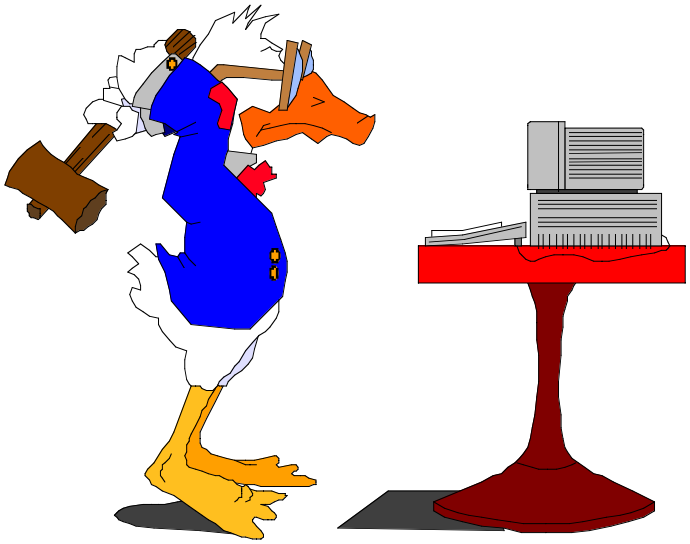
Geek Learning

The Benefits of Discovering Your Inner Geek

By Caleb John Clark 08/27/00



Abstract/Author Bio

This paper posits that technology-training professionals can benefit from studying what makes geeks tick, where they get their motivation, and how they learn. Then one can temporarily tap into their "inner geek" to improve their ability to design engaging learning and performance support systems. The author defines 7 specific actions and provides real life examples using the acronym "GEEKOUT".

The following observations came from the author's walking between the world of geeks and the world of people persons. Caleb John Clark is not a native geek, but he has a lifetime passport to travel there. He's been walking this strange line since he began building Web pages since 1993. Caleb was also a Web master in the Bay Area in 1996, and recently spent two years as a "Techie" in the Instructional Media Lab at San Diego State University while completing a Masters in Educational Technology where he also taught a graduate course in Web design. His portfolio is at http://www.plocktau.com

Table of Contents

Introduction

Geeks Rule! No longer the chicken head eating freaks of the past, they now count one of their own as the richest man in the world. Society has stopped scorning geeks and now sees them as modern hip revolutionaries and even viable and attractive partners to the opposite sex.

Why? Because geeks “get” technology in a time when technology often seems out to get us.

At the same time, businesses need to help their people get technology.

This article focuses on the challenge of fusing those two truths. It posits that technology-training professionals can benefit from studying what makes geeks tick, where they get their motivation, and how they learn. Then one can temporarily tap into their “inner geek” to improve their ability to design engaging learning and performance support systems.

We’ve all been in the presence of true geeks. They have that underlying impatience that seems to scream, "I don't respect you because you are not in love with technology for better or worse." They often get blank looks when you ask them, "How did you learn that?" and answer "I just figured it out."

Are they born such intrinsic motivation? Are they a personality type? Was there something about playing the roll playing board game Dungeons and Dragons in high school that prepared them for the information revolution? (actually I think there was, but that’s another paper).

Anyway, who cares? Let’s move on to trying to find out what they know that can help us, and what traits they naturally have that we can temporarily emulate to our advantage. Ok, it’s time to geekout!

Goof around

There's an old joke that you know you're a geek if you've ever sent email to a person sitting next to you. My definition of geek is a person in love with technology, computers and the Internet. Someone who talks tech. at the drop of an acronym, is easier to email then to call, and absorbs new technology like a sponge.

Geeks are the kings of goofing around on computers. Only under duress will a geek learn a new software program while doing anything even remotely resembling important or productive work with any sort of deadline. To a geek, people who do things like that are the ones who come to them in a panic begging for help.

Goofing around may seem like a waste of time to adult learners, but it is not. When dealing with computers, this kind of non-evaluative environment permits learning without fear. Castleman (1995) effectively reduced phobias and increased confidence by creating a non-evaluative, non-threatening environment for students to learn in.

And so ,when a geek sends six emails with nude pictures attached to them in different ways to a friend in the next room, and then spies on his friend as he checks his mail, is he wasting time? No. Because when he has to send his resume as an attachment, he'll have practiced enough to get it right. And when your resume fails to arrive, you'll go to him and wonder "how'd he learn to do that?"

This idea of wasting time early to save time later employs aspects of Edward De Bono's four critical factors of Lateral Thinking: 1. Recognize dominant ideas that polarize perception of a problem. 2 Search for different ways of looking at things. 3. Relax rigid control of thinking, and 4. Use chance to encourage other ideas (TIPS database).

Goofing around with computers also fits more with the decentralized nature of the new information based world we live in. Computers and the Internet are complex,

decentralized, chaotic hybrids of human and machine. And humans tend to assume centralized control, e.g. that a flock of birds has a leader when it doesn't. (Resnick, 1996). As Mitchell Resnick (1997) at MIT's Media Lab's Epistemology and Learning group recently said in an interview,

> What we need to do is rethink the process of learning in a way that's suited to the digital age. The traditional approach of the teacher who transmits information to the learner is very centralized. We need to find way for learners to take more responsibility for their own learning (p. 136)

Geeks are splendid at self-paced playful learning. But playing games on a computer is also counter to adults' instincts to just get work done. This however, goes against all geek thought, as well as research. Cattleman (1993) found that computer games helped reduce phobias and increase confidence in early childhood education majors' learning about computers.

Geeks not only play a lot of computer games, they make most of them. Using a computer to play a game takes the act of using the computer down a notch or two in seriousness and gives the user a feeling of *control* over what the computer does. The computer disappears into a simple tool leaving only the game. To a geek, most work on computers is a game.

Action: Make time to fool around, bring in a game early in technology exposure to give "newbies" a sense of control. Then encourage experimentation and game playing with software.

Example

A student asks if you can copy text from an email to a Word file. I love this question. I tell them yes! In fact you can copy and paste text between *any application*, e.g. Word to email, Web page to Word, email to spread sheet. Cutting and pasting between applications also has built in motivation and relevancy because it ends re-typing. It also teaches the beginnings of what an operating

system does. Students sometimes ask how I learned how to do something like this. “I just tried it one day when I was playing around. I was copying text within files all the time and wondered if it would work between applications, so I just tried it. Now I don’t retype anything.” I tell them.

Explore

As Terence (185-159 B. C.) said, "Nil tam difficilest quin quærendo investigari possiet.” Loosely translated into geekspeak, this means, "There's nothing that you can't find by checking all the menus and toolbars".

The first geeks who learned HTML back in the early 1990s, before books on the subject, did it by partly by exploring their Web Browsers and stumbling onto the "View Source" option which reveals the HTML of the page you are looking at. With a little exploration one can compare what they see in their browser with the HTML and figure out the basic nature of the language of HTML. With a simple "Select All" and "Copy" one can bring the HTML down to their computer and then open it up in a text editor and start playing around. Change some code, look in the browser, and repeat. This is how I learned HTML in 1993 and how I entered a world of web developers that used "view source" to see and copy each other's code, becoming a community of open source learners.

Right now this free HTML tutorial is still hiding in everybody's web browser as the "View Source" option, yet only a very few people ever discover it.

Exploring the menus and tool bars is an activity I call this the MITS, or Menu Item and Toolbar Search. MITS is an expansion of the Menu Item Search Strategy (MISS) created by Professor B. S. Allen (personal communication, November 24th, 1997) at San Diego State University. MITS is simply the act of starting from the left and dropping down every menu item (“File”, “Edit”, etc.), and searching every toolbar, for the command you need.

There was a running joke in our computer lab that if everybody did the MITS before they asked a question, we'd be out of a job! Geeks do the MITS instinctively because they know that menus and toolbars represent a tree-like schematic of what the software is capable of doing. Realizing this, the most natural thing to do when one is stuck is to:

1. Phrase the question in your head, for example "I want to change the size of this Image"
2. Explore all the options in front of you looking for key words in your question such as "Image" "Size". Sure enough, you'll soon find something like "Image Size" in the menu or toolbar of most graphics programs.

This turns out to be much easier then remembering the actual steps to do something, a sure sign of a beginner. Geeks tend to remember how they found the answer, not the actual steps because it is much less of a cognitive load to remember "I found it by looking at all the menus" then the actual steps of hundreds of actions. If you watch a geek work they will often drop menus down in rapid succession because they can't remember the details of the software they are looking at.

Mitrani (1990) pointed out that one crucial difference between computers and other media is that they are interactive and that interactive things require interactive techniques for learning.

The MITS is a great way to get people communicating with software, to empower them to find their own answers, to show them options they never guessed were possible, and to foster exploratory learning in an indirect way.

But the real value of getting your MITS on a piece of software is that it teaches you the holy grail of geek software knowledge – all software is pretty much alike and share a few common metaphors. For example, Word processing uses margins and tab, graphics programs use the same basic tool bar, animation and video share timeline windows, etc. This is why geeks seem to learn software in an afternoon. To them it is only a matter of learning a few specific details of the new application.

Action: Teach the MITS as the first step when stuck.

Example

On my shifts in the computer lab I used to use the MITS right in front of students who thought I knew the answer to their questions. I didn't. I just found it so fast that they thought I did. It's not practical for me to remember the exact steps for doing things in 20 odd pieces of constantly changing high end software. All I must remember is to explore. I found it once and I'll find it again. And it's not practical for newbies to remember or write down steps either.

Students often panicked as I did the MITS and demanded that I slow down so they can write down each step. I tell them that there's only *one* step, the MITS. Remember to do that, and they'll always find the command they need - and they'll have so much less to remember!

Embrace (at least temporarily) technology

PCs in their current state are crude, hard to use, dehumanizing tools. In an online draft of Donald Norman's (1997) upcoming book, he says computers are difficult to use because they are designed "by technologists, for technologists", and that in their current state, they are way too digital to be used without frustration by biological beings like us humans.

Yet geeks obsess over them, why? Because geeks believe in technology as a whole and they love it for what it is, flaws included. Andrew Yeaman (1992) defines computerism as "Blind faith in the inherent good of computers". Geeks often fall into this definition and don't consider the possibility that others might not. To a geek, technology represents the greatest tool ever invented by man, and possibly the only chance our species has of surviving the next million years. This is important stuff. This embracing of technology as a whole is like love in that it creates passion and acceptance. This is how a geek can gush over the newest

billion MHz chip, coddle their PC when it refuses to print, be frustrated when their computer keeps crashing, but always forgive it once they fix the problem.

Action: Avoid creating expectations about technology working all the time. Teach the relative place of the current technology in the short history of the tool. Suggest embrace and acceptance.

Example

Sometimes students are just plain angry with computers for being so hard to use. When I sensed this frustration in the lab I tried impart some love of technology to them by pointing out that ever since we first used sticks to start bonking our fellow monkeys on the head, we've been a tool using species. Nowadays the greatest tools we have are computers, but we're just learning how to use them and they are crude. Just as our first cars were hard to drive, our first computers are hard to drive and geeks do not necessarily expect the crude computers of today to function *at all*. When they do, geeks are amazed and happy. Thus, geeks are constantly amazed and happy with technology.

Everybody knows deep down that anger and hate are self-destructive, they just need reminders when dealing with technology.

Kill fear

Where there's fear, it is hard to learn. As Yoga said in the Episode one of Star Wars (and the quote on my multimedia class web page) "Fear is the path to the dark side, fear leads to anger. Anger leads to hate. Hate leads to suffering." As you might have experienced, people who hate technology usually end up suffering around it.

Castleman (1995) found that phobias block learning and a willingness to ask for help and are related to how much control people feel they have over what computers do. And Filipczak (1994) defines technoliteracy as not only

understanding what the technology can do, but also overcoming fear of the machine.

Fear of computers is an alien concept to geeks. They seem to have been filled with fearless wonder from the time they first saw a computer. Why is this?

First I'd say it's because geeks immediately see that since there are few moving parts, computers are pretty hard to break.

Second, geeks know that the nature of all software is that you can always reload it. So, if it's hard to break, and you can reload software, the only real fear is fear of losing your own work. From my experiences this is also the most common type of fear in adult learners. This is understandable, since most people don't have the slightest idea where, or how, data is stored on a computer. Given that lack of knowledge, relying on a computer to save a month of work can be terrifying!

Action: Kill fear like a geek does: 1. Save every 10 minutes buy instinct. 2. Back up data to another drive or disk every time significant work has been done. 3. Give descriptive names to files and spend time organizing them to avoid mistakes.

Example

Joan won't tell me how old she is, but I'd say a youthful 60. When she's not baby-sitting her granddaughter she's in the computer lab. She's taken two 400+ level multimedia production classes at SDSU and is in her third. She started with no computer skills and tells me that she was frozen in fear in her first class, because to her computers were, "very complicated and I had respect for them and I didn't want to do anything to hurt them." Being scared she said she was coddled by the instructors. "I like to learn the theory behind things so I can do what I want, but they gave me too much step by step instruction because I was so scared and older. It doesn't mean I'm stupid because I'm scared, I was just out of my element."

Now she tells me she's no longer scared, that she even comes in early to read the manuals and she's very excited about what she can do with her new skills. I asked

her what was the key to killing her fear? “Motivation to use the machine to do what I want,” she said.

Open the Manual

There's an email acronym geeks use amongst themselves when a question is asked that reveals a lack of opening the manual. It's RTFM, for Read The F****** Manual. Over the years, I've received several such emails due to laziness. I usually grumble and open the manual and inevitably not only find what I want, but some other time-saving way to make the software do my work for me. Geeks are autodidactic and love to try and figure out software just by exploring, but they also know the value of reading manuals, taking tutorials and using on-line help when they are stuck.

Action: RTFM

Example

AppleWorks is a pretty easy program with a short manual. When a student starts asking a lot of questions about AppleWorks I give them the manual. They groan and do a little hemming and hawing, but usually about a half hour later they'll come up smiling and say something like, "check this out, you can put a working spread sheet into a word processing letter - and you can also shrink the view to see the whole page, and..."

There are times when students should use manuals. At first with the help of teachers who look up answers with the student, then all by their lonesome. Using manuals teaches learning to learn skills and increases self-confidence.

Understand the basic nature of the beast

For novices, the technology, especially telecommunications, is a mysterious black box. They don't get it. And they need to, at least a little.

Faulty concepts that are a way of propagating misunderstanding. My favorite example is the often heard phrase "is that on Netscape?" to refer to a Web site. I'm on a personal crusade to eradicate that expression from the planet. Once and for all NOTHING IS ON NETSCAPE! That's like saying "Is Seinfeld on Sony tonight?". Netscape is one browser made by one commercial company. A browser simply lets you tell computers on the Internet to send you the files that make up a Web page. The browser interprets the files so you can see and use the site.

After many years of helping students with problems using computers and software, Gregg Koyamatsu, the head technician and software manager at the Instructional Media Lab at SDSU, says the single biggest problem is adult learners lack of understanding about the basics of computers. Basic skills, like saving documents to the hard drive vs. a floppy, or using the operating system to move between different applications, are the biggest stumbling blocks to beginning users, says Gregg.

Adult learners need to know that computers use software called an operating system (Mac OSxx or Windows xx, the next big thing xx) to run specific software like AppleWorks or Word. Too often adults are put in front of a computer and immediately taught an application. In the future this may be the thing to do, but with computers as crude as they are now, it is not. That is why Geeks so often are seen just using the computer, not necessarily an application.

To a geek the computer and its operating system represent an instrument like a guitar. But unlike a guitar, this instrument can change and become any instrument in the world. The operating system enables the computer to make these changes by letting it run different software for doing different jobs, like Photoshop for graphics

or Microsoft Word for text. All too often adults think specific software is the instrument when really it's only an application *of* the instrument - In this light it makes sense that specific software programs are called *applications*.

Action: After initial exposure to a technology, step back to operating system skills, and basic conceptual frameworks.

Example

An interesting and successful way of showing adult learners the nature of the Internet is to make it come alive by having people act it out. Yoder, (1996) found success with college students when she made big signs that represented file servers on the Internet and had people hang them on their necks. Then they would throw an email sign back and forth to represent the path an email letter takes. This is an excellent conceptual lesson for the nature of email.

Transfer concepts between software

Perkins, & Solomon, (1984) defined transfer of knowledge broadly as a side effect. From learning X you find you improve your knowledge in Y. Geeks learn lots of different software, after a while, they all start to look the same. And in fact, there are many design conventions that are the same in all software, regardless of the company. Most applications have "File" and "Edit" menus starting on the left. "Open" is usually under the "File" menu. Databases, let you "Tab" between text fields. Graphics programs have a marquee tool and will force a line straight if you hold down "shift". Word processing programs use a ruler with margin controls just like an old typewriter's.

This is about learning to drive software like you learned to drive a car. When you buy a new car, you don't have to learn how to drive all over again, you just learn how it handles differently from your old one, and you're driving confidently in a few hours. When geeks get their hands on new software they just have to learn how it handles differently.

Action: Teach new software by using what learners learned with the last software. Enforce and make evident common properties between software.

Example

Every semester there are a few learners who have a very hard time with computers. With one such learner I found myself explaining the difference between a "text environment" and a "graphic environment' when using AppleWorks drawing program. I said, "watch the screen very closely, notice what changes," and proceeded to click between a text box and a graphic object. I kept clicking and we clicked back and forth between modes for *five minutes* doing nothing but looking at all the things the software was telling us. If you wanted to go further, you could boot up Microsoft Word and find all the similarities between Word and AppleWorks, and then boot up Photoshop and compare it with AppleWorks drawing program. In half an hour, you can show students that the mountain of software out there is more of a molehill.

To be an efficient self directed learner, you have to be able to transfer every last drop of knowledge that you can from one skill to another so you don't waste time relearning skills.

(NOTE: These seven strategies make the acronym GEEKOUT)

Conclusion

The 1914 definition of geek reads: 1 : a person often of an intellectual bent who is disapproved of 2 : a carnival performer often billed as a wild man whose act usually includes biting the head off a live chicken or snake (Merriam-Webster 1997).

Nowadays geeks rule. Geeks are now enmeshed at a very high level in every facet of the information age that is changing the world spinning under our feet. Geeks

are literally building this new information world by producing the software and hardware that make it run. Geeks get work, geeks are becoming attractive partners, and geeks are actually cool! The times they are a changin!

Perhaps a new definition might read: Geek: 1. A person prone to energetic adoption of emerging technology, esp. computers and the Internet. 2. One who is a self-directed and playful learner of such technology

Geek learning is basically self-directed learning with a heavy dash of exploratory learning thrown in for good measure. But support for self directed learning in higher education has been shown to be very low (Wilcox, 1996). This is evidence of the resistance to the new information age, a natural, but dangerous, reaction to chaos.

I believe that the chaotic and decentralized nature of the exploding information age demands a re-examination of theories like “self-directed” and “exploratory” learning - terms coined by non-geeks to describe things geeks do naturally.

In these times, teachers and instructional designers grappling with how to facilitate technology learning should heed the advice of the old expression “When in Rome" and just let it go and GEEKOUT!

References

Castleman, J. B. (1995). Decreasing computer anxiety and increasing computer usage among early childhood education majors through a hands-on approach in a non-threatening environment. Doctoral dissertation submitted to the Ed. D. program of Child and Youth Studies, Nova Southeastern University. (ERIC Document Reproduction Service No. ED 389-271).

Filipczak, B. (1994). Technoliteracy, technophobia and programming your VCR Training, 31, (1), 48-52.

Merriam-Webster, Incorporated (1997) on-line dictionary available at: http://www.m-w.com/cgi-bin/dictionary

Mitrani, M., & Swan, K. (1990, March). Placing computer use in context. Paper presented at the International Conference on Technology and Education, Brussels, Belgium. (ERIC Document Reproduction Service No. ED 329 760).

Norman D. A. (1997, draft). Taming Technology. To be published in late 1998. Available from: "http://cogsci.ucsd.edu/~norman/DNMss/TamingTech.html"

Perkins, D. N. & Solomon, G. (1984) Transfer and Teaching Critical Thinking. #17, Page 285, Thinking, the second international conference. Lawrence Erlbaum Associates, Inc. (1987).

Resnick, M. (1996). Beyond the Centralized Mindset. Journal of the Learning Sciences, 5(1) 1-22.

Resnick, M. (1997, October). Building a learning society. Wired, 5.03 pp. 136.

Terence (185-159 B. C.) Heautontimoroumenos, iv. 2, (From the translation of Henry Thomas Riley, B. A., with occasional corrections. The references are to the text of Umpfenbach.) Bohn's Classical Library.

TIPs Database: Lateral Thinking section, at: http://www.gwu.edu/~tip/debono.html

Yeaman, A. (1992). The seven myths of computerism. Tech Trends, 37, (2) 22-25.

Yoder, S. (1996). Demystifying the Internet. Learning and Leading with Technology, 24(2) 58-60.

Wilcox, S. (1996). Fostering Self-directed learning in the university setting. Studies in Higher Education, 21(2) 165-176.

URLS

Reggio Emila ERIC digest:
http://www.ed.gov/databases/ERIC_Digests/ed389474.html

A pretty good definition of what a geek is.
http://www.circus.com/~omni/geek.html

Geekspeak. An ATM to you is not the same ATM to a geek.
http://zaphod.cc.ttu.ee/vrainn/nox/geek.html