Thinking Skills and ToonTalk

ToonTalk is a fertile and playful environment for children (of all ages) to learn the following critical thinking skills:

- **Problem decomposition.** When a child tries to build anything beyond the simplest program in ToonTalk, they are immediately faced with the task of breaking the problem down into "robot-sized" (or more ideally "mind-sized") pieces. When done well, it is then easy to build or program each piece. This is a very general design skill that applies throughout science, engineering, and the arts as well. For larger problems, there is a hierarchical structure to this activity, where problems are broken into pieces and the pieces are in turn broken into smaller pieces.
- **Component composition.** This is the second half or dual to problem decomposition. Just because one has pieces that work in isolation does not mean that it is trivial to compose them. There are usually interactions between the parts that need to be dealt with. Often components can be composed in different ways, only a few of which work. Again this is a very broad design and problem-solving skill. Difficulties composing parts often leads to redesign of the problem decomposition. Some argue that this is a special case of the more general "debugging" skills one acquires while programming.
- **Explicit representation.** Software that models something, whether it is a bouncing ball, an ant colony, city traffic, or an ecology, needs to have data structures that represent something else. For the ball, the child may create a structure that holds the ball's position, speed and direction of motion. For an ant it may be the ant's level of hunger, energy, and a representation of the state of various sensors. The ability to design a good representation for a model is critical in science and engineering.
- **Abstraction.** This is related to "explicit representation". Software can be very specific or very general. Consider for example the sample program in ToonTalk which swaps two numbers when the first is bigger than the second. When first constructed the program only works when the first number is 2 and the second one is 1. It is then abstracted to work for any two numbers where the first is larger than the second. It could have been abstracted so that it would work for words as well as numbers. If a word is alphabetically after another, then the robot would swap them. The ability to abstract when needed is a crucial thinking skill. ToonTalk is special in that it encourages children to work through concrete examples and then abstract the results.
- **Thinking about thinking.** Seymour Papert has written extensively about how the right programming environment can facilitate children thinking explicitly about how they solve problems. (See his books *Mindstorms*, *Children's Machine*, and *The Connected Family*.) If, for example, a child is trying to build a program to play tic-tac-toe, they are faced with

questions of how the computer is going to decide which move to make. They need to think explicitly about how they make such decisions in order to program the computer to do so. Papert claims that one becomes a better learner and a better designer and a better problem-solver if one is able to explicitly reflect upon one's own thought processes. And this reflection is much more effective if one has some model of thinking skills like the list presented here.

The argument for ToonTalk isn't that it, or even computer programming in general, is unique in providing an environment for learning these thinking skills. ut that ToonTalk is a rich environment where these kinds of thinking skills are "exercised" frequently in a natural context. ToonTalk is an environment in which there are fewer hurdles to overcome (like a programming language syntax or learning to play a musical instrument and to read music) before one begins to be productive and begins to learn these thinking skills. ToonTalk is a fun, appealing environment that maintains a child's motivation.