Of Art and Algorithms: Rethinking Technology & Creativity in the 21st Century

By Punya Mishra, Michigan State University, Aman Yadav, Purdue University, & the Deep-Play Research Group, Michigan State University

Science is what we understand well enough to explain to a computer; art is everything else
—Donald E. Knuth (1996, p.vii)

What good are computers? They can only give you answers
—Pablo Picasso (Fifield, 1982, p. 145).

Numbers, and how we think about them, played a very important role in the recently concluded presidential election. And we are not speaking of the 47% vilified by one group, or the 1% by the other. Though these particular numbers got a lot of the media attention, our focus is on a special kind of approach towards working with numbers, what we call *computational thinking*. Computational thinking, as exemplified by the data-driven election forecasting from people like Nate Silver (the statistician whose FiveThirtyEight.com blog became an online phenomenon of the election), and Sam Wang (a Princeton neuroscientist and a "spare time" election forecaster), up-ended the "seat-of-the-pants", gut-feeling driven predictions made by most established political prognosticators and pundits. After a long history of relying on polls or talking heads to predict election results, was undermined by a group of self-described "geeks" and number crunchers. Armed with computational simulations, data sets, and algorithms, these data crunchers were unfailingly accurate in their predictions. Political winners aside, many journalists and cultural observers noted afterwards, that the biggest winner might have been mathematics!

Computational thinking was unquestionably influential in the campaign strategies themselves, particularly the manner in which the Obama campaign used data to micro-target voters. Building on work first conducted during Rick Perry's gubernatorial campaign in Texas, the Obama election machine sliced and diced polling data, voter information, and hundreds of other variables to pinpoint and customize voter registration, campaign mailings, television advertising, and door-to-door drives integrated with social media campaigns. These data and analysis driven considerations helped to assemble a remarkable get out the vote effort in the swing states. (See Sasha Issenberg's (2012) book *Victory Lab* for more details of how computational thinking has become a critical part of elections today.) This change in thinking about how campaigns operate parallels a similar shift toward statistical and computational thinking that occurred in baseball a decade or so ago (as documented by Michael Lewis' in his book *Money Ball*—which was also the basis of the movie with the same name).

These are popularized and timely examples of a powerful idea—that of *Computational Thinking*. In this paper we seek to describe computational thinking, its growing influence on other disciplines, and focus on its relationship to creativity (see also Mishra & the Deep-Play Research Group, 2012; Mishra, Henriksen & The Deep-Play Research Group; 2012).

Introducing Computational Thinking

Computational Thinking is aided significantly by the ubiquitous availability of the digital computer, cheaper hardware, as well as better software for data analysis. Though computational thinking draws heavily from computer science; it must be seen as being distinct from it. As Barr and Stephenson (2011) stated, the field of computer science involves many factors, such as: programming; hardware design; networks; graphics; databases

and information retrieval; computer security; software design; programming languages and paradigms; logic; translation between levels of abstraction; artificial intelligence; the limits of computations (what computers cannot do); applications in information technology and information systems; and social issues (Internet security, privacy, intellectual property, etc.).

Computational thinking, though drawing on fundamental computer science concepts is distinct from it. Computational Thinking focuses on problem solving through "seeking algorithmic approaches to problem domains; a readiness to move between differing levels of abstraction and representation; familiarity with decomposition; separation of concerns; and modularity" (Barr & Stephenson, 2011, p. 49-50). In fact some even argue that computational thinking is an approach that does not necessarily need programming of computers, but rather is an approach to problem solving that uses strategies such as algorithms, abstraction and debugging (Yadav, Zhou, Mayfield, Hambrusch, & Korb, 2011). Along the same lines, Bundy (2007) argued that the ability to think computationally is essential to conceptual understanding in every discipline, through the processes of problem solving and algorithmic thinking.

The fact that computational thinking is not just limited to computer science, but is also applicable across disciplines, has been used to argue that computational thinking can be regarded as a fundamental, basic skill that all children need to develop (Wing, 2006). As Wing wrote, "To reading, writing, and arithmetic, we should add computational thinking to every child's analytical ability" (p. 33). The National Council for Research (NRC) further echoed this assertion, in suggesting that computational thinking is a cognitive skill which is of use to everybody (NRC, 2010). The NRC report also highlighted the connections between computational thinking and disciplinary knowledge. It suggested that, "(1) that students can learn thinking strategies such as computational thinking as they study a discipline, (2) that teachers and curricula can model these strategies for students, and (3) that appropriate guidance can enable students to learn to use these strategies independently" (p. 62).

The pervasiveness of computational thinking underscores the importance of exposing students to these ways of thinking and their application in the real world. Hemmendinger (2010) argued that the goal of teaching computational thinking is not for everyone to think like a computer scientist, but "it is to teach them how to think like an economist, a physicist, an artist, and to understand how to use computation to solve their problems, to create, and to discover new questions that can fruitfully be explored" (p. 6).

Computational Thinking and Creativity

Recently, computer science educators have underscored the importance of creativity by incorporating creative processes as one of the big ideas of computer science (The College Board, 2012). The College Board has argued that computing is a *creative* human activity that facilitates exploration and creation of knowledge, enables innovation, and allows individuals to create personally meaningful artifacts. As the *Computer Science Teacher Association* suggested that computational thinking uses, "a set of concepts, such as abstraction, recursion, and iteration to process and analyze data, and to create real and virtual artifacts" (Barr & Stephenson, 2010, p. 51). Computational thinking also draws upon a set of trans-disciplinary skills that we have argued (Mishra, Koehler & Henriksen, 2011) are essential for creativity.

Hence, computational thinking can foster creativity by allowing students to not only be consumers of technology, but also build tools that can have significant impact on society. This emphasis on creativity can be seen in a new computer science course called *CS Principles*, currently being developed and piloted by the College Board. The core argument for including creativity in this mix is that computing not only extends traditional forms of human expression, but also allows the creation of new forms of expression (The College Board, 2012).

This idea of new forms of expression is worth exploring further. Though much of the emphasis of computational thinking has been on what has been called "big data" (the combination of today's access to immense amounts of human behavior data, alongside the capabilities of new computational tools for real-time analysis and representation) there is more to computational thinking than that. We argue that computational thinking goes beyond typical human computer interactions, in which humans initiate the actions that are then computed by the machine. The new forms of expression that are emerging today have significant implications for how we engage and interact with machines. In this "brave new world" machines take on a role in the creative process itself – partners to humans as it were. Seeing the computer in this light has significant implications for how we educate our students for the jobs of the future. Looking forward, we see that human initiative must blend with the capabilities of software programs in ways that have not been possible before.

We are now seeing the rise of a generation of technology enthusiasts and creators who are both computationally savvy *and* deeply knowledgeable about their field of interest. Nate Silver is a prime example of someone who brings both a deep understanding of computation (numbers and statistical techniques) *and* of the subject matter knowledge of the political process (biases that polls have, historical knowledge of voting patterns, and so on). It is this combination that makes him successful.

We will end with two examples of how computational thinking, in combination with deep knowledge of a discipline, can lead to creative solutions that could not have been possible before. In this perspective the creative output is not determined either by only the individual or the technology, but rather through a "partnership" between the two.

New forms of creativity, two examples

For our first example we focus on David Cope, a noted musician and composer, who has been experimenting for years with computer programs that create original music (Blitstein, 2010). At the beginning he created a computer program EMI (Experiments in Musical Intelligence also known as Emmy) that composed music in the style of different historical composers. The music created by Emmy was so impressive that scholars of music often failed to identify them as being computer-generated. This of course raised some fundamental questions about creativity, such as—who *was* the author of the pieces thus generated?

David Cope however, was not troubled by these concerns, since in his opinion Emmy was merely a tool. Everything Emmy created, she created because of software *he* devised. He argued that if he had infinite time, he could have written 5,000 Bach-style chorales. The program just did it much faster. As he said, "All the computer is just an extension of me. They're nothing but wonderfully organized shovels. I wouldn't give credit to the shovel for digging the hole. Would you?" (All the quotes from Cope in this article are from Blitstein, 2010.)

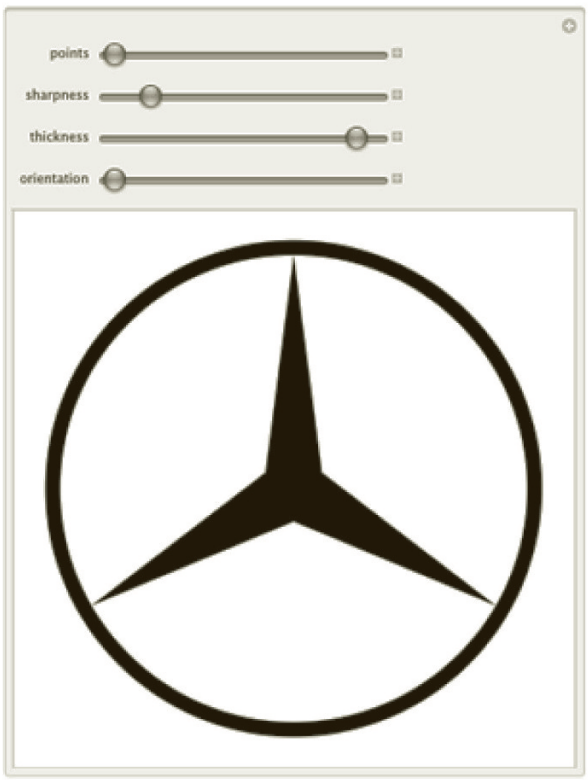
But as it turns out, Cope eventually got tired of his first program, successful though it was, and he began experimenting with a different kind of virtual composer. This time he wanted to build something "with its own personality." This program would be more of a collaborator. Underlying the computer program is what he describes as an "association network—certain musical statements and relationships between notes are weighted as 'good,' others as 'bad.' This new program (called Emily Howell) would not print out a full score (as the previous program Emmy did) but rather *engaged in a conversation* with Cope through the keyboard and the mouse. He would ask it what he called "a musical question," a piece of a composition or musical phrase. The program would then respond with a composition, which then Cope would either choose to accept or not. He would at times tweak the composition created by the software and feed it back to the machine, and ask it to generate a newer piece. Eventually, the exchange would result in a composition that was acceptable to him. Cope compares this process to a sculptor who chops raw shapes out of a block of marble before he teases out the details. Using quick-and-dirty programs as an extension of his brain has made him extraordinarily prolific (he has even published recordings with Emily Howell as a co-composer). In our context, it also makes this an excellent example of the creativity that emerges from combining computational thinking with deep knowledge of a discipline.

Our second example comes from the field of visual design. Christopher Carlson, handles technical communication and strategy for Wolfram, the creators of Mathematica. He also has an interest in graphic design and architecture, and describes one of his "enduring passions" as "exploring graphic design with programmatic and generative systems." In other words, he explores the world of design using computational tools. In an article titled "Exploring logo designs with Mathematica," Christopher shows how one can mathematically develop variations on commercial logo designs by the systematic tweaking of various parameters (Carlson, 2009a). He works with corporate logos because of certain inherent features they all share. As he says, logos, "often distill a single idea into simplified geometric form that is straightforward to parameterize in Mathematica. Once a logo is in Mathematica, exploring its parameter space quickly leads to the discovery of new graphic phenomena, emergent forms, unexpected relationships, and burgeoning lines of inquiry." (All the quotes are from Carlson, 2009a, unless noted otherwise.) One of the explorations that Christopher delved into was with the Mercedes-Benz logo (Figure 1). He started by writing some Mathematica code to re-create the logo, with (in his words) "some obvious parameters controlling the number of points on the star, the sharpness of the star's points, the thickness of the outer circle, and the orientation of the star" (Figure 2). He then set about tweaking the parameters—and the program began to spit out lots and lots of variations. The range of possibilities that emerged is impressive, many of which can be the starting point for the design of other corporate logos (see Figure 3 for a sample of the variations that emerged). Christopher expanded his range to include other logos and then went forward and started combining them together, leading to visual solutions that as he describes them, led him to a "world inhabited by sea creatures and whorled growths, forms with a pleasing balance of rigid symmetry and organic irregularity" (Carlson, 2009b). Interestingly Christopher's description of his exploration has a conversational quality—quite akin to the dialogue between David Cope and his software program.

Figure 1

What is interesting in this example is the "partnership" that emerges between a software program (in this Mathematica) and the designer. The computer does what it is good at, creating, almost instantaneously, a huge number of variations. This does not mean that the designer does not have a role in the creative process. We argue that the software is not a replacement of the creator, but rather is a great tool to automatically generate a range of variations on a theme, based on a set of parameters from the de-

```
In[1]:= Manipulate[
 Graphics[{{Disk[], White, Disk[{0, 0}, ir]},
   Polygon[Table[
     With[{a = (or + 90 + i 180 / n) °},
      If[EvenQ[i], 1, sh] ir {Cos[a], Sin[a]}], {i, 0, 2 n - 1}]
    ]
  }],
 {{n, 3, "points"}, 3, 24, 1},
 {{sh, .14, "sharpness"}, 0, 1},
 {{ir, .94, "thickness"}, 0, 1},
 {{or, 0, "orientation"}, 0, 360 / n}
]
```



Out[1]=

Figure 2

signer. As the research on creativity shows, this phase of open ideation is a critical part of the creative process. However, this does not mean that humans are out of the picture totally. In fact, the designer has two very critical roles to play here.

The first important task the designer accomplishes is in choosing which parameters to include in writing the original code. That is not at all obvious and needs careful thought. In the case of the logo, Carlson had to decide, whether or not he needed to include the thickness of the outer circle as a parameter. What about color?

In the case of the program written by David Cope, the parameters of the musical choices, and options that the program could "play with," were put into place by Cope himself. This is important since there are volumes of options, and the choices are non-trivial. Setting and tweaking different parameters will lead to very different solutions. These parameters can only be determined and set by a person who combines computational thinking with deep knowledge of a domain (music in the case of David Cope, and graphic design in the case of Chris Carlson).

The second place where humans step in, is after all these variations have been created (and the computer creates LOTS of them). In graphic design for example, human insight is needed for selecting one of the solutions for its appropriateness to the organization for whom the logo is being designed, finding the right aesthetic look, understanding the context of use (on a website, on a car dashboard, in a neon sign?), and so on. In David Cope's case the role of the human is even more complicated than that, since the composer can go back and input changes into the first piece developed by the program and then feed it back into the program, in a continual and ever evolving feedback loop.

In each of these cases, *human creativity is augmented by computational thinking, in particular the automation of problem solving and algorithmic thinking.* Computational Thinking allows each individual to become more creative and productive. None of this is possible, however, without the designer or composer having computational thinking skills. For instance, programming is not something we typically teach in design school. But as is clear, a designer with a good visual sense AND a knowledge of programming and mathematics is going to be much more efficient and generative (in terms of total ideas) than one with just the former. Similarly, a programmer without a good understanding of the domain is less likely to creative. The human element is still present, just refracted through the lens of the computational tool. Taking advantage of these new tools requires a new set of skills—*computational thinking plus.*

In conclusion

There were three main points we were making in this article. First, for a range of reasons computational thinking needs to become a key part of our intellectual curriculum. Second, the partnership of deep human content knowledge and technology can lead to deeper and more profound creative insights. Third, and finally, the addition of computational thinking to the creative process does not in any way

diminish the role of the human. In fact, human intuition and agency play a key role in this process.

At a recent MIT conference on big data, one of the leaders of the conference was asked what makes a good data scientist. She noted that while knowledge of computer science and mathematics was clearly necessary – just as important were imaginative capacities like innovative thinking and a "deep, wide ranging curiosity". As the New York Times (Lohr, 2012) article reporting on the conference wrote, speaking of the consensus emerging among the participants:

> Listening to the data is important… but so is experience and intuition. After all, what is intuition at its best but large amounts of data of all kinds filtered through a human brain rather than a math model?

Nate Silver (as an example of one of the creative people who have managed to combine computational thinking with a deep understanding of politics, to become one of the best election forecasters in the business) would agree. In an interview with Fast Company he said that one of the important tasks is to develop, "an intuitive sense—one that is honed and refined through experience—for what's meaningful" (Barr, 2012).

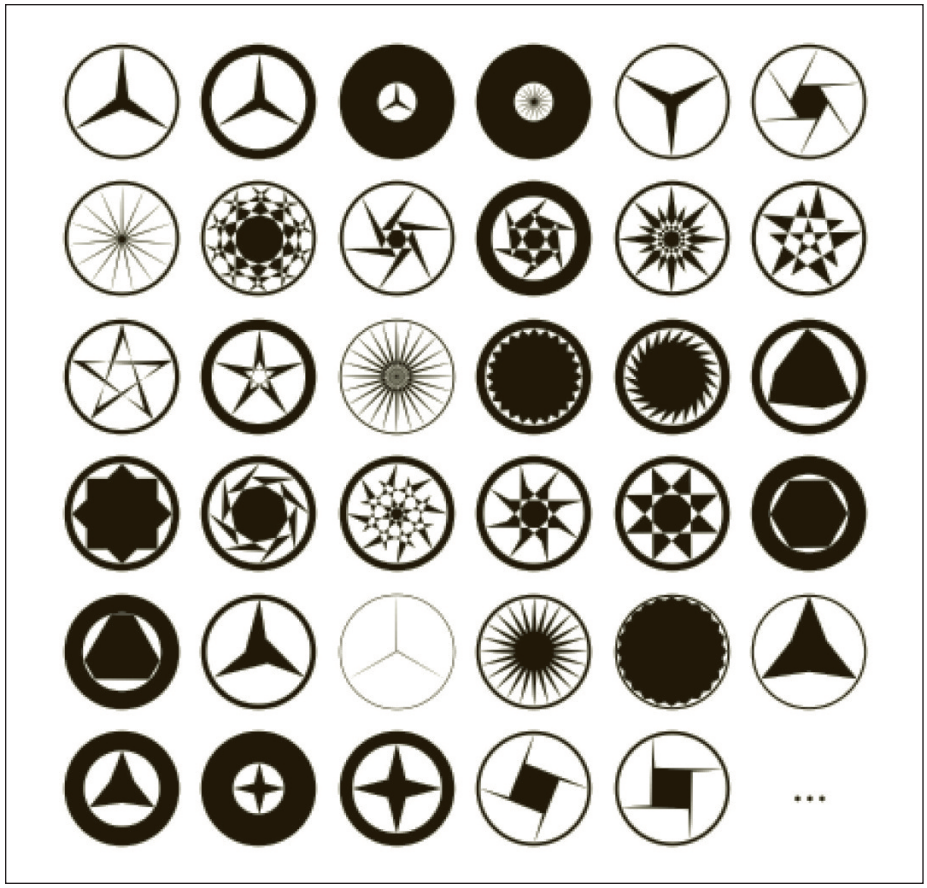


Figure 3

References

Baer, D. (2012). FiveThirtyEight's Nate Silver explains why we suck at predictions (and how to improve). Fast Company. Retrieved from http://www.fastcompany.com/3001794/fivethirtyeights-nate-silver-explains-why-we-suck-predictions-and-how-improve

Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: What is involved and what is the role of the computer science education community? *ACM Inroads, 2*(1), 48-54.

Blitstein, R. (2010, February 22). Triumph of the cyborg composer. Pacific Standard. Retrieved from http://www.psmag.com/culture-society/triumph-of-the-cyborg-composer-8507/

Bundy, A. (2007). Computational thinking is pervasive. *Journal of Scientific and Practical Computing, 1*(2), 67-69.

Carlson, C. (2009a). Exploring logo designs with *Mathematica.* Retrieved from http://blog.wolfram.com/2009/02/26/exploring-logo-designs-with-mathematica/

Carlson, C. (2009b). Hybrid logos and a fortunate mistake. Retrieved from http://blog.wolfram.com/2009/04/22/hybrid-logos-and-a-fortunate-mistake/

Fifield, W. (1982). In search of genius. New York, NY: William Morrow & Co.

Hemmendinger, D. (2010). A please for modesty. *ACM Inroads, 1*(2), 4-7.

Knuth, D. E. (1996). Foreword. In M. Petkovsek, H. S. Wilf, & D. Zeilberger (Eds). *A=B* (pp. vii). New York: A.K. Peters Ltd.

Issenberg, S. (2012). *Victory Lab*. New York, NY: Crown Publishers.

Lohr, S. (2012, December 29). Sure, Big Data Is Great, But So Is intiution. *The New York Times*. Retrieved from http://www.nytimes.com/

Mishra, P., & The Deep-Play Research Group (2012). Rethinking technology & creativity in the 21st century: Crayons are the future. *TechTrends, 56(*5), 13-16.

Mishra, P., Henriksen, D. & The Deep-Play Research Group (2012). On being (in) disciplined. *TechTrends 56(*6), 18-21.

Mishra, P., Koehler, M. J., & Henriksen, D. (2011). The seven trans-disciplinary habits of mind: Extending the TPACK framework towards 21st century learning. *Educaitonal Technology, 51*(2), 22-28.

NRC (2010). *Report of a workshop on the scope and nature of computational thinking*. Washington, DC: The National Academies Press

The College Board (2012). Computational thinking practices and big ideas, key concepts, and supporting concepts. Retrieved from http://www.csprinciples.org/home/about-the-project.

Wing, J. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33-35.

Yadav, A., Zhou, N., Mayfield, C., Hambrusch, S., & Korb, J. T. (2011). *Introducing computational thinking in education courses.* In Proceedings of ACM Special Interest Group on Computer Science Education, Dallas, TX.

Note:

The Deep-Play Research group at the College of Education at Michigan State University includes: Punya Mishra, Danah Henriksen, Kristen Kereluik, Laura Terry, Chris Fahnoe and Colin Terry. Address all communication to Punya Mishra, email: punya@msu.edu.