The Effect of Computer Science and Active Simulation Design on Physics Learning

Rivka Taub
Weizmann Institute of Science
rivka.taub@weizmann.ac.il

Michal Armoni
Weizmann Institute of Science
michal.armoni@weizmann.ac.il

Mordechai (Moti) Ben-Ari
Weizmann Institute of Science
moti.ben-ari@weizmann.ac.il

Abstract

Computational science is a growing scientific field that involves the design of computational models of scientific phenomena. This field combines science, computer-science (CS), and applied mathematics in order to solve complex scientific problems. In the past few years computational science has been taught in secondary schools, a fact that led researchers to wonder about the effect of combining disciplines on students' learning. Specifically, we investigated the effect of CS while actively designing a simulation, on students achieving a higher level of conceptual learning in physics. We used the knowledge integration (KI) framework (Linn & Eylon, 2006, 2011) to analyze the students' learning. This framework describes four processes that should underlie meaningful learning. Our findings indicate that CS frequently caused the emergence of the KI processes. For example, in constructing a computer simulation of physical phenomena, students represented their physics knowledge in a concrete form that provides criterion which they can use to assess their knowledge.

Keywords: technology, computational science, computer science, knowledge integration.

Introduction

Computational science deals with the construction of computational models. It is an interdisciplinary field that combines science, computer science (CS) and applied mathematics, in order to understand and develop models of scientific systems (Yasar & Landau, 2003). Computational science has reached predictive capabilities similar to those of traditional experimentation and theory (Sloot, n.d.). The success of this approach has led to the proposal that computational science be taught in secondary schools to expose students to this scientific method, and will acquire knowledge and strategies to solve complex problems. A number of computational environments are used for teaching computational science (e.g., EJS, VPython); each has its own unique features and effects on students' learning.

This paper describes research into the learning of tenth-and eleventh-grade students who studied a course in computational science using EJS and Maxima. During the initial stages of our research (to be reported elsewhere), we discovered that the students achieved meaningful learning. Our next goal was to investigate what led to this outcome and what, if anything was the unique contribution of CS to it. To analyze the learning processes, we chose to use the knowledge integration (KI) framework (Linn & Eylon, 2006, 2011) that describes the mechanism underlying the achievement of meaningful learning. This document summarizes analysis of many hours of students' work on their final projects (a detailed analysis of one case study appears at Taub, Armoni, and Ben-Ari (2013)).

Theoretical Background

Computational science

Computational science is a field that deals with the construction of computational models. The Journal of Computational Science (Sloot, n.d.) describes it as an interdisciplinary field that uses advanced computing and data analysis. The ACM/IEEE (2008) recommended that the undergraduate curriculum in computer science include computational science as an elective. The K-12 Computer Science Standards (CSTA, 2011) propose that students in grades 9-12 who learn computer science also study a course on computational science.

Software tools

EJS is a software package that enables the construction of simulations. Its user-friendly Java environment Java facilitates graphical design (Christian & Esquembre, 2007; Esquembre, 2004). EJS breaks the modeling process into three activities: documentation, modeling and interface design. In the modeling activity, the designer implements the physics in Java code.

Maxima is a Computer Algebraic System for the manipulation of symbolic and numerical expressions; it can perform computations such as differentiation and solving ordinary differential equations.

Knowledge integration

Knowledge integration brings together recent trends in developmental, constructivist, socio-cultural and cognitive perspectives on learning. According to KI, learners build knowledge by going through four processes:

- Eliciting ideas. Learners become aware of their pre-existing knowledge.
- Adding new ideas. Learners are introduced to ideas that are new to them and that may come from various sources such as a teacher, a textbook, or a peer.
- Using criteria to evaluate the new ideas. Questions and tests are used to evaluate whether the learners consider the newly acquired ideas to be acceptable or not.
- Sorting out and reflecting. The learners reflect and differentiate between their pre-existing ideas and the newly acquired ones.

The four processes do not necessarily appear one after another or in the described above order.

The KI framework has been used, tested and revised over the past two decades. It has been used: as a guide for the design of instructional activities (e.g., Linn et al., 2003); as an assessment tool to evaluate learners' scientific knowledge (e.g., Liu et al., 2008); and, as a tool to analyze learning processes (Yerushalmi et. al., 2012).

Most of the research done within the context of KI dealt with knowledge within a single discipline, such as the physical or life sciences, occasionally in the presence of technological tools (e.g., Kali, Orion, & Eylon, 2003). Some research has been done in interdisciplinary contexts (Chiu & Linn, 2011), but we could not find an example of KI-related research dealing with the effects of learning one discipline on the learning of another.

Methodology

Research question

What is the specific contribution of CS to the development of the students' conceptual understanding in physics?

The setting

The research took place in a course on Computational Science intended for talented high-school students. The three-year course (tenth-twelfth grades) is interdisciplinary: students learn to program computational models of physics phenomena, thereby learning physics, mathematics and computer science. Most of the subject matter was learned independently by the students, while the teacher served as a mentor. The instruction is directed by a textbook. In addition, the students were required to develop interim and final projects on their own. This research was carried out in the tenth-and eleventh-grade classes.

Participants

Seven pairs of tenth-and eleventh-grade students who worked on their final projects participated in our research. The decision to focus on these pairs was taken after observing the whole class and was based on our wish to observe a diverse set of students in terms of gender, level of independence in class work, and more.

Research tools

We observed and recorded the students' work on their final projects. We recorded the students' discourse and computer screens using the Debut screen-capture software. Altogether, more than 70 hours were recorded.

Analysis

A qualitative research method was used to analyze the students' recorded discourse:

(a) Categorization of the statements (sometimes parts of them) in the students' discourse that expressed a specific KI process.
(b) Categorization of each KI statement as related to CS, physics or mathematics.
(c) Understanding what in CS led to the appearance of the KI processes.

Findings

In all the students' recordings, KI processes were observed; some were related to CS, some to physics and some to mathematics. We found that the CS point of view was one of the major triggers to physics conceptual learning. We were therefore able to articulate the effects of learning science in the context of CS on the emergence of KI processes, and consequently on achieving conceptual understanding. Here we bring citations that demonstrate these effects, but due to space limitations, only one citation is given for each effect.

Elicitation of students' existing knowledge

The task of programming requires the students to explicitly list the physics entities (as programming ones) and the relations among them. The transition between the physics and the programming representations enables the students to elicit their current physics knowledge, an essential process for further learning. In particular, the design of the interface in EJS requires the students to define graphical objects, together with their related variables and functions.

For example, two students simulated collision of a pair of balls in order to illustrate the law of conservation of momentum. In order to create the graphics for the balls, the students had to define the variables related to each ball: velocity, x, y. This elicited their knowledge on linear motion and was a trigger for further physics learning:

S1: What are the variables of each ball? v [and] the coordinate of the location. Oh, there's this formula we studied:$x = vt$. It's all related.

Adding new ideas
Environments like EJS enable the students to construct visual simulations that display real-world objects. When students observe the simulation running, they sometimes use it as a tool for learning the physical system being simulated. Since the students developed the simulation themselves, the new added ideas obtained from its execution reflect their physics knowledge, and therefore may be incorrect.

For instance, two students simulated the Iron-Dome system. They did not know how different shooting angles would affect the flight of the Iron-Dome rocket. To solve this problem student S2 suggested:

S2: Let's run the simulation with different angles and see how the paths of the rocket are changing.

Criteria for evaluating the students' physics knowledge
The simulation encodes the students' physics knowledge in a computer program. As a consequence, running the simulation provides visual feedback of this knowledge; if errors occur, this forces the students to check for gaps in their knowledge. Therefore, running the simulation provides a *criterion* for the students' physics knowledge. The students examine their knowledge in several ways:

- Comparing the static appearance of the physical system in the simulation with the students' image of how the physics phenomenon should appear. This image may be based upon the students' experience in the real world or upon their scientific knowledge. The comparison leads to checking the physics knowledge underlying the simulation.

  For example, two students simulated lens and mirrors that create images. They wrote the code for the image of a plane mirror. They ran the simulation and discovered that the image was upside down; therefore, they concluded that the formulas written in the code were wrong:

  S3: Look, it's upside down. There must be a problem in our equations.

- Comparing between the dynamic behavior of the simulation and the students' predictions. Based on their physics' knowledge of formulas and principles, the students predict how the simulation should behave. They run the simulation and compare their predictions to its behavior.

  For example, two students simulated fire that heats air molecules. They ran the simulation and found that – although the molecules' temperature was supposed to be higher in the sections closer to the fire and lower further away – it was identical in all sections. The students re-checked the formula they programmed and found the errors.

- Checking the behavior of the simulated system in different and extreme cases.

  For example, students who wanted to check their simulation planned – before running the simulation – which values they should enter in order to check that it runs correctly in all cases:

  S4: Let's try different cases to make sure it works.

- Searching for the most efficient program enables the learners to re-examine their physics understanding.

  For example, students who simulated the circular motion of a car did not know how to express its location on the road. A friend suggested that they think of the circle as a set of very small lines. One of the students thought that this suggestion was correct but not efficient:

  S5: It's a huge loop. The computer can't handle it.

  They therefore dropped this idea and searched for a more efficient procedure. They re-checked the physics formulas to achieve a deeper understanding, and eventually reached a correct and simple way to express the car's location.

Reflecting and sorting out
The debugging process enabled the learners to reflect and sort out their knowledge. When the students faced an error in a simulation's execution, they re-examined their knowledge in physics and in CS. This helped the students detect incorrect ideas and sort out the correct from the incorrect ones.

Discussion and Conclusions

This document reports on the investigation of students' learning in a course on computational science. Analysis of seven case studies showed that while working on their final projects the students gained new physics knowledge. The learning was well-described by the KI processes that were triggered by the different perspectives of physics, mathematics and CS. This paper focused on the effect of CS on them and showed that fundamental elements of programming promoted all the KI processes, from elicitation to sorting out.

This finding is important since physics instructors many times prefer using ready-made simulations to prevent the hard work related to teaching programming. This paper clearly shows that active programming of simulations stimulates the appearance of the KI processes, which in turn promotes gain of physics' knowledge. We conjecture that the importance of programming the simulation stems from the fact that it represents the physics knowledge of the learners and not of experts. It represents their knowledge of physics' objects and procedures as programming objects, functions and code and enables the learners to test and improve their physics knowledge.

Computational science is an important interdisciplinary scientific field which should appear more often in science education. It includes not only three different disciplines but it also combines an intensive use of technology. Further research is needed to investigate whether the above described contribution stemmed mainly from the activities related to computer science or from the technological ones, and whether these contributions can be distinguished.

References

ACM/IEEE. (2008). Computer science curriculum 2008: An interim revision of CS 2001.

Chiu, J. L., & Linn, M. C. (2011). Knowledge Integration and WISE Engineering. *Journal of Pre-College Engineering Education Research (J-PEER), 1*(1), 2.

Christian, W., & Esquembre, F. (2007). Modeling physics with easy Java simulations. *The Physics Teacher, 45*, 475.

CSTA. (2011). CSTA K–12 Computer Science Standards

Esquembre, F. (2004). Easy Java Simulations: A software tool to create scientific simulations in Java. *Computer Physics Communications, 156*(2), 199-204.

Kali, Y., Orion, N., & Eylon, B.-S. (2003). Effect of knowledge integration activities on students' perception of the earth's crust as a cyclic system. *Journal of Research in Science Teaching, 40*(6), 545-565. doi: 10.1002/tea.10096

Linn, M. C., Clark, D., & Slotta, J. D. (2003). WISE design for knowledge integration. *Science Education, 87*(4), 517-538. doi: 10.1002/sce.10086

Linn, M. C., & Eylon, B.-S. (2006). Science education: integrating views of learning and instruction. In A. E., W. P.H., A. P.A. & C. L. (Eds.), *Handbbok in educational psychology* (pp. 511-544): Lawrence Erlbaum Associates.

Linn, M. C., & Eylon, B.-S. (2011). *Science learning and instruction: taking advantage of technology to promote knowledge integration*. New York: Routledge.

Liu, O. L., Lee, H.-S., Hofstetter, C., & Linn, M. C. (2008). Assessing Knowledge Integration in Science: Construct, Measures, and Evidence. *Educational Assessment, 13*(1), 33-55. doi: 10.1080/10627190801968224

Sloot, P. (n.d.). Aims and Scope of the Journal of Computational Science. Retrieved from http://www.elsevier.com/wps/find/journaldescription.cws_home/721195/description

Taub, R., Armoni, M., & Ben-Ari, M. (2013). The Contribution of Computer Science to Learning Computational Physics. In I. Diethelm & R. Mittermeir (Eds.), *Informatics in Schools. Sustainable Informatics Education for Pupils of all Ages* (Vol. 7780, pp. 127-137): Springer Berlin Heidelberg.

Yasar, O., & Landau, R. (2003). Elements of Computational Science & Engineering Education. *SIAM Review, 45*(4), 787-805.

Yerushalmi, E., Puterkovsky, M., & Bagno, E. (2012). Knowledge Integration While Interacting with an Online Troubleshooting Activity. *Journal of Science Education and Technology*, 1-12. doi: 10.1007/s10956-012-9406-8