Interactive games and simulations may make online learning fun but they add significant cost and production time. But this will not always be the case.

For example, 10 years ago, Visual Basic was considered a toy and its programmers mere "code kiddies", claiming they could produce business solutions faster than anyone else. Likewise, today's code kiddies and their toy programming tools will probably fashion the future of e-learning and possibly the rest of IT.

Over the holidays, I caught a few of these upstart whiz kids in my own house. Two of my sons, David, 11, and Daniel, 9, learned how to program in a day and started building interactive applications. Then they taught a friend in 30 minutes flat.

Their tool is Game Maker - a codeless system of sprites, (pictures or animations), sounds and objects, linked by sophisticated preset events and actions. These are supplemented by backgrounds and rooms (screens), mixed with scoring, health and other game concept objects.

Within hours, the children were building programs that would have taken their old man, a professional programmer, weeks to code in Visual Basic.

Fourth-generation programming tools promise much the same convenience.

That does not mean creating event and action relationships between Game Maker objects is simple. The scariest part was my nine-year-old seemed quicker at debugging this hairy drag-and-drop world than I was. Yet their comments showed universal programming lessons were being learned.

"Sometimes when you need an example you can go to their games to see how they do it," Daniel advised me. "It's not like you get it right the first time."

The Game Maker site (www.gamemaker.nl) shows how to use the two-dimensional play system in an educational context. But the tool is not impressive only because it enabled my nine-year-old to start programming five years earlier than I did. What is amazing is the type of interactive applications it compiles is similar to that commanding big bucks in today's corporate e-learning world.

Of course, Game Maker is not an e-learning tool, so it does not support the industry's standards or even the basic types of form fields found in HTML. But it does support end user interactivity, two dimensional simulations and games similar to how I think tomorrow's e-learning tools will work.
"The best one I'm making right now is probably my billy-cart game," Daniel says. "You go around and there's obstacles you have to dodge and you can play with two players and you cross the finish line to go to a different level."

After seeing her brothers in action, an older daughter started planning her "Mission to the Moon" game. I realised she was not thinking about the tool at all - while by comparison, my imagination seemed terribly code-bound.

My children have yet to learn how to use variable, program flow and timing objects - or delve into the underlying language if they ever need to. Yet like millions of spreadsheet users, they belong to a generation whose sophisticated e-learning applications will be inspired by bright ideas without necessarily knowing code.

"It's fun playing it because it's your type of game instead of finding one on the internet," David tells me. "You can make it how you want it."

This is exactly what business wants to hear.

The same thought occurred to me in 1979 when I was looking at the crop of next generation programming languages starting to appear. Liberated from the painstaking line-by-line complexity of conventional code, today's kids can think much bigger thoughts.

"And you can do something with this you can't do with games from the internet," Daniel says. "You can sell the games you make. It says you're allowed."