# The Computer Generation of Novels: Theory and Practice

Nick Montfort*
* Massachusetts Institute of Technology, Cambridge, USA.
Email: nickm@nickm.com
Orcid: 0000-0001-7558-5160

**Abstract**

*How do authors use computers, not to write novels with word processors, but to generate novels with computer programs? For those interested in this practice, how to join those who are now working as author/programmers, combining writing techniques with computational techniques, developing innovative new ways of producing plots and prose? There are plenty of exciting opportunities for innovation in the area of computer-generated novels. To look ahead, it's first appropriate to look back and discover a bit about the history of computer-generated writing. Some discussion of the novel is important, too. Understanding more about this allows us to see how we can now elaborate, question, and combine past work — using new computational methods.*

**Keywords:** Novel, Computer Generation, Theory, Generative Novel, Electronic Literature

**Résumé**

*Comment les auteurs utilisent-ils les ordinateurs, non pas pour écrire des romans avec des logiciels de traitement de texte, mais pour générer des romans avec des programmes informatiques ? Pour ceux intéressés par cette pratique, comment se joindre à ceux qui travaillent désormais en tant qu'auteur-programmeur, combinant des techniques d'écriture avec des techniques informatiques, développant de nouvelles méthodes innovantes pour créer des intrigues et de la prose ? Il existe de nombreuses opportunités passionnantes d'innovation dans le domaine des romans générés par ordinateur. Pour anticiper l'avenir, il est d'abord approprié de regarder en arrière et de découvrir un peu l'histoire de l'écriture générée par ordinateur. Une certaine discussion sur le roman est également importante. En comprendre davantage à ce sujet nous permet de voir comment nous pouvons désormais élaborer, remettre en question et combiner le travail passé en utilisant de nouvelles méthodes informatiques.*

**Mots-clés :** Roman, génération du texte assistée par informatique, théorie, roman générative, littérature numérique

## 1.0. The Early History of Creative Text Generation

For more than seventy years, people have been using computers to produce creative text of different sorts. For instance, in 1952, Christopher Strachey wrote a program on the Manchester Mark I to automatically generate love letters. Although this text generator seemed to be more of a joke at first, he later wrote an article in an art magazine explaining how the program worked (Strachey 1954) — suggesting it was a project of artistic merit.

There are many other early examples of computer-generated creative writing from the "batch" or punched-card era of computing, a time before people could sit and type programs at a terminal (Montfort 2016). Early author/programmers had to write programs — often writing then by hand and having a keypunch operator encode them on punched cards — then submit those cards to a computer center and wait for the result. There might be a minor error which caused the program to loop endlessly or to not run at all, in which case the author/programmer would have to correct the program and repeat the whole process, probably not getting the next result until the next day. Despite these challenges, several remarkable early projects were developed even before people were able to program interactively on minicomputer or had their own microcomputers.

In Germany in 1959, Theo Lutz wrote a generator of lines that introduced what sounded like logical propositions, with language taken from Franz Kafka's *The Castle.* Lutz published a scientific paper about his project. Similarly, Victor H. Yngve, working at MIT in 1961, developed a generator of random sentences in the course of his academic work in machine translation. Although it was based on simple language at the beginning of a children's book, it could recursively build phrases within phrases and produce uncanny results. Brion Gysin, working with Ian Sommerville, automated a process he originally had been undertaking by hand and began to produce computer-generated permutation poems in the 1960s in England. In South Africa in the early 1960s, J. M. Coetzee, the author who would later win the Nobel Prize for Literature, was working at IBM and wrote a program to generate random five-word lines. One of the most remarkable early computer-generated poems was the 1967 "A House of Dust" by Alison Knowles & James Tenney, which balanced repetition and variation, had a distinctive stanza form, and was issued as a printed set of unique outputs. I have re-implemented these historical text generators so that (although they are materially and visually different from the originals) they function, as best I can tell, as the original programs did. They are available in JavaScript and Python versions (Montfort 2014–2018). The first computer-generated novel is not easy to pin down. Sheldon Klein was working on an "Automatic Novel Writing" project and issued a report about the work (co-authored by eight others!) in the 1970s (Klein et al. 1973). The project is remembered as the first major effort at a novel-writing system, but it did not bear fruit in the form of one or more computer-generated novels.

There are other relevant academic projects, but when it comes to published books, many points to *The Policeman's Beard is Half-Constructed,* attributed to the computer program Racter (Racter 1984). This book is a provocative one and led some early critics to denounce it as not really being computer-generated. These early critics were misguided — the text is the output of a computer program — but at the same time, the book is obviously not a novel in any strict sense. Its pages include prose text, poems, and human-created illustrations.

Author/programmers have produced something like novels, and persistent work in creative text generation and in computer storytelling (Sharples & Pérez y Pérez 2022). To begin to consider what these are, we should think about computer-generated book that engage with the novel tradition. *The Policeman's Beard* is an innovative and interesting book, but it does not present itself as a novel or connect to the tradition of this genre. There are, however, other computer-generated books worth considering as novels. These are of quite varied sorts, some online, some in print. Many of them are much harder to read and appreciate in traditional ways than is *The Policeman's Beard.* The discussion will proceed to some of these. First, however, it's worthwhile to consider what some prominent examples of (human-written) novels are, and how different these can be.

## 2.0. Novels of Many Different Sorts

Consider just a few of the major differences between Chinua Achebe's *Things Fall Apart* (1992) and Amos Tutuola's *The Palm-Wine Drinkard* (1953). Achebe's novel is about the initially traditional life of an Igbo man, Okonkwo, who is a successful (if flawed) figure at the beginning of the book. He has a personal downfall, goes into exile, and returns to find his village transformed by colonialism. Tutuola's novel, in contrast, has as its full title *The Palm-Wine Drinkard and His Dead Palm-Wine Tapster in the Dead's Town.* It tells the story of a man who consumes copious amounts of palm wine. When he cannot replace his skilled supplier of this wine, who died in a fall from a tree, he goes on a fantastical quest to find him in the land of the dead, one blending many elements of Yoruba folklore. During an early encounter, he declares himself "Father of gods who could do everything in this world." While Achebe used a phrase from a William Butler Yeats poem as the title of his novel and wrote in a way that seemed more conventional to Western readers, Tutuola invented the word *drinkard* and wrote in a very different and distinctive style. Yet these are not from radically different times, places, or language communities: They are two English-language novels from Nigeria, originally published in the same decade, Achebe's in 1958, Tutuola's in 1952.

Of course, this sort of novelistic variety is not specific to English. Albert Camus, born in Algeria when it was part of France, wrote one very influential French novel of the 20th Century, the spare, direct, and brief *L'Etranger* (1978). Marcel Proust's *À la recherche du temps perdu* (1987) seems to be different along every dimension, although the two novels are not from radically different times. Proust's was published during 1913–1927; Camus's came out in 1942.We can even find cases in which the same author has written radically different novels, as a brief excerpt makes clear. Here are the first sentences of James Joyce's first novel, originally published in 1916, *A Portrait of the Artist as a Young Man* (1928).

The wide playgrounds were swarming with boys. All were shouting and the prefects urged them on with strong cries. The evening air was pale and chilly and after every charge and thud of the footballers the greasy leather orb flew like a heavy bird through the grey light. He kept on the fringe of his line, out of sight of his prefect, out of the reach of the rude feet, feigning to run now and then. He felt his body small and weak amid the throng of the players and his eyes were weak and watery. Rody Kickham was not like that: he would be captain of the third line all the fellows said. This is what we find at the top of page 1 of Joyce's last novel, from 1939, *Finnegans Wake* (2010):

> Riverrun, past Eve and Adam's, from swerve of shore to bend of bay, brings us by a commodius vicus of recirculation back to Howth Castle and Environs. Sir Tristram, violer d'amores, fr'over the short sea, had passencore rearrived from North Armorica on this side the scraggy isthmus of Europe Minor to wielderfight his penisolate war: nor had topsawyer's rocks by the stream Oconee exaggerated themselse to Laurens County's gorgios while they went doublin their mumper all the time: nor avoice from afire bellowsed mishe mishe to tauftauf thuartpeatrick:

*Finnegans Wake,* abundant with neologisms and with references that run down to the morphological and etymological levels, asks for entirely different reading strategies than does Joyce's first novel. It is, indeed, an entirely different kind of book, one that questions the boundaries of the novel and of the English language.

## 3.0. Novel Writing and Novel Generation Goals

Since there is no one kind of novel, we should not expect there to be only one kind of computer-generated novel. Just as writers have different purposes, author/programmers do, too.

So, what reasons? It's impossible to enumerate them, but some of the motivations include:

1. Producing a bestseller and making money
2. Embodying cultural ideas
3. Expressing personal experience
4. Exploring narrative possibilities
5. Exploring language

My own background, expertise, and motivations do not include (1) — I would have no idea how to go about writing a blockbuster thriller or other massively popular book. I also don't know of many author/programmers who are really aiming to do this, although some of them seek certainly publicity and try to gain wider interest in their work.

My creative writing (whether it is computational or not) is also not about (3), the expression of personal experience. It may be that computer-generated literature is simply not suitable for such expression, when compared to traditional novel writing. Or, it may be differently suited to the task. But I will have to leave it to others to discuss this from a practical standpoint.

My goals as an author/programmer are focused on (4), the exploration of narrative possibilities and (5), working with and through language itself to make discoveries. Among novelists, writers of metafiction are some who have investigated narrative extensively. Joyce, in *Finnegans Wake,* is an explorer of language, as is Gertrude Stein in her *Making of the Americans,* which has a much smaller vocabulary but develops new types of syntax and works on our reception through repetition and variation.

While I seldom start a project with goal of (2) embodying cultural ideas, as Achebe and Tutuola accomplished in dramatically different ways, I seek to explore and uncover aspects of culture *through* my engagement with narrative and language.

Now I can offer a revision of this list, from the perspective of my own authorial and computational practice, with one elaboration, to consider another possible reason to specifically *generate* a novel:

1. ~~Producing a bestseller and making money~~
2. ((((Embodying cultural ideas))))
3. ~~Expressing personal experience~~
4. Exploring narrative possibilities
5. Exploring language
6. Exploring computation

Just as my own novel-generation processes involve the exploration of language, it is also important that I explore computing and its potential for engagement with language, literature, and culture. Computer generating a novel is not only *writing;* it is a process of creating software art, *computational art.*

### 4.0. What Do Computers Do Well?

Because I want to explore language and computation, I don't want to do violence to either one. I want to use the capabilities of the computer as a textual system. I don't want to go against the grain. That leads me to ask, what are computers good at doing?

Are they good at (computationally, programmatically) expressing personal experience? Of course, an online system such as a social network can foster personal expression and storytelling. But in my experience, it is harder for a computer program to be as personally expressive as a traditional text — a novel, a memoir, an essay. To the extent that one is expressive, it is usually because of the *data* involved, not the computational processes. The computer generation of novels does not, at this point, seem inclined toward the expression of personal experience.

What about working through historical, cultural, and mythical intricacies? There are certainly special cases where computational techniques, like non-computational techniques for experimental

writing, can be applied so as to shed light on such topics. But even in these cases, it is through working with and understanding these techniques (which involve the exploration of narrative, language, and computation) that cultural engagement is possible. So, I don't find this to be among the most obvious capabilities of the computer.

What are the obvious ones?

- Regularity
- Repetition
- Randomness

Computers are formal and can produce language regularly, by permutation, by combination, by recursively including sentences as phrases within sentences, by appending more, bit by bit. A program can of course repeat language at any level, such as by letter, by word, by phrase, by sentence, by paragraph. And the computer has the ability to choose from a distribution of language at random — "pseudorandomly," if we want to be precise, but this approximation is certainly good enough for artistic purposes. Each of these can be done with at most a few lines of code. The development of personal expression and the modeling of cultural phenomena are much grander goals that would take much more work to attain.

### 5.0. Novels Aren't New

With this in mind, what are easy but substantial things for a computer-generated novel to be *about*? It seems clear to me that (just as Tutuola conflated existing Yoruba folktales in his first novel) they can be about earlier stories and earlier novels. By connecting to such previous literary work (and orature), computer-generated novels can be significant and can question and extend this work in important ways.

It's not just computer-generated novels that have this sort of work, and not just *The Palm-Wine Drinkard.* James Joyce's *Ulysses* (1990) could not exist without *The Odyssey.* In the literature of the United States, Cormac McCarthy's *Blood Meridian* (1986) is a conflation of Samuel Chamberlain's *My Confession: Recollections of a Rogue* (a memoir), *Moby-Dick,* and *Paradise Lost.* One of the foremost early works of hypertext fiction, Shelley Jackson's *Patchwork Girl* (1995), draws on Mary Shelley's *Frankenstein* and L. Frank Baum's *The Patchwork Girl of Oz.*

It seems ironic, because the word *novel* literally means "new." But to make a "new" novel, a powerful way is to find existing writing and make it new by combining, elaborating, and questioning. Doing so can allow an author to drive forward into new possibilities for narrative and language. In the case of an author/programmer, there will be new computational possibilities to discover, too.

One example from my own work is the writing/programming of *Megawatt* (Montfort 2014). This book is based on Samuel Beckett's second novel, *Watt* (1970), which includes several very unusual passages that are permutations of language. Those parts of *Watt* seem as if they were computer-generated, although Beckett wrote the book before digital, general-purpose computing existed, when he was working for the French resistance in World War II. In developing *Megawatt,* I decided to take these bizarre passages and, first, write code that would actually generate them computationally — it would do the permuting, automatically, that Beckett did by hand. Having done that, I went on to extend these passages and make them even more absurd and unreadable. Why would I bother with this? For me, writing the code, producing the generating program, was a way of reading *Watt* in more detail. It helped me understand exactly what Beckett had done as a writer. The publication of my *Megawatt* provided both a bizarre text and all of the code that generated it, so that readers who wanted to deeply explore *Watt* in the same way I did could see my methods all laid out.

I am hardly alone in using earlier novels as the basis for new computer-generated work. Consider Milton Laufer's *A Noise Such As a Man Might Make* (2018), published by Counterpath Press in the Using Electricity series (which I edit). The text generation is done by a simple Markov chain process, with two pages of code in the back of the book revealing the details of how it was done. The

book conflates two famous American novels in a fragmentary and confused way, taking advantage of how proper names almost never appear in either book and how both books have an old man and a boy as their main characters. By bringing together two stories of struggles against nature and challenges to traditional notions of masculinity in this provocative framework, the novel builds something new and compelling.

Another printed novel is *1 the Road* by Ross Goodwin (2018). This one is hard to read in a different way, with text tagged with different times of generation, produced based on camera images, GPS data, and in-car conversations during a road trip. The specific AI generation method used was a long short-term memory (LSTM) recurrent neural network. This book refers in its title, subject matter, and to some extent its process to Jack Kerouac's *On the Road,* a chronicle of a road trip typed on a long scroll of paper.

Sofian Audry's *for the sleepers in that quiet earth.* is a set of 33 unique books (2017–2019), each of which was generated using the same process and had only Emily Brontë's novel *Wuthering Heights* as data. As with *1 the Road,* an LSTM was used. Each epoch of learning is represented in each book, so each of the books progressively shows an increasing "understanding" of the text. The texts were generated in 2017 and the printed book published in early 2019 by my micropress, Bad Quarto.

*World Clock* (Montfort 2013) came about in the context of Darius Kazemi's NaNoGenMo (National Novel Generation Month), an annual activity that happens each November. NaNoGenMo is not actually national (anyone around the world can participate) and most of the computer-generated books developed are not novels; many of them, including excellent ones, are more in poetic traditions. My *World Clock,* programmed for the first NaNoGenMo, is not directly based on an earlier novel, but it draws on two experiment writing works, "The Chronogram for 1998" by Harry Mathews and the "One Human Minute" by Stanislaw Lem, which is a review of a non-existent book and, in English translation, appears in Lem's book of the same name. My program in this case narrates 1440 different episodes, different ways that people read all throughout the world at different times — at each different minute — in a single day.

Another NaNoGenMo production is Robin Camille Davis's *If on a winter's night a library cardholder* (2016), inspired conceptually and formally by Italo Calvino's novel *Se una notte d'inverno un viaggiatore* (*If on a winter's night a traveler).* The original novel has a frame story about a reader referred to as "you" who is seeking a complete edition of a book that is off to a very compelling start. Each time "you" think you have found the book, a new chapter in a completely different genre is presented instead. Camille's computer-generated novel takes "you" to every public library in New York City, where, flipping through a random book, a random passage is presented to you.

A very recent publication that can be approached as a novel is Arwa Michele Mboya's *Wash Day* (2023, also in my Using Electricity series from Counterpath). Her book is not directly based on a novel; it draws its text from a different source. The language it uses comes from transcripts of YouTube videos and braids together discussion of hair care rituals from Black women. Computation collages this together to provide a new sort of reading experience and an intimate perspective on the many stages and details of caring for curly hair. Mboya is from Nairobi and now lives and works in Los Angeles.

### 6.0. Getting Started

While it's possible to start a novel generation project with a grand plan, an author/programmer can also work in a *bottom-up* fashion by modifying and elaborating a simple program. I created an untitled program specifically so that participants in AELAIWC2022 can study the code and use it for any purpose, including as the basis for an artistic project. It is available here and will run automatically when one visits this link: https://nickm.com/if/aelaiwc22.html

The "View Page Source" option will reveal the code. To facilitate study, I am including the entire short program below. The JavaScript program at the core of this generator is embedded in a Web page, written in HTML. The actual program is found between <script> and </script>.

```
<!DOCTYPE html>
<html>

<head>
   <meta charset='utf-8'>

   <title>For AELAIWC2022 (by Nick Montfort)</title>
   <style>
      body {
         margin: 10vw;
         font-size: 4vh;
      }
   </style>
   <script>
      const templates = [
         'so OPPONENT-1 MEETS OPPONENT-2 . OPPONENT-2 FIGHTS OPPONENT-1 . OPPONENT-1 FIGHTS OPPONENT-2 . OPPONENT-2 FIGHTS OPPONENT-1 . OPPONENT-2 TIRES . OPPONENT-1 TIRES . There is peace',
         'now OPPONENT-1 MEETS OPPONENT-2 . OPPONENT-2 MEETS OPPONENT-1 . OPPONENT-1 FIGHTS OPPONENT-2 . OPPONENT-2 FIGHTS OPPONENT-1 . MEDIATOR RESOLVES . There is peace',
         'OPPONENT-1 MEETS OPPONENT-2 . MEDIATOR MEETS OPPONENT-1 . There is peace',
      ];

      const opponents = ['the soldier', 'the bureaucrat', 'the engineer', 'the businessman', ];

      const mediators = ['the elder', 'the child', 'the scholar', ];

      const meets = ['notices', 'sees', 'observes', 'hears', 'meets', ];

      const fights = ['argues with', 'strikes', 'punishes', 'insults', 'decries', ];

      const tires = ['tires', 'becomes weary', 'has enough', ];

      const resolves = ['chastises them', 'shows them how to behave', 'shames them', ];

      function choose(array) {
         return array[Math.floor(Math.random() * array.length)];
      }

      function typeset(sentences) {
         let text = '<p>\n';
         for (sentence of sentences) {
            text += ' ' + sentence.charAt(0).toUpperCase() + sentence.slice(1) + '. ';
         }
         text += '\n<\p>\n<hr />\n';
         text = text.slice(0, -7);
         return text;
      }

      function generate() {
         let text = choose(templates);
         let opponent_1 = choose(opponents);
         let opponent_2 = choose(opponents);
         if (opponent_1 == opponent_2) {
```

```
            opponent_2 = "the other " + opponent_2.split(' ')[1];
        }
        let mediator = choose(mediators);
        text = text.replace(/OPPONENT-1/g, opponent_1);
        text = text.replace(/OPPONENT-2/g, opponent_2);
        for (i = 0; i < 4; i++) {
            text = text.replace('MEDIATOR', choose(mediators));
            text = text.replace('MEETS', choose(meets));
            text = text.replace('FIGHTS', choose(fights));
            text = text.replace('TIRES', choose(tires));
            text = text.replace('RESOLVES', choose(resolves));
        }
        x.innerHTML += typeset(text.split(' . '));
    }
  </script>

</head>

<body onload='for (let n=0;n<1000;n++) {generate();}'>
  <div id=x />
</body>

</html>
```

Here is a brief excerpt of sample output — the system produces a much more extensive list of paragraphs:

> So the bureaucrat notices the soldier. The soldier punishes the bureaucrat. The bureaucrat argues with the soldier. The soldier strikes the bureaucrat. The soldier tires. The bureaucrat tires. There is peace.
>
> Now the soldier meets the other soldier. The other soldier hears the soldier. The soldier insults the other soldier. The other soldier strikes the soldier. The elder chastises them. There is peace.
>
> So the businessman observes the engineer. The engineer punishes the businessman. The businessman strikes the engineer. The engineer decries the businessman. The engineer has enough. The businessman has enough. There is peace.

As written, this program is repetitive and limited, but it is not meant to be a compelling text generator in its current state. Instead, it is an invitation. You are welcome to modify the words and phrases, and you can do so even if you have no background in programming. Simply save the Web page to your own computer (for instance, your Desktop) and open it in a text editor. After you change elements of it and save the changes, open the page in a Web browser to see what new sorts of text will be generated.

What if you are someone without programming background, and as you are editing the program, you make some horrible mistake and break some of the code, so the program no longer works? Just download the Web page again; it will still be there, online. By making small changes a bit a time and checking your results, you will be able to "undo" any edits that break the program while developing your own creative voice. In doing so, you'll not only develop a creative project. You will also learn more about what is code, what is data, and how computing and language can work together.

Good luck with your story generation and with developing your own computer-generated novel!

**Works Cited**


Achebe, Chinua. *Things Fall Apart.* New York: Alfred A. Knopf. 1992.

Audry, Sofian. *for the sleepers in that quiet earth.* Boston & New York: Bad Quarto. Texts generated 2017. 2019.

Beckett, Samuel. *Watt.* New York: Grove Press. 1970.

Davis, Robin Camille. *If on a winter's night a library cardholder.* https://github.com/robincamille/nanogenmo2016/blob/master/outputs/novel3.md 2016.

Camus, Albert. *L'étranger*. Paris: Gallimard. 1978.

Goodwin, Ross. *1 the Road.* Paris: Jean Bôite Editions. 2018.

Jackson, Shelley. *Patchwork Girl, or, A Modern Monster a Graveyard, a Journal, a Quilt, a Story & Broken Accents*. Watertown, MA: Eastgate Systems. 1995.

Joyce, James. *A Portrait of the Artist as a Young Man.* New York: Modern Library. 1928.

Joyce, James. *Ulysses*. 1st Vintage International ed. New York: Vintage Books. 1990.

Joyce, James. *The Restored Finnegans Wake*. Eds. Danis Rose & John O'Hanlon. London: Penguin Classics, 2012.

Klein, S., et al. *Automatic Novel Writing: A Status Report*. Technical Report 186. Computer Science Department, University of Wisconsin. 1973.

Laufer, Milton. *A Noise Such as a Man Might Make.* Using Electricity series. Denver: Counterpath. 2018.

McCarthy, Cormac. *Blood Meridian, or, The Evening Redness in the West*. New York: Ecco Press. 1986.

Mboya, Arwa Michele. *Wash Day.* Using Electricity series. Denver & New York: Counterpath. 2023.

Montfort, Nick. *World Clock.* Cambridge, MA: Bad Quarto. 2013.

Montfort, Nick. *Megawatt.* Cambridge, MA: Bad Quarto. 2014.

Montfort, Nick. *Memory Slam: Reimplemented computational poetry, electronic literature, and digital literary art that was originally developed on batch processing systems.* https://nickm.com/memslam/ 2014–2018.

Montfort, Nick. "Batch/Interactive." *Time: A Vocabulary of the Present,* pp. 309-322. 2016.

Proust, Marcel, and Jean-Yves Tadié. *À la recherche du temps perdu,* Nouv. éd. Paris: Gallimard, 1987.

Racter. *The Policeman's Beard Is Half Constructed.* Intro. Will Chamberlain, Illus. Joan Hall. New York: Warner Books. 1984.

Sharples, Mike and Rafael Pérez y Pérez. *Story Machines: How Computers Have Become Creative Writers.* London & New York: Routledge. 2022.

Strachey, Christopher, "The 'Thinking' Machine," Encounter 3:4, 25–31. 1954.

Tutuola, Amos. *The Palm Wine Drinkard and his Dead Palm Wine Tapster in the Dead's Town.* New York: Grove Press. 1953.

**Short Bio**

As a poet and artist, Nick Montfort uses computation as his medium. His computer-generated books include *#!* and *Golem.* His digital projects include the collaborations *The Deletionist* and *Sea and Spar Between.* Montfort is a scholar, researcher, and educator. His MIT Press publications include *The New Media Reader* (which he co-edited) and *Twisty Little Passages, The Future,* and *Exploratory Programming for the Arts and Humanities.* He is professor of digital media at MIT and principal investigator in the Center for Digital Narrative at the University of Bergen. He directs a lab/studio, The Trope Tank, and lives in New York City.