

Constructivism, Group Work, and Epistemic Games to Enrich an Introduction to Programming Class

Graciela González
Sam Houston State University
Department of Computer Science
Huntsville, TX, 77341 U.S.A.
csc_ghg@shsu.edu

Any instructor that has tried to explain pass by reference parameters to the novice programmers can attest to the difficulty they have with some of the basic concepts of programming. This is especially true for those that do not have it as their major area of study and those that encounter it for the first time. Other subjects, starting with the model of a computer and culminating with more abstract notions such as that of object oriented design are no easier for the students to learn and for professors to teach.

The approach described here attempts to present material in an Introduction to Programming class in ways that may appeal to a diversity of styles of learning and to let students take a more active role in the class, constructing their knowledge of programming on their own terms through diverse activities and “hands-on” metaphors. For example, students can learn about memory locations by building a physical cardboard model with labeled locations, and “store” and “retrieve” values from it to complete an assigned task. To learn about algorithms, they might be requested to write the necessary steps and then program a toy truck to carry a message from one office to another. A preliminary application of some of these concepts showed promising results, though no formal evaluation was made.

The proposed approach incorporates several techniques around the educational theory of constructivism. Basically, constructivism claims that knowledge is actively constructed by the student, not passively absorbed from textbooks and lectures. Effective learning demands not just discovery of facts, but construction of viable mental models, and that teachers must actively guide the student in this effort. Group work and other activities must be designed to allow the students to construct these models. Constructivism does not reject classical means of instruction, but challenges the assumption that students know (learn) what the lecturer tells them. Good references for constructivism can be found in [1].

The approach developed here is a combination that allows students to apply his or her own thoughts about the lecture with practical applications in the classroom, weaving the new knowledge into what they already know through diverse guided activities. Aside from problem solving and writing short programs as a group, which are part of the more advanced lectures, activities might include constructing models, running animations or robots, or epistemic games[2]. As described in [2], an epistemic game is the process of completing an epistemic form or knowledge structure that guides the inquiry process. It shows how knowledge is organized or concepts are classified, as well as illustrating the relationships among the

different facts and concepts being learned. The completion or creation of the structure is the object of the epistemic game.

Though lectures are not eliminated, they are necessarily shorter, and almost always followed by an enriching activity. Many activities are done without a computer, others, such as problem solving and programming in groups, or running animations and simulations would require one. Also, since many of the activities are carried out in groups, the ideal classroom setting would include work tables for 4 to 5 students, where material is readily available and where they can use a computer as a group. “Material” includes anything and everything needed to realize each activity. For example, imagine having wooden blocks of different shapes, colors and sizes, and a shelf with small divisions fitted with interchangeable “doors” that have openings that match the shapes. The shelves represent memory locations. The shapes are data types: square for a float, a square with rounded corners for an integer, a large coin for a boolean. “Declaring a variable” becomes then the labeling of one of the shelves and fitting over it a cover with the right shape (data typing) so that only blocks of that shape might fit in it. A particular block of a certain color/size becomes the value for that variable, and it can be “stored” in the shelf reserved for it.

While some tables are equipped with the blocks, others might have software running that helps student put all of these concepts together without having to write a full-fledged program. Consider also an activity where students become a “live” model of a computer: a team becomes “Input Device” while others become CPU, memory, storage, and an output device. All teams perform their role and complete a task (program) following the instructions on what they as “system components” can and cannot do.

In conclusion, a new approach to teaching Introduction to Programming courses is necessary in order to produce new, and hopefully better, results. At the very least, activities that follow a lecture have the added advantage of allowing immediate feedback to the instructor and to the students, in such a way that errors and error recovery become less traumatic and more pedagogically productive.

References.

- [1] Ben-Ari, Mordechai, “Constructivism in Computer Science Education”, *Journal of Computers in Mathematics and Science Teaching*, ACM 2001.
- [2] Sherry, L., & Trigg, M. Epistemic forms and epistemic games. *Educational Technology*, 36(3), 38-44, 1996.