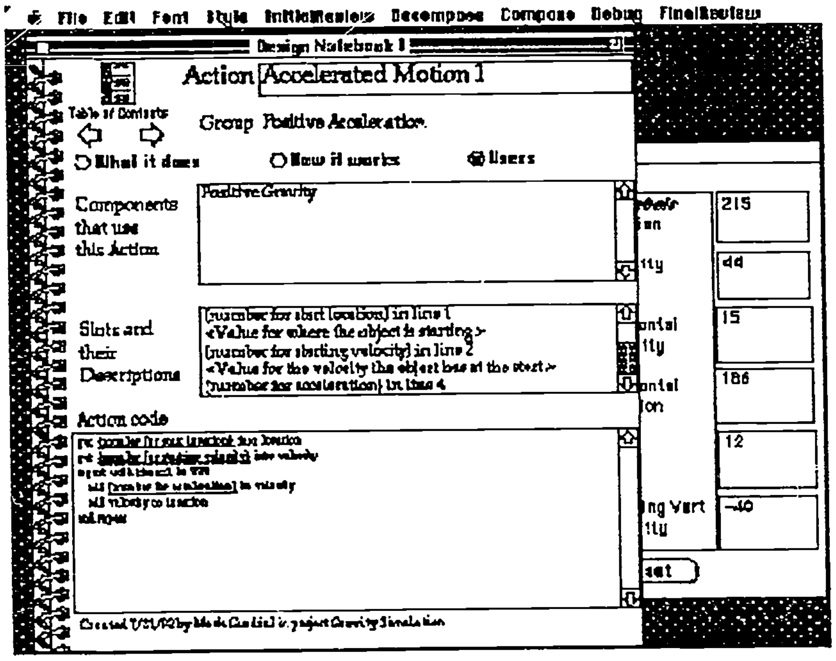
Labels	
Position	215
Velocity	44
Horizontal Velocity	15
Horizontal position	186
Time	12
Starting Vert Velocity	-40

Figure 2. A student's 2-D projectile motion simulation, created in Emile.

There were two key factors which were used in the Emile approach. First, Emile offered extensive scaffolding for modeling activities. Scaffolding is support (most often provided by a teacher) which enables students to succeed at an activity which they might not succeed at without the support [Collins, Brown, & Newman, 1989; Farnham-Diggory, 1990; Palincsar, 1986; Rogoff, 1990] 1. Emile's scaffolding was realized in software, and it included:

- Mixed-media programming framework: Students programming in Emile did not type traditional textual programs. Instead, they assembled programs out of chunks of text program called actions which performed a specific task and could be manipulated as a unit, and graphical objects such as the Positive Gravity button. Students programmed in Emile by assembling actions into buttons, rearranging the actions in the order in which they should execute for the model, and filling blanks in the text programs called slots. These slots had domain-specific roles for which they were named, such as "Number for starting velocity" and "Number for acceleration." (See Figure 4.) Thus, students using Emile did not have to learn text program syntax, they were able to use graphical as well as textual programming elements, and these elements were tied tightly to the kinematics domain.

BEST COPY AVAILABLE


File Edit Font Style InitialReview Decompose Compose Debug FinalReview
Design Notebook 1
Table of Contents
What it does
How it works
Users
Components that use this Action
Slots and their Descriptions
Action code
set

Action Accelerated Motion 1

Group Positive Acceleration

Positive Gravity

[number for start location] in line 1
<Value for where the object is starting>
[number for starting velocity] in line 2
<Value for the velocity the object has at the start>
[number for acceleration] in line 4

[illegible]

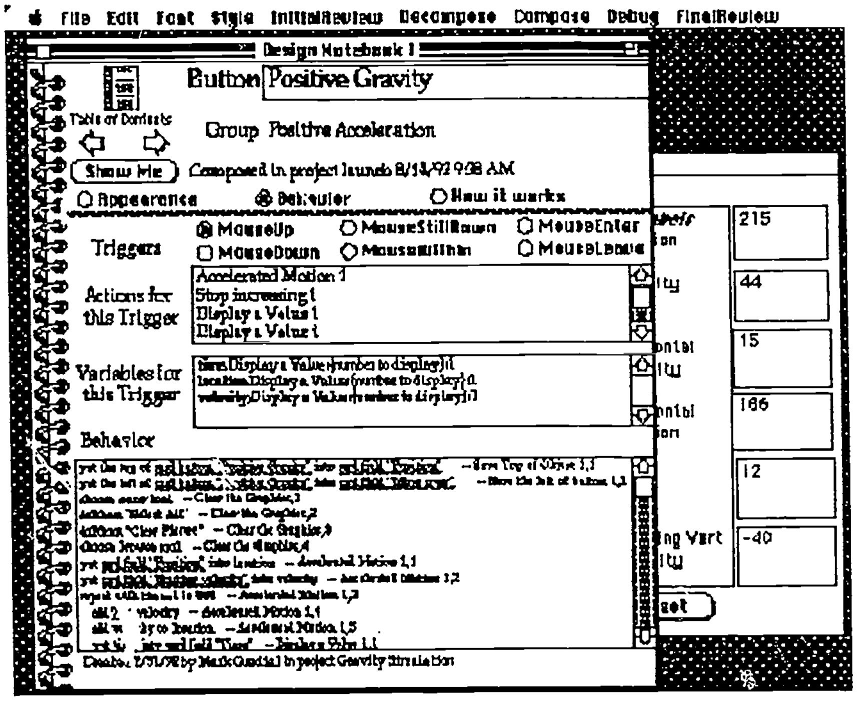
Created 7/21/92 by Mark Guzdial in project Gravity Simulation

[illegible]	215
ity	44
ontal ity	15
ontal ion	186
	12
ing Vert ity	-40

Figure 3. The page for the action Accelerated Motion., provided for the students in the basic Emile library. This is a text program component that a student will use, reorder, and modify in a button's behavior. Notice that the slots (underlined) are named in terms of the kinematics domain, such as "Number for acceleration." The partially obscured window is the Project Window where the student creates the simulation (seen in Figure 2).

- Prompts for articulation and reflection: Emile prompted students to articulate and reflect often on the program being created, on the program as a model, and on their process. These prompts included project descriptions, plans, predictions, and journal entries.
- Design Notebook: All of a student's work on a project, from articulation entries to program components, was stored in a single Design Notebook. The Notebook is a collection of components and tools arranged one to a page with ·veral navigation facilities provided for movement between pages.

BEST COPY AVAILABLE



Button Positive Gravity

Group Positive Acceleration

Composed in project launch 8/14/92 9:38 AM

Behavior [✓]

Triggers: MouseUp [✓]

Actions for this Trigger:
Accelerated Motion 1
Stop increasing 1
Display a Value 1
Display a Value 1

Variables for this Trigger:
time:Display a Value(number to display)1
location:Display a Value(number to display)1
velocity:Display a Value(number to display)1

Behavior:

```
put the top of [illegible] into [illegible]        --Save Top of Object 1,1
put the left of [illegible] into [illegible]       --Save the left of button 1,1
choose eraser tool  --Clear the Graphics,1
doMenu "Select All"  --Clear the Graphics,2
doMenu "Clear Picture"  --Clear the Graphics,3
choose browse tool  --Clear the Graphics,4
put [illegible] into location  --Accelerated Motion 1,1
put [illegible] into velocity  --Accelerated Motion 1,2
repeat with [illegible]  --Accelerated Motion 1,3
  add [illegible] velocity  --Accelerated Motion 1,4
  add [illegible] to location  --Accelerated Motion 1,5
  put [illegible] into card field "Time"  --Display a Value 1,1
```

Created 7/31/92 by Mark Guzdial in project Gravity Simulation

Figure 4. The page for the button Positive Gravity, provided for the students in the basic Emile library. Positive Gravity was used as a template for most of the students' simulations. For example, the student who created Figure 2 used Positive Gravity as a starting place for his 2-D projectile simulation. Actions are identified in the behavior by a suffix on each line (compare the lines annotated with "Accelerated Motion" with Figure 3).

- Library of Components: Emile provided over one hundred basic programming components for students to use in creating their programs. These components ranged from actions that embodied basic kinematics concepts, such as "Accelerated Motion" (Figure 3), and a button that simulated one-dimensional projectile motion (Figure 4), to buttons that displayed digitized video for multimedia demonstrations. Students made extensive use of the library in the creation of their programs.

An important aspect of Emile's scaffolding is that most of it could be faded (that is, reduced or removed) under student control. For example, if a student wanted to type text programs directly, instead of using the mixed media programming framework supplied by Emile, she could easily turn off that scaffolding and directly type text programs into buttons. Three students were typing text programs by the final project in the workshop, while two other students continued to use actions and slots throughout. Thus, in the final program, all five students were using Emile to create projects of similar complexity, but they were using tools and programming at a level chosen by them.

The second key factor used with Emile was the design of the workshop environment and curriculum. The workshop was set up to encourage motivation and collaborative discussion.

- Authentic task: Students were told that they were creating their programs for other physics students. While students never met their audience, they made their design decisions with an audience in mind, e.g., choosing some features over others because of the modeler's estimation of other students' interest. Thus, the students' task had authenticity: it had a real audience. An authentic task helps in promoting student motivation2 which is an important factor in student success at such a complex task as modeling [Blumenfeld, 1992; Blumenfeld, Soloway, Marx, Krajcik, Guzdial, & Palincsar, 1991].

- Collaborative atmosphere. While students worked on their own projects, the students collaborated in the sense of "collaboration in the air" [Kafai & Harel, 1991]. Students would often wander the room, reviewing each others work both for techniques that they might use and to critique each others' project. Students did not hesitate to note "But that's not how the real world works." The ensuing discussion was significant for both the devt oper and the reviewer in encouraging reflection on the model and on its role as a theory of physical ,nen ımena.

Students were evaluated on their science lt arning with a clinical interview (using a similar technique as [Finley, 1986]) where students were asked to verbally solve problems in physics both at the beginning and end of the workshop. Further, the students were probed for a detailed description of their physics understanding. I found that all five students improved in their conceptualizations of velocity, acceleration, and projectile motion.

For example, one problem involved estimating the velocity and time to impact of a rock dropped from a three story building. On a pre-test, student B simply gave up on the problem, saying "And the velocity is, let's see, gravity...ummm...I can't remember that." On the post-test, student B was able to elaborate a far more sophisticated understanding. In fact, as seen in the below quote, he seems to mentally simulate the second-by-second falling of the rock:

B: It would be about one second

R: Okay, where did you get that from?

B: If the acceleration is 30 feet per second per second, then per second it will be going 30 feet per second, then...it will just take a little longer for it to get to the ground.

R: Why?

B: Because you have to divide to get the average velocity, which is how fast it's going, and how you can measure how far it's gone, you have to...let's see...it will be going, it will be going 15 meters per second. Maybe two seconds, I guess.

R: Why?

B: Because...1.5 seconds. Because, by the time it's accelerated the second second, it will be going about 45 feet per second, so it'll have to be between the first and second second that it hits the ground.

An approach such as the one used in Emile may not be any easier for teacher's than the ones described by Tinker. So the issue of supporting teachers who are themselves unfamiliar with computational science is still an open issue. However, the Emile approach does hold promise for reducing the complexity of modeling for students.

Summary

The two key components of computational science are modeling and simulation. This paper has reviewed the literature of approaches to classroom-based computational science and presented a new study which offers a new approach to supporting student modeling.

- Simulations can be used apart from modeling, by using models created by an expert. The advantage to this approach is that it avoids the complex activity of modeling for students. The danger of this approach is that students may not reflect on the underlying model and its applicability as a theory of the real world. A successful approach described here is to use classroom discussion to raise critical questions and to relate the simulation to science.
- Using modeling without computer-based simulation encourages reflection on the model and its role as a scientific theory, but loses the potential advantages of individual, interactive exploration of the model through a simulation, which scientists use to gain insight into theories.
- The approach used with Emile is to allow students to create their own models, but in a programming environment that offers extensive scaffolding. Further, Emile was used in a workshop that featured authentic tasks and encouraged collaboration. While there are still open questions for use of this approach (such as how to prepare teachers for classroom-based computational science), it holds promise for diminishing the complexity of modeling for students.

References

Blumenfeld, P. C. (1992). The task and the teacher: Enhancing student thoughtfulness in science. In J. Brophy (Ed.), Advances in Research on Teaching JAI Press.

Blumenfeld, P. C., Soloway, E., Marx, R. W., Krajcik, J. S., Guzdial, M., & Palincsar, A. (1991). Motivating project-based learning: Sustaining the doing, supporting the learning. Educational Psychologist, 26(3 & 4), 369-398.

Champagne, A. B., Gunstone, R. F., & Klopfer, L. E. (1985). Effecting changes in cognitive structures among physics students. In L. H. T. West & A. L. Pines (Eds.), Cognitive Structure and Conceptual Change (pp. 163-188). Orlando, FL: Academic Press.

Collins, A., Brown, J. S., & Newman, S. E. (1989). Cognitive apprenticeship: Teaching the craft of reading, writing, and mathematics. In L. B. Resnick (Ed.), Knowing, Learning, and Instruction: Essays in Honor of Robert Glaser Hillsdale, NJ: Lawrence Erlbaum and Associates.

Denning, P. (1991). Computing, applications, and computational science. Communications of the ACM, 34(10), 129-131.

diSessa, A. A. (1986). Artificial worlds and real experience. Instructional Science, 14(3-4), 207-227.

Farnham-Diggory, S. (1990). Schooling. Cambridge, MA: Harvard University Press.

Finley, F. N. (1986). Evaluating instruction: The complementary use of clinical interviews. Journal of Research in Science Teaching, 23(7), 635-650.

Guzdial, M. J. (1993). Emile: Software-realized scaffolding for science learners programming in mixed media. Unpublished Ph.D. dissertation, University of Michigan.

Halloun, I. A., & Hestenes, D. (1987). Modeling instruction in mechanics. American Journal of Physics, 55(5), 455-462.

Harel, I. (1991). Children Designers: Interdisciplinary Constructions for Learning and Knowing Mathematics in a Computer-Rich School. Norwood, NJ: Ablex.

Hestenes, D. (1987). Toward a modeling theory of physics instruction. American Journal of Physics, 55(5), 440-454.

Kafai, Y., & Harel, I. (1991). Children learning through consulting: When mathematical ideas, knowledge of programming and design, and playful discourse are intertwined. In I. Harel & S. Papert (Eds.), Constructionism (pp. 110-140). Norwood, NJ: Ablex.

Mandinach, E. (1989). Model-building and the use of computer simulations of dynamic systems. Journal of Educational Computing Research, 5(2), 221-243.

Pagels, H. R. (1988). The Dreams of Reason: The Computer and the Rise of the Sciences of Complexity. New York: Simon and Schuster.

Palincsar, A. S. (1986). The role of dialogue in providing scaffolded instruction. Educational Psychologist, 21(1-2), 73-98.

Papert, S. (1980). Mindstorms: Children, computers, and powerful ideas. New York, NY: Basic Books.

Pea, R. D., & Kurland, D. M. (1986). On the cognitive effects of learning computer programming. In R. D. Pea & K. Sheingold (Eds.), Mirrors of Minds Norwood, NJ: Ablex Publishing.

Richards, J., Barowy, W., & Levin, D. (1992). Computer simulations in the science classroom. Journal of Science Education and Technology, 1(1), 67-79.

Rogoff, B. (1990). Apprenticeship in thinking: Cognitive development in social context. New York: Oxford University Press.

Roschelle, J. M. (1991). Students' Construction of Qualitative Physics Knowledge: Learning about Velocity and Acceleration in a Computer Microworld. Unpublished Ph.D. Dissertation, University of California at Berkeley.

Sherin, B., diSessa, A. A., & Hammer, D. (1992). Dynaturtle revisited: Learning physics through collaborative design of a computer model. Presented at the annual meeting of the American Educational Research Association, San Francisco, CA.

Tinker, R. F. (1990). Teaching theory building. Final report on National Science Foundation grant. Also, Technical Report of Technical Education Research Centers, Cambridge, MA.

1A more elaborated discussion of scaffolding and software-realized scaffolding can be found in [Guzdial, 1993].

2The study described here is not a test of the value of the authentic task, especially since the students were already so motivated (e.g., volunteering to attend a summer workshop).