The Case for Concise Computational Creativity Systems

The Case for Concise Computational Creativity Systems

Nick Montfort

Abstract

In formalizing the notion of creativity – defining that concept so that it can be dealt with in the context of computing – researchers have sought to develop the most concise possible definition that captures the essence of creativity. But in building computational creativity systems, these same researchers are content to develop extremely elaborate and complex systems that implement a cornucopia of different ideas and approaches, or perhaps implement single ideas that are themselves very complex. Using examples from the literary arts, I argue that it is helpful to write very small programs that model specific techniques in different domains of creativity. These programs can then be used to identify how those specific techniques contribute to creativity, serving as the foundation for further work. By aiming for brevity and developing small, elegant systems, researchers can explore new aesthetic territory and better relate their results to the particular ways in which their systems operate.

Keywords

constraint, brevity, modeling, domain-specific techniques, story generation, poetry generation

Introduction

This essay makes the case for developing and encouraging the development of small-scale creativity systems. In concise systems, the concepts and their expression in design and implementation are simple, involving less code and less elaborate processes, much as sought-after theories of creativity are simple. The approach of developing these systems follows the approach of developing a definition or theoretical model of creativity. Rather than building a complex model (however well-grounded this model may be), the developer of a concise system seeks the most elegant and parsimonious approach that suffices to implement interesting aspects of creativity. Building on (Montfort 2012) and (Montfort and Fedorova 2012), this perspective on concise computational creativity systems is explained and examples are provided of concise systems in poetry and narrative, some of them grounded in well-known principles and some of them deriving from less usual explorations.

Simplicity, or the Art of Shaving

A widely-held belief in philosophy is that simpler theories should be preferred to more complex ones, and that of two equally explanatory theories, the simpler should be accepted. The principle, which has strongly influenced scientific inquiry, is classically known as Occam's Razor; it takes its name from the 14th century friar William of Ockham, although the connection between this principle and this thinker is, to say the least, unclear. This principle is explicitly invoked throughout the computational creativity literature both with reference to theoretical models and systems. I could find no examples where the Occam's Razor has been either questioned or extensively discussed within the computational creativity field.

To understand this principle in any detail, it is necessary to explain what makes one theory (or computational creativity system) simpler than another. In part, this can be explained by the philosophical distinction between parsimony and elegance. The question of parsimony is, how many kind of entities are proposed in the theory? To determine how elegant a theory or system is, one asks, how many axioms are being hypothezied, or how many instructions, functions, or logical

propositions constitute the system? Since system-building is a practical as well as philosophical matter, there is another type of simplicity that can be added to these two: simplicity of expression in clear, non-redundant code. For instance, if a task needs to be done six times, it can be more clearly expressed to a person in a loop rather than unrolled into six copy-and-pasted statements.

More could be said about these distinctions. In a digital system, how parsimonious the system is depends on the level of consideration. Different types or classes might reflect the different "kinds of entities" if the program is examined at a high level; at the level of execution on a processor, such kinds end up being flattened, as such things are not modeled in machine code. In any case, having offered a glimpse at how complex simplicity can be, it is enough for now to roll up these three aspects into one. These distinct aspects of parsimony, elegance, and simplicity of expression can be represented by conciseness – the length of code for a particular system, say in lines, characters, or bytes, as understood by software engineering as a classic measure of program complexity.

Now that the nature of simplicity has been described, there is still the question of why a preference for simplicity is justified. One could seek to justify this principle based on epistemic grounds – the nature of the universe is such that simpler theories are more likely to be true. However, one could also understand that this principle is a useful method, good in practice, whether or not this deeper justification has been defended. In the present discussion, it is the latter, more practical justification of concise systems that is advanced, with specific reference to concise systems that have been developed and used.

Surveying fine-grained generators of linguistic creativity, Cardoso, Veale, and Wiggins write "To achieve human levels of computational creativity, we do not necessarily need to start big, at the level of whole poems, songs, stories, or paintings; we are more likely to succeed if we are allowed to start small, at the level of simple but creative phrases, fragments, and images." (Cardoso et al. 2009, p. 18) While the point of this essay is precisely that we should "start small" with computational creativity, writing concise systems does not mean that one must focus only at the detailed level of phrases. Concise systems can work at the discourse level or the level of the overall narrative, working on text at many different scales. It is the principles that animate them, the elements of their processes, that should be kept "small." A system that has one elegant way of generating larger-scale poems or stories is, in this regard, more concise than a system that brings dozens of techniques to bear in generating short phrases.

Disadvantages of Complexity

Complex systems are necessary for modeling complex processes, and they will no doubt continue to have a place in the field. However, there are significant disadvantages to such systems.

Large-scale systems are less agile and adaptable to new purposes in several ways. Maintaining a large-scale system over years or decades, as platforms change and as new researchers join a team of collaborators, is labor-intensive and takes time away from developing new systems or extensions to the existing system. Understanding large-scale systems via analysis of code is difficult for the system's original creators and for new collaborators. All of these are the sorts of problems identified decades ago in the field of software engineering. Since computational creativity is an intellectual community that shares its results in publications, it is also worth noting that large, complex systems are hard to document and hard to reimplement based on documentation. In the case of system based on linguistic creativity, we can add to these difficulties that large, complex systems are more difficult to translate into other human languages.

The challenges of working with such systems can be seen in comparing some reimplementation efforts. Scott Turner's MINSTREL is an important early system in computational

creativity, one that specifically works in the domain of story generation. Work was completed on the system in 1993, and resulted in a book (Turner 1994). In 2009, Brandon Tearse began a grant-funded project to reimplement the system, to see what could be learned from the reimplementation process and from having a working version available for research use (Tearse et al. 2010). More than two years elapsed between the submission deadline for the first paper on Skald, February 5, 2010, and the first release of the code, on February 8, 2012, so it seems fair to say that this was an extensive project. Similarly, Iván Guerrero recently undertook the creation of a new version (a port) of Rafael Pérez y Pérez MEXICA, for purposes of extending the MEXCIA research in new ways. While the development of this Java version was done while also adding models of social and collaborative processes, work on the port was also an extensive project, even though Guerrero was collaborating with the original system's creator.

While these systems have been reimplemented once each, and with great effort, it is possible to consider, as a comparison, an older system that, while not explicitly a part of the computational creativity effort, was definitely a serious effort to simulate human conversation. This is Eliza, or more specifically the Eliza platform for conversation (written in Michigan Algorithm Decoder, using the SLIP libraries) running the Doctor script, developed during 1964-1966 by Joseph Weizanbaum, then at MIT. As of the end of April 2015, Google Scholar shows 2262 citations for Weizanbaum's major paper on Eliza (Weizenbaum 1966), and Github hosts literally dozens of reimplementations of the system in different programming languages – even though the original MAD source code cannot be found, and the earliest version available is one from the 1960s in Lisp. Systems that are significant from a cultural and research standpoint clearly do not have to be extremely complex; they do not have to require multi-year efforts at reimplementation. They can be both significant and simple, offering considerable impact, as Eliza did in its historical moment. Weizenbaum was able to program a simple and influential system even before it was possible to access complex linguistic resources such as tokenizers, part-of-speech taggers, chunkers, taggers, classifiers of many sorts and lexical and semantic networks with a one-line call through an API.

"Through The Park"

"Through the Park" (Montfort 2008) is extremely concise – its Python implementation is just an import statement, a list of 25 strings, and four lines of code. Nevertheless, it provides a highly simplified model of an important narrative technique, *ellipsis,* which will likely be a capability of any full-scale story generation system. Its small size and simple design make it easy to translate and re-use with other story content, showing its generality and allowing explorations of how the same principle works with different textual data.

Ellipsis can be understood as one possible tempo for narrating. One can narrate slower or faster; in this view of ellipsis, it is the leaping over of one or more events in an instant, which corresponds to telling the story at the fastest possible speed – an infinite speed (Prince 1982). The importance of this narrative technique has been articulated by narrative theorists, including Seymour Chatman: "Ellipsis is as old as The Illiad. But … ellipsis of a particularly broad and abrupt sort is characteristic of modern narratives" (Chatman 1978, p. 71). These omissions can allow the reader's imagination to fill the story in, as Fielding explains at the beginning of book III of *Tom Jones:*

> The reader will be pleased to remember, that … we gave him a hint of our intention to pass over several large periods of time …
>
> In so doing, we do not only consult our own dignity and ease, but the good and advantage of the reader: for besides that by these means we prevent him from throwing away his time, in reading

> without either pleasure or emolument, we give him, at all such seasons, an opportunity of employing that wonderful sagacity, of which he is master, by filling up these vacant spaces of time with his own conjectures; for which purpose we have taken care to qualify him in the preceding pages. (Fielding 2012)

Understanding ellipses has been the subject of some research, but generating ellipses has not been as well-studied. As recently as 2006, it appeared that computational narrative systems did not incorporate an ability to use ellipsis (Gervás et al. 2006). Those in the field have noted the relevance of this particular technique to cinematic and textual story generation, however. A capability for ellipsis was implemented in Curveship, an interactive fiction system (Montfort 2007, p. 107). Ellipsis was also supported in the development of the Mimesis system, because "narrative effects in [3D] environments are often achieved by selecting elements of the story world to elide from the narrative discourse (e.g., temporal and causal ellipsis) …" (Young 2007, p. 14)

"Through the Park" was first posted on the blog *Grand Text Auto* on November 20, 2008. It contains 25 sentences; all but eight are removed each time the system runs, with each of the remaining sentences kept in their original order. The sentences are:

> The girl grins and grabs a granola bar.
> The girl puts on a slutty dress.
> The girl sets off through the park.
> A wolf whistle sounds.
> The girl turns to smile and wink.
> The muscular man paces the girl.
> Chatter and compliments cajole.
> The man makes a fist behind his back.
> A wildflower nods, tightly gripped.
> A snatch of song reminds the girl of her grandmother.
> The man and girl exchange a knowing glance.
> The two circle.
> Laughter booms.
> A giggle weaves through the air.
> The man's breathing quickens.
> A lamp above fails to come on.
> The man dashes, leaving pretense behind.
> Pigeons scatter.
> The girl runs.
> The man's there first.
> Things are forgotten in carelessness.
> The girl's bag lies open.
> Pairs of people relax after journeys and work.
> The park's green is gray.
> A patrol car's siren chirps.

The system is meant to tell a story that is distantly but recognizably related to the folktale Little Red Riding Hood. On *Grand Text Auto,* readers were asked if they considered a system this simple to be a story generator. While not all commenters agreed that it was one, game developer Gregory Weir was

the first to reply, echoing Fielding in some ways:

> It's definitely a story generator. I like how my interpretation of the story can vary drastically on which cues are included. This is partly due to a few sharply-charged cues: the girl's smile, the knowing glance, the blank stare, and the police siren. Depending on which of these are included, cues like the girl's bag or the movement can be erotic or horrific.
>
> It does depend heavily on the mind's ability to fill in gaps … (Quoted in Montfort 2008a)

The sentences were consciously written to suggest (although not directly assert) that the two characters might be in a friendlier or more antagonistic relationship, and that the situation is more playful or sinister.

Developing this generator led to an improved understanding of ellipsis and of the characteristics (both ontological and linguistic) of story elements and their representations. In this simple system, there is no representation of the underlying fabula or story levels that is separate from a potential text, which may or may not be included in the final, realized discourse. Linguistically, it is problematic to include pronouns or other words that refer to other sentences; if such words are used, "she" or "he" might appear before "the girl" or "the man" are introduced. The more cohesive a text is, the harder it is to elide a sentence from it without adjusting the other sentences.

The underlying events in a story also should be able to stand apart, but for narrative interest, it is appropriate that they are, in Weir's terms, "charged" with varying emotional implications. While it seems valuable for the events to be of different valences, it is also helpful that they contribute to a consistent scenario and agree on, for instance, who the two main characters are and what the setting is.

Herbert Simon famously described the complex path of an ant's path on the beach, as it moves toward home around obstacles, as "really a complexity in the surface of the beach, not a complexity in the ant" (Simon 1969). Some might object to the extremely simple model of ellipsis presented here, pointing out, correctly, that the complexity comes from the "beach" of the 25 sentences and not the "ant" of the algorithm. However, it is nevertheless the case that the entire system presented includes both beach and ant, and narrative variation can be provided by assembling a system of this sort. If one wishes to later move the system's creativity, intelligence, or capability for interesting variation onto the "ant" of the algorithm, it would be wise to study how both the sentences and the algoritm for ellipsis work together in "Through the Park."

A more general model of ellipsis would allow different events/sentences to have different probabilities of being omitted; an even more general model would allow for conditional probabilities. Since experience with "Through the Park" suggests some qualities of the relationship between intersentential cohesion, the relationship between underlying events, and the opportunity for ellipsis, there are insights that could be applied in the development of more elaborate systems.

"The Two" and "About So Many Things"

In "The Two" (Montfort 2008c) the following sorts of stanzas/stories are generated:

> The student knocks on the teacher's door.
> She seeks help from her.
> They pray together.

The indigent turns to the librarian.
She seeks help from him.
One ends up with a broken will.

The student knocks on the teacher's door.
She begs him.
Rude gesture meets rude gesture.

The first line is simply one selected from several possibilities; the second line has "He" or "She" followed by a verb and associated words and then "him" or "her"; the final line is also one selected from several. All choices are made uniformly at random.

This is not a system for the generation of metrical verse, but the system does generate simple stanzas, and these stanzas are also stories. I call them "stanzories." [¿¿¿ "estroforias" ???]

The point of this generator is not a compelling new sort of computation or the production of novel situations or plots. Rather, the system is meant to challenge the reader to resolve referring expressions. In the second stanzory above, the reader must decide whether "she" is the librarian or the indigent, and, jointly, which one is "him." The subject and object of the first line may contribute to the reader's interpretation; there is a tendency to maintain the subject of the first sentence as the subject of the second. However, our ideas of power relations are also a factor: Why would a librarian seek help from an indigent? And, additionally, our gender stereotypes of different roles are significant: When no other information is presented, most people (in my country, certainly) will think of a woman when imagining a librarian, and assume that an indigent is a man.

As challenging as this can be (depending upon the particular generated story) for the reader, it can be even more challenging for a translator. Serge Bouchardon undertook a translation of this generator into French, where the gender of a character is usually indicated as soon as that character is introduced. By using the names of professions that begin with vowels and eliding what would otherwise be written beginning with the gendered definite articles "La" and "Le," Bouchardon kept the gender of characters ambiguous in the first line so that the reader is compelled to resolve pronouns according to syntax, power relations, and gender stereotypes:

L'économiste interpelle l'entrepreneur.
Il la congratule.
Six ans après, ni l'un ni l'autre ne se souvient de l'incident.

Since then, "Two Two" has also been translated into Spanish (by Carlos León), Russian (by Natalia Fedorova), Polish (by Aleksandra Malecka and Piotr Marecki) and Japanese (by Andrew Campana).

"The Two" was based, in part, on "About So Many Things" by Nannette Wylde (Wylde 1998). This was part of her Electronic Flipbooks series and originally written in Macromedia Director. "About So Many Things" places some strings that are selected uniformly at random into a simple template, the nature of which is self-evident. The template is simply "He" followed by a sentence completion and then, on the next line, "She" followed by a sentence completion, to produce text such as: "He likes chocolate / She thinks things should be different." The sentence completions range from being rather gender-neutral to being quite different when applied to people of different genders. For instance: "feels stressful," "is a good parent," "has a crush on the teacher," "is a firefighter." A lesson for large-scale creative text generator, evident even from this earlier work, is that determining the gender of a character, or transforming the gender of a character in an existing story, or leaving the

gender of a character open to the reader's guess, can be an important decision that is part of the creative process.

"Taroko Gorge"
This is a poetry generator that was written in Taiwan's Taroko Gorge National Park and on the plane afterwards on a single day, January 8, 2009 (Montfort 2009). It was written to model nature poetry, initially in Python and later ported to JavaScript. It can produce, for instance:

> Stone ranges the rocks.
> Heights sweep the vein.
>
> progress through the encompassing arched clear —
>
> Shape sweeps the flow.
> Heights dwell.
> The crags hold.
> Mist roams the shapes.
>
> enter the straight objective —

There are three types of lines generated. The first two, and the first and last line in the third stanza, are "path" lines meant to suggest the experience of walking along an outdoor path and seeing natural features either above or below. One or more "site" lines may be generated between these, as happens in the third stanza. These lines are less active and are meant to suggest the experience of stopping at a viewing area. Finally, the indented one-line stanzas are "cave" lines meant to suggest the experience of moving through the tunnels that, like the paths, were carved through the rock by Chiang Kai-shek's Nationalist army. In many of these tunnels, which are not lit with artificial light, it is not possible to see the exit from the entrance on the other side. This is why there is no visible ending to the "cave" line.

In part, "Taroko Gorge" was developed to show that just as one can visit a pleasing natural space and write a nature poem about it, one can write a generator of nature poetry and even do so at the natural site. The stanzas generated here, unlike those in some of my generated texts, are meant to be expressive and related to experience.

On March 27, 2009, Scott Rettberg announced his "remix" of this poetry generator, which he called "Tokyo Garage" (Rettberg 2009a). Rettberg changed the strings in the original program and added more of them without modifying the code. He did not keep to the original one page limit, producing a 6.3k file in comparison to the original 2.7k one. As Rettberg wrote in describing his project,

> Where Montfort's poem is an elegant nature poetry generator, I attempted to develop a poetry generator about … a city I have never been to, Tokyo. "Tokyo Garage" is about the idea of Tokyo, barraging us with images and archetypes absurd and sublime, in seemingly infinite variety. Reading the poem should feel much being dropped by a spaceship onto a busy corner of a strange city in the neon glow of a bustling, baffling night as the denizens stream past you now. (Rettberg 2009b)

Rettberg's version of the generator produces text of this sort:

> Chat client rescues the casinos.
> Transvestites walk.
> Zombies rest.
> Private security agents bribe the dancers.
>
> paint the disorganized strangely quiet disoriented peripheral--

By May 2010 J. R. Carpenter also had remixed "Taroko Gorge" as "Gorge," a piece about overeating and the sickeningly sweet textures of gastronomic excess (Carpenter 2010). Or, as Carpenter writes in describing the piece: "This never-ending tract spews verse approximations, poetic paroxysms on food, consumption, decadence and desire." The following is example output:

> Thirst agitates the vinaigrette.
> Veins dissolve.
> Blood vessels perfume.
> Incisor strengthens the finger tip.
>
> translate the faint cardamom cinnamon liquorice -

It is output from "Gorge" and from "The Chronicles of Pookie & JR" (a remix of "The Two") that forms the basis of Carpenter's book *Generation[s]* (Carpenter 2010).

More than two dozen remixes of "Taroko Gorge" have now been published or documented online; some of them have been exhibited, presented at festivals, and otherwise framed as new works. The remixes are quite various in tone and suggested subject matter. For instance, Andrew Plotkin's "Argot Ogre, OK!" (Plotkin 2011) is a meta-remix which automatically remixes all of the versions presented up to that point individually and in pairs. Almost all of these systems engage with the original structure of path, site, and view lines and relate the generated nature poem to new domains of poetic discourse.

Conclusion

Concise systems can be easily understood and modified, even without any involvement by their original creators. The new systems that are developed in this way can by interesting new types of cultural production, having value inside and outside the computational creativity community. They can be provocative, challenging the ideas that have been developed using large-scale systems and helping to develop some that have been overlooked. They can be used in teaching as the starting point for literary work or more elaborate exercises in computational expression. They can be translated to other human languageas more easily than complex projects can be, connecting different national and cultural contexts in which literary creative work is being done. Finally, they can be used to sketch, as an artist would, in preparation for undertaking a large-scale work.

Despite the worth of small-scale projects and the slight effort that is needed to execute them, the context of computer science, and many interdisciplinary contexts related to computing, discourages work on such smaller sketches and, instead, encourages researchers to proceed more directly to the development of elaborate, large-scale systems. There are a few cases where small-scale systems are seen to have a place – for use as examples, for instance, or as subsystems in a larger system – but not

many.

A student's dissertation project usually corresponds to a highly complex system, and the master's thesis and undergraduate capstone projects line up with slightly reduced, but still very complex, versions of the same. Most conference papers are based on work with complex systems. Ph.D. students in every field are already expected to understand their research area thoroughly by reviewing and understanding the relevant literature. It seems appropriate for them to spend as much time as they would reading a handful of articles in the development of one, or a few, small-scale systems. Such systems allow for different perspectives and approaches to attempted; they also encourage a focus on the essential and on extreme abstraction of method and of the domain of creativity.

There are institutions that support, or could support, the development of small-scale systems. In particular, hackathons, codefests, demoparties, and other sorts of competitions, as often arranged outside of an academic context as inside it, can be employed to encourage the development of simpler creativity systems. Being aware of these less conventional and formal contexts for work provides benefits for interdisciplinary researchers and artists, who can learn how short-form programs and concise works has proven valuable. There is good evidence for how being concise can be effective not only in computer-related practices, but also, for instance, in the literary arts, where poets, aphorists, and others have found it beneficial to compress language and write very short texts. Those who understand the point of composing a very concise haiku, for instance, may be better able to see why it would be worthwhile to develop a very concise haiku-generating system.

Simpler systems have definite benefits, and are philosophically aligned with the principles that encourage simple theories of creativity. These systems are easily portable across platforms, easily translated, easily generalized to different domains, and capable of capturing the essential aspects of important narrative techniques. Since they are also quick to put together, it would be sensible to do more to allow and encourage their development and to encourage the discussion of simple systems that have been developed.

Bibliography

Cardoso, Amílcar, Tony Veale, and Geraint A. Wiggins. "Converging on the divergent: The history (and future) of the international joint workshops in computational creativity." AI Magazine 30, no. 3, pp. 15+, 2009.

Carpenter, J. R. "Gorge." <http://luckysoap.com/generations/gorge.html>, <http://luckysoap.com/lapsuslinguae/2010/05/gorge/>. 2010.

Carpenter, J. R. *Generation[s].* Traumawien: Vienna. 2010.

Fielding, Henry. *The History of Tom Jones, a Foundling*. In *Wikisource.* <http://en.wikisource.org/wiki/The_History_of_Tom_Jones,_a_Foundling>. Last modified September 3, 2012.

Gervás, Pablo, Birte Lönneker-Rodman, Jan Christoph Meister, and Federico Peinado. "Narrative models: Narratology meets artificial intelligence." In *International Conference on Language Resources and Evaluation. Satellite Workshop: Toward Computational Models of Literary Analysis,* pp. 44-51. 2006.

Montfort, Nick. "Generating Narrative Variation in Interactive Fiction." Ph.D. diss., University of Pennsylvania, Department of Computer and Information Science. <http://www.cis.upenn.edu/grad/documents/montfort_000.pdf>. 2007.

Montfort, Nick. "Story Generation in 1k." On *Grand Text Auto.* <http://grandtextauto.soe.ucsc.edu/2008/11/20/story-generation-in-1k/>. 2008a.

Montfort, Nick. "Through the Park." On *nickm.com.* <http://nickm.com/poems/through_the_park.html>. 2008b.
Montfort, Nick. "The Two." On *nickm.com.* <http://nickm.com/poems/the_two.html>. 2008c.
Montfort, Nick. "Taroko Gorge." On *nickm.com.* <http://nickm.com/poems/taroko_gorge.html>. 2009.
Montfort, Nick. "XS, S, M, L: Creative Text Generators of Different Scales." Technical Report, The Trope Tank: TROPE-12-02. <http://trope-tank.mit.edu/TROPE-12-02.pdf>. 2012.
Montfort, Nick, and Natalia Fedorova. "Small-scale systems and computational creativity." In *International Conference on Computational Creativity*, p. 82. 2012.
Plotkin, Andrew. "Argot Ogre, OK!" <http://eblong.com/zarf/argot-ogre-ok.html>. 2011.
Prince, Gerald. *Narratology: The Form and Functiong of Narrative.* New York: Mouton Publishers. 1982.
Rettberg, Scott. "Tokyo Garage." On *retts.net.* <http://retts.net/tokyogarage.html>. 2009a.
Rettberg, Scott. Artist's statement about "Tokyo Garage," Digital Arts and Culture 2009 Literary Arts Extravaganza. <http://writerresponsetheory.org/dac09/gallery.htm>. 2009b.
Simon, Herbert A. *The Sciences of the Artificial,* MIT Press: Cambridge, Massachusetts. 1969.
Tearse, Brandon, Michael Mateas, and Noah Wardrip-Fruin. "MINSTREL Remixed: a rational reconstruction." In *Proceedings of the Intelligent Narrative Technologies III Workshop,* p. 12. ACM, 2010.
Turner, Scott R. *The Creative Process: A Computer Model of Storytelling and Creativity*. Psychology Press, 1994.
Weizenbaum, Joseph. "ELIZA—a computer program for the study of natural language communication between man and machine." *Communications of the ACM* 9, no. 1: 36-45, 1966.
Wylde, Nanette. "About So Many Things." In *Electronic Flipbooks,* CD-ROM. 1998.
Young, Michael. "Story and discourse: A bipartite model of narrative generation in virtual worlds." In *Interaction Studies* 8:2, 177–208. 2007.