ISBN

978-1-138-84430-8

Author

Wardrip-Fruin, Noah

Publication Date

2018

Peer reviewed

46

EXPRESSIVE PROCESSING

Interpretation and Creation

Noah Wardrip-Fruin

Background: Software Studies

There are many things that a computer is not. It is neither an interactive movie projector, nor an expensive typewriter, nor a giant encyclopedia. Instead, it is a machine for running software, which can enact processes, access data, communicate across networks, and, as a result, act like a movie projector, typewriter, encyclopedia, and many other things.

Outside engineering and mathematics, most studies of software have considered software in terms of what it acts like and how that behavior is experienced outside the system. But some authors have consistently written about software as software. This approach includes considering software's internal operations (as my work does), examining its constituent elements (e.g., the different levels, modules, and even lines of code at work), studying its context and material traces of production (e.g., how the workings of money, labor, technology, and the market can be traced through whitepapers, specification documents, version control system archives, beta tests, patches, and so on), observing the transformations of work and its results (from celebrated cases such as architecture to the everyday ordering and movement of auto parts), and, as the foregoing implies, a broadening of the types of software considered worthy of study (not just media software, but also design software, logistics software, databases, office tools, and so on).

These investigations form a part of the larger field of "software studies"—a field that had its start long before scholars began using the term. Often this field is positioned as part of digital humanities, and we can see why this might be the case. First, much of software studies involves the interpretation and critique of digital objects and culture that are common in digital humanities. Other software studies work, such as Warren Sack's forthcoming *The Software Arts*, involves making novel software inspired by (or enacting) a critical interpretation of software's specifics. This method fits relatively comfortably with the parts of digital humanities that engage making through scholarly communication. Third, there is another set of software studies work—such as the "cultural analytics" collaborations of Lev Manovich, Jeremy Douglass, and William Huber—that continues the quantitative humanities computing tradition (now sometimes positioned as the "big data" of the humanities) (Manovich, Douglass, & Huber 2011).

But if I examine my own route into software studies, the picture looks quite different. Some books that influenced me—such as Paul Edwards's *The Closed World* (1997), a history that fits within humanistic study of software objects and institutions—could be positioned within digital humanities. But other books that influenced me cannot. Consider Lucy Suchman's *Plans and Situated Actions* (1987). This is a work of social science, applying ethnomethodological concepts to understanding software and how we interact with it. Not only does this move into the social sciences take us outside digital humanities, but Suchman's book is also explicitly positioned as a piece of thinking about how software is *designed*, and should be designed. Such research is software studies as part of a practice of making software (rather than making software as part of a practice of software studies). Or consider Phil Agre's *Computation and Human Experience* (1997). It uses humanistic approaches to expose conceptual problems in approaches to artificial intelligence (AI) and identify alternative conceptualizations. Then it describes actual architectures—that is, computer science research—that result from this inquiry. In other words, in Agre's book, software studies is a method of computer science. Finally, teaching software studies also leads me to believe that software studies is a route to thinking critically about software models—their histories, commitments, and potentials. Software studies is a route to "procedural literacy" (Mateas 2005).

Procedural literacy is one of the core aims of what I suspect is the first software studies book: Ted Nelson's *Computer Lib / Dream Machines* (1974). Published the year before the arrival of the first kit for assembling a personal computer, Nelson's book contains much that could be seen as purely informational, including introductions to different types of computers and programming languages. But it also contains much that is clearly the critical interpretation of software, connecting the technical level to the cultural one—ranging from discussing the "drill and practice" assumptions built into the Tutor programming language to exposing the surprisingly simple workings of *Eliza* and other systems used to market artificial intelligence ideas (Nelson 1974: DM27, DM14). Nelson's book introduced critical thinking about software as a basic part of understanding computing.

Given all this, my view of software studies includes all investigations that use humanities and/or social science methods and encompass the specific operations of particular software, for purposes ranging from understanding digital culture and software society to the development of procedural literacy and inspiring, guiding, and evaluating how novel software is made.

The phrase "software studies" was coined by Manovich in his widely read book *The Language of New Media* (2001). Manovich characterized software studies as a "turn to computer science," perhaps analogous to the "linguistic turn" of an earlier era. In his book, software studies operates analytically through programmability (rather than, say, signification). In 2003, Matthew Kirschenbaum offered his own expansion of Manovich's term, one influenced by Kirschenbaum's background in bibliography (the study of books as physical objects) and textual criticism (the reconstruction and representation of texts from multiple versions and witnesses). Kirschenbaum argued that in the field of software studies—as opposed to the rather loose, early "new media" field—"the deployment of critical terms like 'virtuality' must be balanced by a commitment to meticulous documentary research to recover and stabilize the material traces of new media" (2003: 150). Kirschenbaum's *Mechanisms* made good on this assertion in 2008, which also saw the publication of the field's first edited volume: *Software Studies: A Lexicon* (Fuller 2008). Soon after, the MIT Press established the first book series for software studies. All this helped create the conditions of possibility for my book, *Expressive Processing* (2009), which I see as an example of software studies.

Expressive Processing

I am drawn to software studies, in part because it brings together currents of work in computer science, humanities, social sciences, and the arts. In computer science, there is a long tradition of those who see their work on software in terms of culture—from Don Knuth's "literate programming" (1984) to Phil Agre's "critical technical practices" (1997). Similarly, in other fields there are those who have felt a need to engage the specifics of software in the course of their research and creation, in areas ranging from computer games to software art to multinational firm organization. My *Expressive Processing* focuses on software studies for computational media, especially games and fictions. I use the term "expressive processing" to point toward two important critical issues.

First, "expressive processing" encompasses the fact that the internal processes of computational media are designed artifacts; in this sense, they are like buildings or transportation systems. As with other designed mechanisms, processes can be seen in terms of their efficiency, their aesthetics, their points of failure, or their (lack of) suitability for particular purposes. Their design can be typical or unusual for their era and context. The parts and their arrangement may express kinship with, and points of divergence from, design movements and schools of thought. They can be progressively redesigned, repurposed, or used as foundations for new systems—by their original designers or others—all while retaining traces and characteristics from prior uses.

Second, unlike many other designed mechanisms, the processes of computational media operate on, and in terms of, elements and structures meaningful to humans. For example, a natural language processing system (for understanding or generating human language) expresses a miniature philosophy of language in its universe of interpretation or expression. When such a system is incorporated into a work of computational media such as an interactive fiction, its structures and operations are invoked whenever the work is experienced. This invocation selects, as it were, a particular constellation from among the system's universe of possibilities. In a natural language generation system, this invocation might be a particular sentence shown to an audience in the system output. From looking at the output sentence alone, it is not possible to see where the individual elements (e.g., words, phrases, sentence templates, or statistical language structures) once resided in the larger system. It is not possible to see how the movements of the model universe resulted in this constellation becoming possible—and becoming more apparent than other possible ones.

To put it another way, in the world of computational media, and perhaps especially for computational games and fictions, we have as much to learn by examining the model that drives the planetarium as by looking at a particular image of stars (or even the animation of their movement). This is because the model universes of these works are built of rules for character behavior, structures for virtual worlds, techniques for assembling human language, and so on. They express the meanings of their fictional worlds through the design of every structure, the arc of every internal movement, and the elegance or difficulty with which the elements interact with one another.

Trying to interpret a work of computational media by looking only at the output is like interpreting a model solar system by looking only at the planets. If the accuracy of Mars's surface texture is in question, then looking only at planets is fine. But it will not suffice if we want to know if the model embodies and carries out a Copernican theory—or, instead, places Earth at the center of its simulated solar system. Both types of theories could produce models that currently place the planets in appropriate locations, but examining the models' wires and gears will reveal critical differences, probably the most telling differences.

That is, the processes of computational media can themselves be examined for what is expressed through their selection, arrangement, and operation. As I have just discussed, a system operating on language (or other elements meaningful to humans) can be interpreted for what its design expresses. However, expressive processing also includes considering how the use of a particular process may express connection with a particular school of cognitive science or software engineering; or how the arrangement of processes in a system may express a very different set of priorities—or capabilities—from authorial descriptions of the system; or how understanding the operations of several systems may reveal previously unrecognized kinships (or disparities) between them. Recognizing such things can open up important new interpretations for a computational media system, with aesthetic, theoretical, and political consequences.

Interpreting: The Goldwater Machine

In *Expressive Processing*, one of the systems I interpret is the earliest explicitly political project in computational media, best known by its nickname: "The Goldwater Machine." I analyze it at the time of a 1965 paper by Robert Abelson and J. Douglass Carroll (Abelson & Carroll 1965), though it is a project that Abelson and his students had pursued since the late 1950s and would continue to pursue into the 1970s. At the point of their 1965 paper, the "ideology machine" consisted of an approach to belief structures and a number of operations that could be performed on such structures. Sample belief structures from the paper range from common Cold War views ("Russia controls Cuba's subversion of Latin America") to absurd statements ("Barry Goldwater believes in socialism") and also include simple facts ("Stevenson ran for President").

As these examples foreground, the Goldwater Machine is a system built in the midst of the Cold War. The Cuban Missile Crisis, President Kennedy's assassination, and the Gulf of Tonkin Resolution were all recent events. The world seemed polarized to many, and, within the United States, names such as Adlai Stevenson and Barry Goldwater did not simply indicate prominent politicians with occasionally differing philosophies. As the Republican nominee for President of the United States in 1964, Goldwater was an emblematic believer of the idea that the world's polarization was an inevitable result of a struggle between good and evil—a position that would be echoed by his ideological descendants (e.g., Ronald Reagan's "evil empire" and George W. Bush's "axis of evil"). On the other hand, Stevenson—who was the Democratic candidate for president in 1952—was emblematic of those with a more nuanced view of world affairs and a belief in the potential of international cooperation. He was publicly derided for this view and belief by those with more extreme views.

Interaction with the Goldwater Machine consisted of offering the assertion that a particular *source* (e.g., an individual) has made the claim that a *concept* (e.g., a particular nation) has the stated relation to a *predicate* (generally a verb and an object). For example, "Stevenson claims Cuba threatens Latin America." The statement was evaluated, and a response was generated. For example, if a "good" actor was asserted to be engaged in a bad action, then the system would attempt to deny the alleged fact or that the assertion was made. On the other hand, if a "bad" actor was asserted to be engaged in a good action, then the system would attempt to rationalize the alleged fact or the making of the assertion.

The Goldwater Machine system was presented as a structure and set of processes for modeling human ideology. Abelson and collaborators suggested it could then be populated with data (belief structures of the sorts mentioned above) to represent a particular ideology. If they had succeeded in building such a system, then the machine's processes would be

ideologically neutral; only the data would carry a particular position. In the absence of an expressive processing analysis of the system, there would be no reason to question this claimed underlying neutrality.

However, if one undertakes such an analysis, looking at the specifics of the system's operations, then they can reveal that the Goldwater Machine's ideology is also encoded *in its processes*. This encoding begins at the center of its operations, with the system motivated to dismiss any statement by a negatively viewed source, even a statement with which the system data agrees. It is also found in the design of the processes for denial and rationalization, which are predicated on a world divided into "good actors" and "bad actors." Further, in addition to being designed to operate in terms of good and bad, the primary processes for interaction are dedicated to finding routes to deny even the smallest positive action by the bad actors and seeking means to rationalize away even minimally negative actions by the good actors on the basis of paranoid fantasies, apologetic reinterpretations, and misdirections.

This is not a general model of ideology. It is a parody of one particular type of ideology, one that depends on fear to gain power. One can imagine Stevenson being critiqued exactly because his ideology operated by processes rather different from those encoded in the Goldwater Machine. As a result of this difference, it would be impossible to create a "Stevenson Machine" simply by providing a different set of concept-predicate pairs. Yet, without an examination of its processes, we would have had no reason to question its creators' claims to the opposite.

We live in a world in which software processes play increasingly consequential roles in our lives. We are generally asked to take at face value the claims of software creators about what their systems do and to treat those systems as ideologically neutral unless explicitly positioned otherwise. We cannot afford to do this. But to enact an alternative, we must have practice critically interpreting computational processes and must also develop knowledge of different software approaches in a context that makes them legible (rather than deliberately murky or obfuscated, as with high stakes areas such as voting machine systems and terrorist watchlist generators). I believe works of computational media provide some of the most legible and interesting examples for developing this ability, this kind of procedural literacy, as well as for understanding how such media influence our culture—and my approach to these examples is one aspect of what I mean by "expressive processing." The next section of this chapter explores another aspect.

Guiding: Prom Week

Expressive processing is not just an approach that helps us interpret computational media that already exist. It can help us guide the creation of new works. And we need approaches that can help us create, because applying existing criteria for guidance and evaluation (e.g., a technical focus on efficiency or maintainability, or an arts critique honed for fixed media) is not enough as we explore the new computational media experiences that matter.

Just as we are in a period in which software increasingly shapes our lives in a broad sense, we are also in a period of computational media growing in its scope and connection with our lives—from videogames to social media to smartphone apps. Yet we have a paucity of technical and design approaches for making computational media that engages with the things that matter most, from social relationships to ideology to storytelling. We need means to guide ourselves as we seek new uses for existing tools and, perhaps even more, as we seek to invent means to broaden what computational media can meaningfully address.

The three "effects" I outline in *Expressive Processing* might guide computational media creators. The first, the "*Eliza* effect," was already widely discussed before the book. It is what happens when a piece of software (like *Eliza/Doctor*) tries to fool audiences into thinking it is significantly more complex than it is. The results are interactions that can succeed based on initial expectation, but that suffer eventual breakdown (determined by the underlying system's shape) unless interaction is severely restricted. The second, which I call the "*Tale-Spin* effect," is what happens when the system's complex internal processes are never apparent to the audience. The result is that the systems contribute little to audience experience, usually only revealing their interest in moments of breakdown. The third, which I call the "*SimCity* effect," is what happens when software is designed to transition audiences (often through experimentation and feedback) from their initial expectations to an understanding of the underlying system's shape. This is an effective route for making novel processes the center of audience experiences, opening new possibilities for the issues and situations that computational media can address interactively.

I have attempted to use expressive processing as an approach to guide the creation of particular computational media works, and in this section I will briefly describe the attempt to do so with the experimental game *Prom Week* (2012), a social simulation set in the time leading up to an end-of-year dance at a U.S. high school. Each character in *Prom Week* has their own personal characteristics and desires, but these are defined against an elaborate set of rules and beliefs that determine "normal" behavior in the game's fictional high school. Players interact with *Prom Week* by selecting pairs among the characters, examining what actions they most wish to take with one another, and choosing which actions to attempt (see Figure 46.1). Each action is an attempt to change the social state in some way—one desired by the initiating character—which may succeed or fail. In either case, each attempt results in a short scene, which will play out differently depending on the specifics of the world history and character relationships, as well as a number of effects. These effects include changing the characters' relationships with each other, giving one or both characters temporary statuses that may change how future exchanges unfold (e.g., "embarrassed" or "angry at" a particular character), recording events in the ongoing history of the world, and triggering events based on patterns in the world state and character relationships. As a result, players can explore a wide range of possible social worlds, make important events happen in the lives of the characters, and fulfill (or ignore, or subvert) some of the character-specific goals presented at the start of each level—the last of which determines the ending of that character's story.

My interest in helping create *Prom Week* goes back to a question I pose on page 317 of *Expressive Processing*: Given how *The Sims* succeeds through the *SimCity* effect, "[c]an we find similar success with characters more complex than eight mood meters, and fictions more well formed than *The Sims*' implied progression through possessions and careers?" I was excited when, shortly after my arrival at UC Santa Cruz, there was a critical mass of student and faculty interest in pursuing a game of this sort. Josh McCoy was spearheading work on a new AI system that would allow characters to have motivations much more complex than Sims have—and also for characters to be caught up in systems of social expectation that, at that point, games had largely ignored (McCoy & Mateas 2009). Over the course of the project, the game evolved considerably, as we were guided by an expressive processing approach (for the final shape, see McCoy et al. 2013, 2014).

In particular, while we developed *Prom Week* and its underlying AI system, we regularly had meetings during which we stepped back and tried to ask two questions. First, is the design of the system expressing the ideas we have for *Prom Week*? Second, what are we building

Figure 46.1 In this image from *Prom Week*, a player has selected Oswald and is examining other characters from his perspective. Currently, the game shows what social actions Oswald most wants to initiate toward Doug, as well as high-level summaries of how Oswald and Doug feel about each other in terms of friendship, romance, and coolness.

into the system to communicate its design to players, so they can see their opportunities for creativity and play and get appropriate feedback? Obviously, these two questions are deeply connected, especially for a system, like a game, that only comes to mean through play. For example, much of the *Prom Week* system is designed to make the world function like a fictional high school when everything is "happening normally." The shy people do not get up the nerve to tell people how they really feel. Mean, popular people hold sway in the hallways. No one breaks out of their stereotypes.

In nonplayable media, we see the world working this way and then someone (the main character, usually) decides not to play along. We then see the world change. In *Prom Week*, the player not only gets to see how the world operates through depiction, but also experiments with it and gains a sense of its rules, and then decides which characters to nudge (or even use "social influence points" to nudge harder) and push out of their comfort zones—to make things happen that would not otherwise. This is tricky because a lot of what the system does, a lot of what its rules seem to express, is only part of what we want a play session to look like. We use the system's rules and initial scenario to not just express the way the world works but also show *the way the world does not have to be and help the player learn how to make a different world*. A player who understands the game can make radical shifts in the social landscape—one beta test player joyfully reported inverting the high school's entire popularity structure. It is that experience of changing the world, and of coming to understand some of the many ways the world could change, that is (hopefully) the core experience of *Prom Week*.

While through approaches such as playtesting (which we employed) we can gather some insight into whether an experience like *Prom Week* is succeeding, players only see the surface of a game. Playtesting might tell us that the underlying ideas we intend to express through system design are not coming through, but it will not help us press, critique, or even understand those underlying ideas. For instance, I think *Prom Week* is a better game for the critical discussions we had about issues such as how "popularity" should function in a game, and how popularity should be expressed to players. Perhaps counterintuitively, some of our most important decisions were about what we chose *not* to encode in our processes and how this absence was communicated to players. While *Prom Week*'s simulated school encodes many stereotypes and biases, it does not include, for example, racism and heteronormativity. The second of these proved particularly difficult for some players to understand, even though the first nontutorial level is designed to encourage players to discover it. (This level focuses on Oswald, a character who wants a date for the prom. His level is populated by a number of female characters with whom he has little connection, as well as by Nicholas, a "frenemy" with whom there is a lot of potential. But, despite the nudging of the level introduction, this potential proved difficult for some players to discover, or to accept.) In short, we learned that the power of initial audience expectation—which is key to both the *Eliza* and *SimCity* effects—must not be underestimated. Computational media that means to challenge, invert, or even leave out biases of the audience faces a particular challenge when pursuing the *SimCity* effect.

More broadly, I think an expressive processing approach could help many computational media creators give their processes the attention they deserve. We need to ask questions beyond, "Does this work?" and "Is this fun?" We need to ask questions such as, "If this part of the system were taken as an idea, then is it what you would want your audience to understand?" and "If so, then how does the system help them understand the idea through interaction?"

Further Reading

Wardrip-Fruin, N. (2009) *Expressive Processing: Digital Fictions, Computer Games, and Software Studies*, Cambridge, MA: MIT Press.

Wardrip-Fruin, N. (2011) "Digital Media Archaeology: Interpreting Computational Processes," in E. Huhtamo and J. Parikka (eds.) *Media Archaeology: Approaches, Applications, and Implications*, pp. 302–22.

Wardrip-Fruin, N. (2015) "We Can and Must Understand Computers NOW," in D. R. Dechow and D. C. Struppa (eds.) *Intertwingled: The Work and Influence of Ted Nelson*, New York, NY: Springer International Publishing, pp. 105–12.

References

Abelson, R. P. and J. D. Carroll (1965) "Computer Simulation of Individual Belief Systems," *The American Behavioral Scientist (pre-1986)* 8(9), 0–24.

Agre, P. (1997) *Computation and Human Experience*, Cambridge, UK: Cambridge University Press.

Edwards, P. N. (1997) *The Closed World: Computers and the Politics of Discourse in Cold War America*, Cambridge, MA: MIT Press.

Fuller, M. (2008) *Software Studies: A Lexicon*, Cambridge, MA: MIT Press.

Kirschenbaum, M. G. (2003) "Virtuality and VRML: Software Studies after Manovich," in M. Bousquet and K. Wills (eds.) *The Politics of Information*, Alt-X Press, pp. 149–53.

Kirschenbaum, M. G. (2008) *Mechanisms: New Media and the Forensic Imagination*, Cambridge, MA: MIT Press.

Knuth, D. E. (1984) "Literate Programming," *The Computer Journal* 27(2), 97–111.

Manovich, L. (2001) *The Language of New Media*, Cambridge, MA: MIT Press.

Manovich, L., J. Douglass, and W. Huber (2011) "Understanding Scanlation: How to Read One Million Fan-translated Manga Pages," *Image & Narrative* 12(1), 206–28.

Mateas, M. (2005) "Procedural Literacy: Educating the New Media Practitioner," *On the Horizon* 13(2), 101–11.

McCoy, J. and M. Mateas (2009) "The Computation of Self in Everyday Life: A Dramaturgical Approach for Socially Competent Agents," in *Papers from the AAAI Spring Symposium: Intelligent Narrative Technologies II*, pp. 75–82.

McCoy, J., M. Treanor, B. Samuel, A. A. Reed, M. Mateas, and N. Wardrip-Fruin (2012) "Prom Week," Center for Games and Playable Media, UC Santa Cruz, retrieved from promweek.soe.ucsc.edu.

McCoy, J., M. Treanor, B. Samuel, A. A. Reed, M. Mateas, and N. Wardrip-Fruin (2013) "Prom Week: Designing Past the Game/Story Dilemma," in *Proceedings of the Foundations of Digital Games*, pp. 94–101, retrieved from games.soe.ucsc.edu/prom-week-designing-past-gamestory-dilemma.

McCoy, J., M. Treanor, B. Samuel, A. A. Reed, M. Mateas, and N. Wardrip-Fruin (2014) "Social Story Worlds with Comme il Faut," *Computational Intelligence and AI in Games, IEEE Transactions* 6(2), 97–112.

Nelson, T. H. (1974) *Computer Lib / Dream Machines*, independently published.

Suchman, L. A. (1987) *Plans and Situated Actions: The Problem of Human-Machine Communication*, Cambridge University Press.

Wardrip-Fruin, N. (2009) *Expressive Processing: Digital Fictions, Computer Games, and Software Studies*, Cambridge, MA: MIT Press.