3D Game Design with Programming Blocks in StarLogo TNG

Kevin Wang, Harvard Graduate School of Education, Technology in Education Program, Longfellow Hall Rm. 321, Appian Way, Cambridge, MA 02138, kevinkw@gmail.com
Corey McCaffrey, Daniel Wendel, Eric Klopfer, Massachusetts Institute of Technology Teacher Education Program, 77 Massachusetts Ave, Rm. 10-337, Cambridge, MA 02139
Email: coreymcc@mit.edu, djwendel@mit.edu, klopfer@mit.edu

Abstract: Research on the effectiveness of using a block programming language with a three-dimensional environment and game design as the basis for curriculum to teach children concepts in Computer Science. We observed a lower barrier to entry and faster learning in seventh to ninth graders exposed to the new curriculum using StarLogo TNG.

Background

In a recent speech to an audience of computer science faulty, Bill Gates lamented that despite all the leaps in technology, he wondered why there more people are not interested in CS as a career. “Are they making breakthroughs like speech recognition or AI? I’m dying to see these new games they’re inventing” (Seattle Post, July 19, 2005). The Computing Research Association reported in May 2005 that interest in CS as a major dropped 60% between 2000 and 2004. Even more distressing is that only 0.5% of incoming female freshmen expressed an interest in a CS undergraduate degree, an all time low since the early 1970s when programming involved paper hole punches (Vegso 2005). StarLogo TNG is a 3D graphics-based programming environment that uses drag-and-drop programming blocks instead of the more traditional text-based approach. We hypothesize that lowering the entrance barrier, making the learning curve shallower, and producing more exciting end results will yield increased interest in CS in middle school and high school age students while improving their logical thinking and problem solving skills.

StarLogo TNG

While StarLogo TNG is designed upon the basic framework of Logo, there are two distinct improvements that separate it from its predecessor. The programming is now done with programming blocks instead of text commands (see Figure 1). The obvious advantage is that now the program has moved from the abstract to the visual. As many students were already familiar with using a computer by dragging and dropping objects, StarLogo TNG came very naturally to them. The programming blocks are arranged by color based on their function, and it enables students to associate similar programming blocks with each other. Since the programming blocks are puzzle piece shaped, only syntactically sensible blocks can fit into each other, which eliminates a whole host of bugs for the students. Seymour Papert found in *Mindstorms* that modular form representation of a process makes it easier to debug (Papert 1993), we took it one step further and have a visual representation of that modular form. In the past we have found that in StarLogo 2, text-based syntax errors seemed to be a significant barrier to Logo implementation by teachers. For example, the equals block has two cutouts that reminds the students that they need to compare a variable to a number, not just saying “= 17” and forgetting that the computer needs to be explicitly told what to compare 17 to. In addition, studies have shown that block programming lessens the intimidation factor of programming to girls and has a greater appeal than that of text-based languages (Begel 2004).

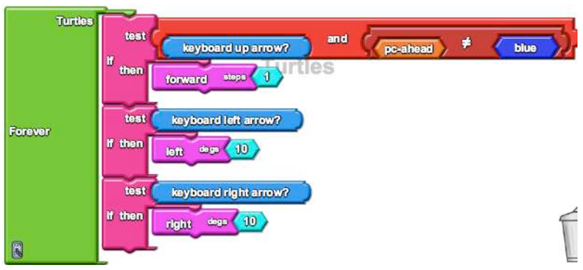

Figure 1. Programming blocks.

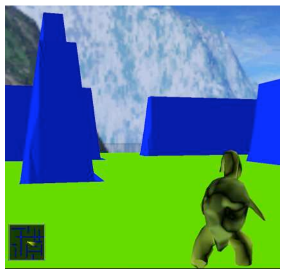

Figure 2. 3D maze.

Approach

Because of the significant differences between StarLogo TNG and previous versions of Logo, we decided that the math and physics approach to teaching the language was not the best use of its capabilities. The goals have changed from Papert’s pioneering work with the Lincoln School in Brookline, MA, but much of the core ideas remain the same (Papert 1979). We still wanted the students to be comfortable with telling computer what they want it to do (Papert 1979). We also wanted the students to learn the Logo block programming language and be able to make interactive programs while learning informally about some basic concepts of CS. Due to the focus as a game design course, the concepts in programming were introduced as needed. Most of the effort was put towards the interactivity and the creation of simple playable games. Certain concepts in programming lend themselves more readily to games than others. For example, forever and repeat loops made much more sense than recursion, game character interactions dictated that students understood the concepts of object-orientated programming as different agent breeds and behaviors. The timeline for when CS concepts were determined by need, not by difficulty level or their traditional place within a CS textbook.

Process

As the pilot project to introduce StarLogo TNG, we assembled a group of eight students (grades 7 to 9 with diverse backgrounds) from a local secondary charter school and formed an after school class around them. The class met once a week for 90 minutes and attendance was not mandatory. We were not sure how quickly students would be able to grasp certain concepts in StarLogo TNG, so we initially based the pace of the class on a class that used traditional Logo from prior experience. We planned to be somewhat more instructionist in our teaching style first rather than spending sparse meeting time for constructionist moments, as there were a basic set of programming blocks that the students needed to understand before they can do with it what they want. After the first of two such meetings, the instruction was gradually cut back to less than one-third of the meeting time. After the third meeting, the teaching process became much more constructionist, and we became programming counselors more than teachers. The identification of a newly needed game element would lead to a new problem that they would need to solve. That, in turn, leads to an application of their current understanding combined with new information that is scaffolded for them to build new game elements upon. The idea of a 3D Pac-Man style maze led to the need of having a first person perspective and the appropriate controls (see Figure 2). The idea of a loop that continuously checks for input was then introduced. All of these ideas introduced to the students had a practical need in the game design. The feedback we received from the students is also affecting changes to our software; block arrangement and organization, as well as the upcoming debugging features have all benefited from student input.

Evaluation

What we found was that it took students much less time to get started and use StarLogo TNG than traditional Logo, thus lowering the entry point for the language. Because of the inherent nature of block programming, as soon as the concept is clear, programming usually yielded instant gratification that the kids enjoyed. The nature of the programming blocks prevented students from making errors that would usually frustrate them; it kept their interest level high without getting bogged down. The visual blocks provided a certain amount of implicit programming guidance that text does not offer. When bugs did happen, the students ended up debugging their programming logic rather than syntax. The textual abstraction of their idea was visually represented in StarLogo TNG, and they often pointed to and followed the programming blocks as they were debugging. Even though the meetings were weekly, the highly visual aspect of StarLogo TNG made it easy for students to recall blocks they learned in the previous weeks. They only needed to recognize the blocks as opposed to the commands plus the syntax. In a short span of 5 meetings, students were able to create an interactive maze that provided a first person view and score keeping. Object-oriented programming came naturally, and when we introduced the concept of different breeds of turtles with their own set of behavior, it was no different from the students learning the repeat loop. Making a new object did not involve a series of new syntax, just more blocks. We conclude that the advantages provided by block programming lowers the entry point for programming, and the built-in error prevention mechanisms give students a more structured programming environment than that of the traditional text entry model.

References

Begel, A., & Klopfer, E. (2004). StarLogo TNG: An Introduction to Game Development. *The Journal for E-Learning*. (in press).

Papert, S. (1979). Final Report of the Brookline Logo Project, MIT AI Lab Memo No. 545.

Papert, S. (1993). *Mindstorms*. Cambridge, MA. Perseus Books.

Vegso, J. (2005). Interest in CS as a Major Drops Among Incoming Freshmen. *Computing Research News*, Vol. 17 No. 3.