CREATIVE CODING

# HOW TO BUILD ON THE SUCCESS OF THIS PAST WEEK'S SUMMER CAMP

If your child is one of those who is really excited about making things with their computer, they will want to do great things. And they will soon begin to create pretty complex code. We're talking long chains of commands filling up more than 2 screens. Lots of nested loops (yellow code blocks enclosing other blocks in loops. Ask your child). Some of them are already there, and beyond.

When this happens, the very thing that makes Scratch so great for beginning programmers will begin to frustrate intermediate beginners, like most of your kids are now.  Scratch is great because instead of having to type code letter-by-letter, kids can just drag-and-drop building blocks of commands, like Legos. So they focus on the structure and method, rather than the typing.

This is great because it keeps kids from getting frustrated by missing semi-colons or misspelled words, the bane of manually-written code.

But it's bad when the kids begin to create long complex blocks of code, because of the quirky way Scratch makes assumptions about what you're trying to do.

## QUIRKY INTERFACE ALERT!

HELP YOUR CHILD AVOID FRUSTRATION! Here's the quirky thing to be aware of: As you move code around, you can't just grab one line.

When you try to drag and drop a line of code to move it away or to a new location, Scratch will grab ALL THE CODE BELOW THAT LINE, TO THE END OF THE LOOP. Most of the kids have become pretty adept at dealing with this quirk by now, but when the code gets really complex, putting the code back in the right loop requires some precision mouse work, and you really have to pay attention to what you're doing.

For this reason, intermediate beginning programmers who are trying to change a bit of code will often do collateral damage to existing, working code that is nearby, resulting in some very weird bugs. Basically, stuff that used to be working will stop working.

Here's the deal.

When you see them get really frustrated, their ambitions have begun to outpace their abilities.

I highly recommend that when this happens, you suggest they take a break from that project, and start something new and totally different. Get that working, regain some confidence, then they can come back to their other project later if they want. But it's probably better if they abandon it.

Bottom line is that they will learn more if they spend some time creating a lot of short little programs that do clever little things, rather than embark on some magnum opus with lots of complex code.

Challenge them with a lot of short little animations or "mini-games" that are very different. Encourage them to see what they can accomplish by combining new commands in unexpected ways.

If they never get back to that epic they started, that is OK. It is actually better to start something new at this stage than to finish something where they've gotten over their head. No program is every really "done" because you can always add more complexity. You just eventually have to say "let's stop working on this."

At this stage, a lot of small victories and "pilot projects" are better than one big monstrous project that is never quite working. Those small projects will help them really get solid in understanding the core, basic concepts that underlie all modern computer languages. This will serve them well if they want to program in a type-written language like Python when they get a little older, and it will better accomplish the real goal

After all, the goal of this class is not to become an expert on Scratch. That's just the tool, not the subject. The subject is gaining confidence in their power over computers by writing code, thus unlocking their creative potential through technology for the rest of their lives.

I think they have that confidence right now, but it is a bit fragile. Let's not let them defeat themselves by trying to slay twenty dragons next week. Keep it fun, keep it short, start many things. That'll help them build on that confidence. Small victories. That's my advice for now.

## WHAT IF MY KID ASKS ME TO HELP THEM CODE?

*No worries. Even if you know nothing about computers, you can help them.*

*1) Praise their accomplishments, and encourage them to show you what else they can do. Be their cheerleader! Let them do the learning, while you do the encouraging.* ***This is really the key, and it's enough.*** *If they really love this, they will continue to learn on their own, so long as you are encouraging and praising them.*

But, if you do want to jump in, here's what I suggest:

2) Say "I don't know anything about this, can you show me how this Scratch thing works?" They can teach you.

3) Say "Ok, so you're saying this is code. Can you explain what these commands mean, and what they do?" In explaining the code, they will often find the solution. In fact this is the crucial step in the de-bugging process: they need to read the code line by line, like the computer does, and try to understand what is actually happening.

4) Here's a hint: **they can right-click on any command, and the word "help" will appear**. Tell them to click that for a brief explanation of what that command does. It's not much info, but it's a start.

I hope this has been helpful.

Sincerely,

Eric Fredrickson
Owner, Teacher
creativecoding.com
Please Like us on Facebook!