Everything You Ever Wanted to Know About Programming at TIC
by Roshini Sandhlla, Former TIC Programming Director

Computer programming was the only technology option offered at TIC for many years, and it remains the most popular. But while programming is not new to TIC, it is new to most of our first-time campers. Thus, it inspires more questions from campers and their parents than any other option. Here we answer such questions as:

- Does learning programming have any value to campers who don't plan to become professional programmers?
- How does the camp decide what language to teach each camper? Why are first-time programmers virtually always placed in *Microworlds LOGO*?
- Do professional programmers only use C++ or Java? Are these languages better than LOGO?
- Why are parents discouraged from purchasing *Microworlds LOGO* before the last day of the session?

Why Learn Programming?

The vast majority of TIC campers will probably not become professional programmers. So, why study programming at all? For one thing, computer programming teaches problem solving skills and logical reasoning. Writing a computer program requires the camper to develop an overall plan for her project, and then break the program down to a set of specific functions that the computer can perform. For instance, if a camper wants to make a horse racing game, she needs to create a function to draw the race track on the screen and to display the horses at their starting positions; a function that uses keyboard input from the players to move the horses; a function to prevent the horses from straying off the track; and a function that tells when a horse reaches the finish line and announces the winner. Each function must be expressed as a sequence of decisions the computer must make, and she must write these decisions as True/False conditions using the appropriate variables. Can seven-year-olds do this? They can at TIC.

No computer program works correctly the first time – either for campers or professional programmers. In "debugging" their programs, campers learn to think on their feet and to deal with unexpected challenges. Fixing all the "bugs," or errors, in a program requires both the flexibility and creativity to try several different approaches, and the persistence to not give up when things don't work as expected. Debugging problems are usually solved not through a moment of brilliant insight, but through persistent hard work and determination. Debugging also allows campers to better understand how their code works, and why their code must be written in a specific way in order for it to run the way they'd like it to.

Campers at TIC work in pairs, with two campers sharing one computer and choosing a single project to create together. At first, this might seem half as valuable to the campers as letting each of them work by themselves. On the contrary, placing the campers in pairs makes their experience at TIC many times more valuable, for one simple reason: collaboration. In working collaboratively, the campers must learn to communicate their ideas to their peers, to make compromises, to fairly divide the workload involved in creating a program, and to share ownership of the finished project. Because writing a computer program involves constant decision-making, two campers working together must constantly discuss and negotiate. Campers are able to creatively and uniquely design a game of their own, and work through bugs as a team. In learning to work through their differences and to agree, campers gain perhaps the most valuable skill that TIC has to offer.

Programming requires campers to learn many mathematical concepts. Even a "simple" game like the classic arcade game "Pong" requires an understanding of variables, Cartesian coordinates, Boolean algebra, and vectors. Before working at TIC, I would never have believed that seven-year-olds could learn basic high school algebra, or that 11-year-olds could learn to use trigonometry. But I've seen it happen. Microworlds also removes the

need for drag-and-drop programming, which denies

campers the opportunity to gain a firmer grasp of programming by writing and editing code themselves. It allows campers to proactively think about what they are typing, and *why* they need to type their code a specific way, while simultaneously learning, no matter how big or small the impact left behind.

Because TIC has one counselor for every four campers in programming, the kids receive individual instruction at an appropriate pace. This allows them to learn far more advanced concepts than they are taught during the school year. And because they learn these concepts on a project of their choosing, the experience is more like "fun" and less like "school." Teaching campers that learning can be fun is itself a valuable achievement.

Language Placement at TIC

We don't mean the language placement process to be a mystery, but it is inherently complicated because we must consider many things. Campers' and parents' requests are always taken account of (especially if the request is made on the forms sent in prior to the start of camp).

However, we don't necessarily honor language placement requests, such as: "My son wants to learn Java." Why not? Sometimes there are logistical reasons. Each counselor has four campers of about the same age and experience level. If all four campers are not working using the same language, you could have 2 campers work in one language, and two campers work in another language. So, for instance, if a 12-year-old senior has worked with LOGO for two summers, and wants to start learning Java, but there are 3 campers in your group learning LOGO, then you are left with an uneven partner group ratio. The 12-year-old would most likely, if group changes are not able to be made, be working in LOGO for a 3rd summer. This isn't a bad thing, but must be seen as an opportunity to further increase a camper's skills in LOGO. The better they are now, the easier learning a second language will be.

However, the most common reason for not granting a language placement request is not logistical. Language placement is governed

feel is best for teaching this camper? The reason why we assign virtually all first-time campers to *Microworlds LOGO*, for instance, is because LOGO is the ideal language for teaching programming to first-time campers. The LOGO language was designed to teach programming to children. It can be seen as a base for more widely used programming languages, and allows younger children to be exposed to the world of computer science in a language that is intuitively easier for a seven-year old to understand, with the ability to also artistically express themselves with the expanded set of graphics tools that Microworlds provides them with.

Unlike C++, where a single mistake can produce 50 inscrutable error messages, or even cause the computer to freeze-up and require a reboot, an error in LOGO will produce a single message in plain English, such as "I don't know how to *run*" or "*high_score* has no value." Also, campers usually enjoy producing highly visual, graphical projects. While most languages do graphics with a complicated, coordinate based system, LOGO represents moving objects on the screen with a concept called the "turtle." The turtle responds to many simple commands, such as: move forward a certain distance, or make a right turn. Campers can give the turtle a command and immediately see its response. By contrast, most computer languages require many lines of code just to initialize the graphics, before even a single dot can actually be displayed on the screen. In LOGO, campers don't need to work for long periods of time without seeing any results. The language gives campers immediate feedback and makes programming an interactive experience.

While LOGO is relatively simple to learn, it is still a real programming language, based on structures and concepts that are typical of programming languages. You can think of it as a watered-down version of Java, which is why it's pretty simple for a seven-year-old to pick up over the course of a two-week session. Programming in LOGO involves procedures, conditions, variables, Boolean logic, the evaluation of complex expressions, various sophisticated data structures, argument passing, responding to keyboard input,

primarily by one question: What language do we

and all the other tasks and concepts used in any

programming language. Learning to program in LOGO gives campers a strong foundation for programming in any language they choose.

TIC's goal is to introduce and teach campers programming, teamwork, and other life skills mentioned above. The goal is not to teach programming languages. Only to expose them to it, and to instill a true interest for future learning of computer science. For an experienced programmer, learning new languages is a simple matter, regardless of what language they are used to working with. (For example, I myself first learned programming as a camper working in LOGO at TIC, and have since learned C++, Visual Basic, QBasic, Fortran, OpenGL, Java, Pascal, SQL, and Forth. I am basically self-taught in virtually all of these, with the exceptions of LOGO and BASIC. So I feel that I have some authority to say that learning programming is challenging, but once you know programming, learning languages is easy.)

The Myth of the Language Hierarchy

So, am I saying that the point of learning LOGO is to make it easier to learn C++? That is the eventual goal, right? After all, everyone knows that real programmers only work in C++, Java, or maybe Python. Right? In fact, the popular myth that professional programmers only use these languages is just that: a myth. It is true that these are currently some of the more popular languages, but professional programmers understand that every language has strengths and weaknesses, and generally prefer to work in the language best suited for the particular sort of programs they are writing. (I had a professor in college who could give you a dozen reasons why he'd rather work in the relatively obscure language Forth than in any of the more popular languages.) Also, it is worth noting that today's computer languages may well be supplanted in popularity by the "next big thing" by the time many of TIC's current campers reach adulthood. However, even now, these languages hardly have a stranglehold on the programming industry.

LOGO itself, although still mainly used for educational purposes, lends itself very well to recursion and fractal generation. (In fact, for campers interested in learning to produce fractals, LOGO is far-and-away the most obvious choice of any of the languages we teach at TIC). Moreover, LOGO is closely related to LISP and has been called a dialect of LISP. LISP is particularly useful in artificial intelligence, and LOGO is also well suited for such applications.

The Real Purpose of TIC

The only programming language we sell at TIC is *Microworlds* LOGO, because that is the only language we are licensed to sell. Other programming languages can be purchased from software stores, college bookstores, websites, and other sources. But why do we prefer that parents not purchase *Microworlds* LOGO until the last day of the session? The reason is that camp isn't school, and we don't want the campers to have homework. Three hours of programming a day is already a lot, and if campers work at home on their projects, they risk burning out and no longer feeling as motivated to work during the camp hours. Also, with two campers collaborating on each project, it isn't fair for one camper to be doing extra work on his own without his partner's involvement.

TIC pedagogy is based on the idea of intrinsic motivation: campers learn and produce their projects not because we tell them they have to, but because they have fun doing so. With their own enthusiasm to fuel them (and a 4:1 teaching ratio), campers learn far more in two weeks than they would in a much longer period of time at school. Even if campers learned more doing extra work after camp, it wouldn't be worth it if it made camp less fun, and if it deprived them of the discovery that learning can be fun.

TIC is a summer camp, and like any good camp its real purpose is to be a place where kids can have fun. The fact that they can have fun learning is a gift they will never forget.