StarLogo TNG – Making game and simulation development accessible to students and teachers

Eric Klopfer, *klopfer@mit.edu*
Associate Professor, Director, MIT Scheller Teacher Education Program, MIT Media Lab

Abstract

Prior Work on Learning Science Through Simulations and Models

For years we have been working with students and teachers on the Adventures in Modeling (AIM) program, a program which helps secondary school students and teachers develop an understanding of science and complex systems through simulation activities using our StarLogo programming environment. AIM offered computer modeling to science and math teachers as part of their professional development. AIM-trained teachers developed simulations that have become integrated into their standard classroom practices. However, activities in which students program their own simulations have been limited to a small number of classrooms. Our research has shown that there are several barriers to students developing their own models in science classes (instead of isolated computer classes) These barriers include the time it takes to teach programming basics, and teacher and student comfort with the syntax of programming languages. These barriers stand in the way of the deeper learning we have observed when students are given the opportunity to develop their own models.

StarLogo: The Next Generation (TNG)

To address these problems and to promote greater engagement of students in programming, we have designed StarLogo: The Next Generation (TNG). StarLogo TNG provides two significant advances over the previous version. First, the programming is now done with graphical programming blocks instead of text-based commands. This innovation adds a powerful visual component to the otherwise abstract process of programming. The programming blocks are arranged by color based on their function, which enables students to associate semantically similar programming blocks with each other. Since the blocks are puzzle-piece shaped, blocks can fit together only in syntactically sensible ways. This eliminates a significant source of program bugs that students encounter. Procedures are made from snapped together blocks, forming a visual chunk which helps students chunk the goal of the procedures in their mental models of the program. StarLogo TNG encourages students to spatially organize their block programs into “breed drawers,” which affords an object-oriented style of programming. StarLogo TNG’s second significant advance is a 3D representation of the agent world. This provides the ability to model new kinds of physical phenomena, and allows students to take the perspective of an individual agent in the environment, which helps them bridge the description of individual behaviors with system level outcomes.

Learning Programming Through Games

Our first take on introducing game development through programming was a course focused specifically on game design and development. The course was first implemented at a local secondary charter school for students ages 13-15. The classes were largely taught through “coaching” the students through projects. As the students conceptualized new game elements, they created for themselves new programming challenges that built on their current understanding, while requiring them to seek out or design new commands and algorithms. For example, the idea of a 3D Pac-Man style maze led to the need for a first person perspective and associated controls. To take direction from the game player, the idea of a loop that continuously checks for input was introduced.

We found that it took students less time to get started and use StarLogo TNG than traditional Logo (or StarLogo), thus lowering the floor for the language. Students were able to readily use the block programming, focusing more on concepts and less on syntax. The programming blocks prevented students from making errors that formerly frustrated beginning programmers and provided a certain amount of implicit programming guidance that text does not offer. When bugs did occur, the students only needed to debug 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.

This game design component has been extended and become the basis for a workshop for middle school students learning to develop computer simulations to better understand issues in their community. Instead of jumping right into the development of research-based simulations, the students first become familiar with the fundamentals of models and interactive design through developing model-based games. This empowers them to take programming and simulation into their own hands.

Learning Physics Through Programming

Many high school kids find that analyzing the world, as they are asked to do in the physical sciences, is difficult and uninteresting. But ask a kid to create a world and you get a different response. This motivation was harnessed to teach basic mechanics concepts in high school physics. Building on the experience in the game design courses, we designed a physics curriculum around game development. We felt that this unit was justified on the programming experience alone, but believed that the potential existed for significant physics content learning as well. StarLogo TNG basics were introduced through a series of task-oriented activities.

Programming skills learned in a game design unit were then harnessed to teach physics content in a unit we called Modeling Change. Beginning physics students have a hard time believing that the vertical and horizontal motions of a projectile can be independent of one another. The concept of simultaneous but independent change is difficult for them to grasp. A programming task was used to introduce the idea of simultaneous but independent change. They saw that changes in color, size and location which they programmed separately, could be run simultaneously. They had no difficulty seeing that their agent’s color change was independent from its movement in the x-direction. When students programmed vertical motion in the manner describe above, they were not satisfied. Their agents seemed to float up and down. To get realistic vertical motion, students needed to use procedures for accelerated change that were developed through a series of programming challenges. To get a turtle to jump over a ditch, they had to build and simultaneously execute separate move-forward and vertical-jump procedures.

The programming unit was followed by a unit on vectors and projectile motion. Assessments showed that the programming experience helped students learn vectors and see horizontal and vertical components of motion operating separately and simultaneously in projectile motion. In a report on a projectile motion lab, one freshman student wrote, “Programmers program motions independently in the virtual world, on the other hand physicists see the vertical and horizontal motion as independent of one another in the physical world. … I believe when an object is moving vertically it is independent and not interfered by horizontal motions. Our ball fell within our predicted value. This confirms our assumptions of vertical and horizontal motions being independent.”

Conclusions and Future directions

Teaching students to write programs can help them become digital producers as well as computer consumers. But they need powerful tools that can engage, motivate and empower them as they learn to program. StarLogo TNG is such a tool. Young people can use it to build 3D video games. This is highly motivating. TNG also supports the construction of scientific models. There is a productive overlap between games and models, and we have seen student game-makers become student model-builders, building and using scientific models to enhance their content learning. Our work is driven by the vision of many students using StarLogo TNG to program exciting 3D video games driven by scientifically sound models.