

Aisling Kelliher
Virginia Tech

Technology and the Arts: Educational Encounters of the Third Kind

My introduction to the world of programming as a child was through the line editor on an Atari 800XL, as I copied out Basic commands from computer magazines and waited for them to run... slowly. The fun quota for this not-exactly Minecraft activity was, perhaps unsurprisingly, reached rather quickly, and it took another 15 odd years or so before I attempted to wrestle with bending machines to my will, this time through the object-oriented joy of Java. Over time, I slowly began to enjoy the creative possibilities of communicating with computers at varying levels of programmatic abstraction, and became aware of the tremendous numbers of people doing quite extraordinary, beautiful, and mind-blowing work in concert with programmable machines.

Discovering the technology-arts community and the possibilities of creative coding provided me with a “career re-orienting” encounter for which I am very grateful. Having neither succeeded as a young hobbyist programmer nor pursued a formal computer science education, I am fortunate to have noodled along sufficiently as a design/art/technology hybrid to find a professional and creative outlet and home for my skills and interests.

In looking around today, I can certainly identify less circuitous routes toward achieving an integrated tech-arts education and career path preparation. However, some familiar arguments and scenarios regarding arts and science education continue to appear—albeit in different guises.

Still with the Two Cultures?

In 1959, when C.P. Snow spoke of the “Two Cultures,” he was referring to the “traditional” arts/humanities and the “forward-thinking” sciences. He described the stark separation, misunderstandings, lost “creative chances,” and

social rigidity delineated between these increasingly separate and distinct forms of human inquiry.¹ Speaking within a Western context, he decried the “fanatical belief in educational specialization” making unbridgeable chasms between the sciences and the nonsciences. Although the notion of the Two Cultures has attracted support, derision, and extension in the last half century (for example, popular or streetwise science has been considered as a “third” culture²), the ghosts of Snow’s concerns still linger somewhat in the current and ongoing battle between the science and arts proxies of STEM versus STEAM.

The momentum in favor of STEM (Science, Technology, Engineering, and Math) learning is propelled in part by economic fears about faltering innovation and a reduction in technical workforce skills, particularly in Europe and the US in the face of growing foreign competition. Encouraging and enticing students into focused STEM education and career paths is widely seen as a solid and practical approach to addressing this issue and is favored and supported by many politicians, industry analysts, and funding agencies.

When pressed to include the “A-related” disciplines in this educational model, some proponents insist that the arts are already covered within STEM, while others believe that adding another leg to the chair only serves as a distraction within an arena already struggling with student engagement.³ However, advocates for a more inclusive form of learning describe adding the “A” as being innovative in directing students through hands-on STEM-related projects that are imbued with opportunities for creative thinking, risk taking, and the cultivation of necessary design skills (see, for example, the STEAM initiative at the Rhode Island School of Design; www.risd.edu/about/STEM_to_STEAM).

Although there are arts practitioners who caution about simply throwing the arts into the mix in a subservient or “service-based” role, careful arguments are also made as to how a STEAM formulation can serve as a hook or “on-ramp” for communities of students not typically oriented toward STEM careers or education. Positing a larger umbrella term as a means to increase and sustain diversity is supported in the literature, where art- and design-based technology experiences (from elementary school on up) prove helpful in expanding the pool of participants in technical fields.^{4,5}

To Code or Not to Code

A possible bridging argument that has found favor to varying degrees within both STEM and STEAM camps is the claim that everyone should learn how to code. Championed by NBA players (Chris Bosch), former New York city mayors (Michael Bloomberg), rock stars (will.i.am) and giant corporate geeks alike (Bill Gates and Mark Zuckerberg), organizations such as code.org are pushing the idea that a populace that knows how to program will be able to think systematically, break down problems, and become “wizards” of the future with their “superpowers” (see for yourself at www.youtube.com/watch?v=STRPsW6IY8k).

One could posit that marketing this flavor of computer science as a linear walk from hacking with JavaScript to Internet riches and fame is both self-serving to the affiliated corporate partners and overly narrow in conceptualizing the scope and degree to which programming as a skill and practice could manifest itself in people’s lives. Indeed, perhaps it’s all just a colossal waste of time, because surely the era of intentional programming⁶ is just around the corner—or, more bluntly, the imminent time in which “we won’t program computers” but rather will “train them like dogs.”⁷

Speaking pragmatically, I still think we have quite a ways to go before the AI/machine learning/neural net combination shifts engineering to some form of metababysitting. In the meantime, we can continue to tease out the how and why of the variety of ways in which programming, and indeed, technology more broadly, is encountered, learned, and taught within and across the so-called “two cultures.”

Encountering Technology-Arts Education

In the last 20 years, there has been strong growth in the number of integrative tech-arts

certificates, minors, majors, and graduate degree programs offered in universities and colleges across the globe. Foundational initiatives include studio research/atelier labs—such as the MIT Media Lab, New York University’s Interactive Telecommunications Program (NYU’s ITP), and Carnegie Mellon University’s Entertainment Technology Center. Other contemporary exemplars include Interaction Ivrea (now defunct), Umea’s Institute of Design, and the cross-institutional Stanford d.school. During this time, the evolution of new fields of inquiry, such as human-computer interaction, computational storytelling, information visualization, and interaction design, have helped drive the development of the creative technologies industry. Furthermore, the accessibility of cheap and powerful computers and the growth of open source creative tools have introduced new avenues for hybrid tech-arts practitioners.

Integrating Tools and Techniques

Processing, OpenFrameworks, Cinder, and Pure Data are just some of the languages, environments, and libraries taught in innovative and creative tech-arts courses, many of which are also exploring meaningful ways to integrate contemporary algorithmic innovations into expressive and provocative projects. For example, there’s a rich history, tracing back to the early 1970s, of interactive artworks and projects incorporating computer vision techniques for detecting motion, presence, gaze, or facial expressions in participatory installations.⁸

Although computer vision has been a familiar part of tech-arts curricula and syllabi for many years, more recently we see the addition of machine learning as a fundamental component in media arts training. At NYU’s ITP, Heather Dewey-Hagborg, Patrick Hebron, and Gene Kogan have developed and taught a series of courses introducing machine learning to creative practitioners. In a recent article, Kogan outlined some of the challenges (such as the need for big and sometimes hard-to-find datasets and the need for fast machines) and opportunities (such as the wealth of online tutorials and guides and generalizability of machine-learning applications) encountered in teaching and learning machine-learning techniques for artists and designers.⁹ Furthermore, Kogan added an as-yet cautionary note with regard to the use of deep-learning libraries in interdisciplinary machine-learning courses, highlighting the lack of “high-level abstraction found in creative



Figure 1. An exercise from the “Copy. Connect. Remap. Repeat” course at the Bergen Academy of Art and Design—remaking a glitched digital image into a quilt by Johanne: (a) the whole quilt and (b) part of the quilt in more detail. (Source: Ben Dalton and Amber Frid-Jimenez, Bergen Academy of Art & Design; used with permission.)

coding libraries,” and the deep and focused expertise required for effective and efficient debugging.

While the next few years will likely yield solutions to this deep dive issue, opportunities also exist to expand the toe-dipping experiences encouraging new types of learners.

Hooks and On-Ramps

One such example is the “Copy. Connect. Remap. Repeat” course taught back in 2011 at the Bergen Academy of Art and Design by Amber Frid-Jimenez and Ben Dalton (see <http://globe.khib.no/remaprepeat>). Using the course title constructs to introduce students to computational principles, the instructors created a rich variety of novel pedagogical frameworks, including teaching using “glitch” and “reading code as tourists.” Over email, Dalton recently explained the motivation for using glitch as emerging “from the way that a glitch reveals the mechanics of a file format, data stream or process.”

Inspired by the works revealed in Iman Moradi’s book *Glitch: Designing Imperfection* (Mark Batty Publisher, 2009), the instructors created a series of glitching exercises (see Figure 1), enabling students to examine concepts of compression “by breaking various file formats and other people’s code.” Another framing mechanism

used in the course to introduce novices to more complex programs, and indeed, entire software ecosystems (such as GIS mapping software), involved treating programming languages as “if you were a tourist experimenting with a few key phrases, rather than learning a language completely from scratch.” This high-level, almost disposable approach to encountering technical languages provides a compelling hook enabling novices to at least get started using a commonplace metaphor.

Alternative Venues for Learning Tech-Arts

There are also a variety of alternative online and bricks-and-mortar venues that offer novel, complementary, or continuing education opportunities to develop skills and expertise relevant to the field of technology-arts. As noted earlier, nonprofits such as code.org provide online programs and resources for elementary ages up, aimed at promoting coding as a valuable form of 21st century literacy.

The recently launched Kadenze website (www.kadenze.com) partners with academic institutions around the world to produce and distribute tech-arts courses from recognized leaders and innovators. With a combination of free and membership-required offerings, Kadenze aims to be, in the emailed words of cofounder Perry Cook, “a truly arts/media friendly learning

management system,” where courses are prepared and taught by “the rock stars who created the language or system or technique... who are regarded as the very best teacher of a given subject, and rock-star artists who use these tools as guest faculty.” This combination of theory, technique, and practice, together with automated grading and feedback tools, aim to shape Kadenze into a lively and engaging environment for tech-arts learning.

The beautifully named School for Poetic Computation in New York provides a hybrid school/residency/research group-based experience (<http://sfpc.io>), where “students and faculty work closely to explore the intersections of code, design, hardware and theory—focusing especially on artistic intervention.” Launched in 2013, the *NY Times* profiled the school by asking readers to “imagine the Robin Williams character from the movie ‘Dead Poets Society’ teaching Objective C instead of ‘O Captain, My Captain.’”¹⁰ The artist-run school offers a 10-week program of integrated tech-art courses three times a year, exploring concepts including physical computing, language design, and building the commons.

Beyond the STEM/STEAM Menu

The notion of a 21st century liberal arts education without technology should really be as unthinkable as a 20th century version is without the arts. Many of the examples I’ve discussed posit the importance of technical proficiencies embedded within an arts context; however, the reverse also holds equally true. Preparing technologists of all stripes to deal with the complexity of today’s world requires a thorough understanding of the full scope of human experience.


To achieve this, our students need to do more than simply add on a discrete “A” here or an “E” or an “S” over there. Instead, they need to develop broad, encompassing problem-solving abilities through integrative humanistic technology training. This generalist approach provides a strong foundational base that can also directly connect in parallel, or down the line, to fields of specialization and focused expertise. Creating and supporting educational trajectories imbued with authentic encounters with diverse learning cultures can ultimately provide value to individuals, organizations, and institutions invested in cultivating future generations of creators and innovators.

MM

References

1. C.P. Snow, *The Two Cultures*, Cambridge University Press, 1959.
2. J. Brockman, *The Third Culture: Beyond the Scientific Revolution*, Simon & Schuster, 1995.
3. G. May, “STEM not STEAM,” *Inside Higher Education*, 30 Mar. 2016, www.insidehighered.com/views/2016/03/30/essay-criticizes-idea-adding-arts-push-stem-education.
4. K. DesPortes, M. Spells, and B. DiSalvo, “The MoveLab: Developing Congruence Between Students’ Self-Concepts and Computing,” *Proc. 47th ACM Technical Symp. Computing Science Education (SIGCSE)*, 2016, pp. 267–272.
5. I. Greenberg, D. Kumar, and D. Xu, “Creative Coding and Visual Portfolios for CS1,” *Proc. 43rd ACM Technical Symp. Computer Science Education (SIGCSE)*, 2012, pp. 247–252.
6. C. Simonyi, *The Death of Computer Languages, The Birth of Intentional Programming*, tech. report MSR-TR-95-52, Microsoft, 1995.
7. J. Tanz, “Soon We Won’t Program Computers. We’ll Train them Like Dogs,” *Wired*, 17 May 2016; www.wired.com/2016/05/the-end-of-code.
8. G. Levin, “Computer Vision for Artists and Designers: Pedagogic Tools and Techniques for Novice Programmers,” *J. Artificial Intelligence and Soc.*, vol. 20, no. 4, 2006, pp. 462–482.
9. G. Kogan, “Machine Learning for Artists,” *Medium*, 3 Jan. 2016; <https://medium.com/@genekogan/machine-learning-for-artists-e93d20fdb097#rk07a0a0z>.
10. A. O’Leary, “Code to Joy: The School for Poetic Computation Opens,” *The New York Times*, 12 Aug. 2013; <http://bits.blogs.nytimes.com/2013/08/12/code-to-joy-the-school-for-poetic-computation-opens>.

Aisling Kelliher is an associate professor in the Department of Computer Science and a Catalyst Fellow at the Institute for Creativity, Arts, and Technology at Virginia Tech. Contact her at aislingk@vt.edu.

 Selected CS articles and columns are also available for free at <http://ComputingNow.computer.org>.