Generators that Read

Max Kreminski
UC Santa Cruz
1156 High St
Santa Cruz, California 95064
mkremins@ucsc.edu

Isaac Karth
UC Santa Cruz
1156 High St
Santa Cruz, California 95064
ikarth@ucsc.edu

Noah Wardrip-Fruin
UC Santa Cruz
1156 High St
Santa Cruz, California 95064
nwardrip@ucsc.edu

ABSTRACT

Most discussions of procedural content generation have focused primarily on the artifacts that generators produce or the process by which these artifacts are created. Less focus, however, has been placed on the methods by which generators interpret their input. Many generators take complex input, act as part of a generative pipeline, are part of a mixed-initiative communication with the user, or otherwise need to take context into account during generation. In these cases, the process by which the generator reads and makes sense of its input is often just as interesting as the process by which it produces an output artifact. It is worthwhile to take a closer look at how generators read. Via a case study of two erasure poetry generators, we propose the concept of a *generativist reading*: a process of reading that produces generative models. Many existing generators have dual input/output or reading/writing processes that are presented as a monolithic unit, but our understanding of both processes and results is enriched when we clearly distinguish between how generators write and how they read.

CCS CONCEPTS

•General and reference → Design; •Applied computing → Computer games; **•Human-centered computing →** Interaction design theory, concepts and paradigms;

KEYWORDS

procedural content generation, reading, generative pipelines, close reading, context-sensitive generation, mixed-initiative co-creativity, generativist readings, erasure poetry generation, proceduralist readings

ACM Reference format:
Max Kreminski, Isaac Karth, and Noah Wardrip-Fruin. 2019. Generators that Read. In *Proceedings of Foundations of Digital Games, San Luis Obispo, California, USA, August 26-30, 2019 (FDG'19),* 7 pages.
DOI: 10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Recent years have seen an increased interest in approaches to procedural content generation that interpret and meaningfully respond to complex forms of input, often forms of input that were not originally intended to be used as input to a generator. Challenges such as the Settlement Generation Challenge of the Generative Design in Minecraft Competition [33] have explicitly encouraged a focus on the development of *context-sensitive* generators, capable of taking an arbitrary *Minecraft* map as input and generating a settlement that fits well within the context of that particular map. Projects like *WikiMystery* [2] have used existing corpuses of open data as a foundation for the generation of murder mystery scenarios. And essentially all approaches that fall under the umbrella of *procedural content generation via machine learning* [37] begin by training on a large corpus of input data.

Nevertheless, there remains a tendency in the community to talk about generation as though it is primarily a process of *writing* or ex nihilo creation of artifacts, sidelining or even outright erasing the sophistication of the increasingly complex components of generators that concern themselves primarily with the *reading* of input. We believe it is worthwhile to look more closely at how generators read.

At the same time, we observe a tendency within computer science research to place an emphasis on *correctness* and *unambiguousness* in the development of algorithms for interpreting complex forms of input. From natural language processing to emotion recognition from face images, much of the literature implicitly assumes that it is both possible and desirable to produce an objectively correct and unambiguous interpretation of these complex inputs within the computer. We contend, however, that extracting machine-usable meaning from complex input necessarily entails a creative act of interpretation: the complexity of the input ensures that it could always be read differently, and the approach to interpretation that you choose to apply will affect the nature of the interpretation you produce.

Moreover, we argue that, for the purpose of procedural content generation, it is often beneficial to accept "incorrect" and "ambiguous" readings of complex input as valid, enabling our generators to produce surprising outputs by selecting from among a wide range of mutually incompatible but equally viable interpretations of the same input. Likewise, when designing generators to be used in mixed-initiative contexts, it may be desirable to embrace multivocality by exposing the user to a variety of possible interpretations of the same input, thereby helping them to see alternatives they might not otherwise have considered.

There are many different approaches to reading, and within the humanities, many forms of critical practice are organized around a particular theory or methodology of reading. Borrowing this lens, we suggest that different approaches to reading might inform or inspire the development of novel approaches to procedural generation, and that analyses of existing generators may benefit from thorough examination of how these generators read their input in a way that engages deeply with a particular theory of reading.

FDG'19, San Luis Obispo, California, USA

DOI: 10.1145/nnnnnnn.nnnnnnn

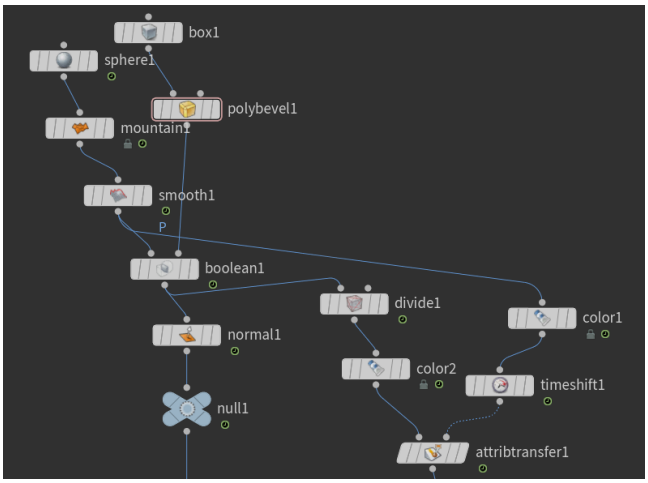

Figure 1: A portion of a node-graph in SideFX's Houdini, a procedural 3D modeling and effects software, demonstrating how the web of nodes is connected via inputs and outputs.

Rather than viewing generators as monolithic black boxes, generators can be viewed as a pipeline of data transformations [7]. Many common procedural content generation techniques can be subdivided into modular units that chain into each other, often forming a complex web. In some fields, such as parametric 3D modeling or shaders, these connections are made explicit through the use of node-based interfaces. Node based interfaces explicitly route the outputs of one stage of the pipeline into the inputs of the next (Fig. 1). In the PCG field, this has been expanded by the Generominos ideation cards, which model a pipeline of data transformations, explicitly highlighting the importance of matching inputs and outputs [8].

For example, consider WaveFunctionCollapse (WFC), a constraint-based and example-driven approach to procedural content generation. WFC is composed of two linked generative processes: an input model that translates an image into adjacency constraint data, and a probabilistic constraint solver that turns the data into new generated images [19]. The constraint solver cannot act on its own without some form of input to specify the constraint data. The generator needs both its input and its output components to operate. Further, WaveFunctionCollapse has multiple input models that read input data with different approaches. By examining how the different input models process the data that the generator reads, we can gain greater insight into the process of the generator as a whole.

As another example, consider that many of the novel generators created for National Novel Generation Month (NaNoGenMo)[1] draw on the same source texts—frequently including *Alice in Wonderland*, *Moby-Dick*, and *The Odyssey*—yet produce very different outputs. This is possible in part because each generator takes a different approach to reading its input text.

[1]https://nanogenmo.github.io/

In this paper, we first discuss the reasons we might want our generators to interpret and respond to complicated forms of input—or, in other words, to read. Then we investigate, from a humanities perspective, what exactly it means to read, with an eye to how a diverse range of existing approaches to reading might inform the development of new approaches to procedural content generation. Next we introduce the concept of *generativist readings*: readings of texts that take the form of sets of rules for producing more texts of a similar nature. We present a brief comparison of two similar erasure poetry generators that are differentiated almost exclusively by their approaches to reading. Finally, we discuss the broader implications of a deeper investigation of reading in the context of generative methods.

We focus our investigation primarily on generators that read complex inputs which were not originally intended primarily as inputs to a generator. A generator that reads complicated generator-specific configuration files, for instance, is of less interest to us than a generator that reads *Minecraft* worlds or arbitrary English texts. Nevertheless, generators that read complicated forms of generator-specific input may still be amenable to some of the same forms of analysis, so we do not exclude them entirely from the scope of our interest.

2 WHY DO WE WANT OUR GENERATORS TO READ?

2.1 Context-Sensitive Generation

There are a wide range of problem domains that call for *context-sensitive* forms of procedural content generation: the generation of artifacts that, rather than standing alone, are expected to fit in to some existing complex context. One example of a problem that calls for context-sensitive generation can be found in the Settlement Generation Challenge of the Generative Design in Minecraft Competition [33], which tasks competitors with building a generator that can produce a convincing settlement on any arbitrary chunk of terrain in the voxel-based construction game *Minecraft*. In order to reward competitors for producing generators that are truly context-sensitive, competitors are not given access to the specific maps that will be used as testbeds for their settlement generation processes. Therefore, they must do their best to produce generators that are capable of functioning in a wide range of potential contexts, and are disincentivized to produce generators that are prone to overwriting large swaths of the existing terrain without regard for how a generated settlement might fit more naturally into its surroundings.

In other cases, it is often desirable to generate content that fits around or fills the gaps between a number of fixed "landmarks" without overwriting those landmarks. This too necessitates generators that are capable of reading and responding to an established context.

2.2 Generative Pipelines

Building on the notion of context-sensitive generation, it is important to acknowledge that many generators do not operate in a vacuum. In particular, especially in games that make extensive use of procedural content generation, a single generator is often merely one component of a larger generative pipeline that consists

of many generators wired together end-to-end. In these situations, the complex output of one generator becomes complex input to another generator, and the downstream generator must then interpret the input in some nontrivial way in order to generate an output artifact that matches or meaningfully adapts to the input artifact.

When working with procedurally-generated base terrain, a frequent problem is to appropriately respect the elevation of the terrain when placing objects, particularly when generating buildings, towns, and road networks on rough terrain [12]. For example, *Minecraft* settlement generation is itself an instance of a problem where generative pipelining leads to a need for downstream generators that are capable of interpreting and responding to the complex output of upstream generators (in this case, the base terrain generator itself) [33]. Likewise, world generation in the roguelike *Caves of Qud* makes use of multiple distinct generators, each of which feeds into other generators in the pipeline [14].

2.3 Mixed-Initiative Co-Creativity

When building mixed-initiative co-creative tools [40] that attempt to use procedural generation to supplement or augment the work of a human user, it is especially critical for the generative systems employed by the tool to be capable of reading or interpreting whatever the human user has created so far. In cases where the generator is not capable of doing this, it is likely to step on the user's toes in various ways, for instance by overwriting their work and replacing it with generated content.

This behavior can be seen in the context of mixed-initiative 2D platformer level design tools with *Morai Maker* [15], an AI-driven game level editor in which a human user and an AI level designer take turns collaborating on a single shared design. Unlike earlier human/AI collaborative level design tools such as *Tanagra* [35], which provides the human user with a suite of tools for communicating their design intent to the AI directly, *Morai Maker* attempts to infer what it should do in response to the human user's actions largely without explicit guidance. Partly as a result of this lack of guidance, the AI collaborator has a tendency to apparently ignore or repeatedly overwrite the human user's edits to the shared design, which can produce frustration in users who desire a greater degree of control over the design process. The user experience of *Morai Maker* under its current design constraints, then, hinges on its ability to read the design the user has created so far, ideally with an eye to deriving an understanding of the user's intention purely from the actions they have taken. Improving the AI collaborator's capability to read the human user's partial level designs would directly result in an overall improvement to the user experience of collaborating with the AI.

3 WHAT DOES IT MEAN TO READ?

There are many different kinds of reading. Within the humanities, the term "reading" has taken on an expansive definition as an umbrella term under which a wide variety of approaches to the analysis and interpretation of texts may be considered. Indeed, following the *linguistic turn* [32] in the humanities, the term "text" has itself taken on a broader meaning than in its original sense of purely linguistic or written works, and is now widely understood to encompass all kinds of cultural artifacts [13, 21], from advertisements to zoo signage. As such, the notion of "reading" is a contested one, and merits further examination if we are to apply it as a lens to the understanding of generative methods. We do not attempt a comprehensive survey of all possible approaches to reading, as such an undertaking would be well outside the scope of this work. Instead, we offer samples of several diverse perspectives on the question of what it means to read, with the goal of illustrating the range of approaches that are possible and hinting at how different approaches to reading might inform or inspire different approaches to procedural generation.

3.1 Close Reading

Close reading is "the thorough interpretation of a text passage by the determination of central themes and the analysis of their development" [18]. "Close reading concerns close attention to textual details with respect to elements such as setting, characterization, point of view, figuration, diction, rhetorical style, tone, rhythm, plot, and allusion," often examining the gap between what is said and what can be inferred [30]. The methodology is evaluated, in part, by its explanatory power for the details of the presentation.

A generator that performs a close reading of its input is concerned with the details of the input. Interactive Data Visualization Inc.'s SpeedTree, to give one example, pays careful attention to the shape of the nearby terrain—at a fairly high level of granularity—when placing a tree.

3.2 Distant Reading

Positioned in contrast to close reading, in *distant reading* "the reality of the text undergoes a process of deliberate reduction and abstraction" [27]. Rather than concerning itself with the details of presentation, distant reading is a process that operates on models and visualizations of the text. This thousand-foot view reveals commonalities and structures that would otherwise go unseen but that can now be visualized by graphs, maps, trees, and other data structures.

One approach to designing a generator is to program a model of the process that created the desired result, or has a visual similarity to the desired result. In the first case, a teleological terrain generator [1] might include simulations of geological processes, erosion, the shifting flow of rivers, and so on. In contrast, an ontological terrain generator [28] might use Perlin noise to emulate the shape of the desired terrain. In either case, the generator is modeling a system, and both creating and analyzing the generator involve a process of reading via that model.

3.3 Critical Approach

A critical approach to reading is performed by mapping a theory onto a literary work to explain its meaning, a two-directional process where "the theory should illuminate a work, and a work should illuminate a theory" [30]. We can characterize image generation via deep learning neural networks (such as Deep Dream [26]) as a generator that reads its input by mapping a theory (learned in training) onto the input image.

3.4 Hermeneutics

A *grammatical hermeneutic* reading attempts to derive the meaning of a text through analysis of elements that are present within the text itself, rather than from elements outside the text, such as the intention of the author [39]. For our purposes, an important factor to recognize is that this often deliberately results in multiple parallel readings of a single text. For example, the European medieval exegesis of sacred texts, influenced by Aquinas, simultaneously looked for four senses in every text: a literal (*sensus literalis*), moral or tropological (*sensus tropologicus*), allegorical (*sensus allegoricus*), and mystic (*sensus anagogicus*) sense [5] [17, p. 99].

The recognition that a single work can have multiple senses challenges the assumption that a reading will arrive at a single, unambiguous classification. A text can be read in multiple ways simultaneously, and multiple generators can produce a variety of valid interpretations—even mutually incompatible interpretations—of the same input.

3.5 Poetics

In contrast to hermeneutic approaches to reading, poetics represents an alternative perspective that focuses instead on the *felt effects* of a text in the reader [10, 31]. Whereas it is fairly straightforward to see how close or distant reading might be employed in the construction of a generator, it is less obvious how the framework of poetics might be applied to a machine reader, especially insofar as the term "felt effects" may be interpreted as concerning itself primarily with a text's *physiological* effects. Nevertheless, generation based on a subjective experience of a text—perhaps from the perspective of one interpreting agent among many in an artificial artist commune such as CheapArtistsDoneQuick [9] or The Digital Clockwork Muse [34]—remains an intriguing possibility.

For an example of an existing generative technique that may exhibit something like mechanical felt effects, consider word vectors, which are constructed through a mechanical analysis of word adjacencies [23, 24]. Because word vectors describe points in a much larger *continuous* space, they allow for a kind of "semantic bleed" between adjacent words, similar to how ambiguity and wordplay operates in textual poetics for human readers. Further, word vectors capture semantic relationships in the text that was read [25], indirectly modeling or mimicking some of the subjective effects of reading in human readers through their very method of construction.

3.6 Proceduralist Readings

Proceduralist readings represent still another approach to reading, this time an approach native to game studies and focused primarily on the interpretation of interactive or rule-based texts such as videogames. Proceduralist readings "address a convergence point between [...] expression and interpretation" and focus on "internal readings of a game's dynamic" yielding "meaning derivations" [38]. These meaning derivations are structured logical arguments for the interpretation of the game's mechanical and sensory cues as the higher-level meanings that emerge as the game's dynamics and aesthetics.

Proceduralist readings have themselves been proceduralized: building on the Operational Logics framework as a game description language, Martens et al. describe a procedure for automated reasoning about games [22]. This proceduralization, in turn, has become an essential component of Gemini [36], a generator of abstract games. Gemini provides users with a specification language that they may use to specify what arguments they want the generated games to make. Gemini then uses this specification to guide its search within the design space of possible games, identifying and returning games that may be read in the desired ways.

3.7 Takeaways

As evidenced by the brief sampling here, a wide variety of theories of reading have been introduced, and each such theory has interesting potential implications for generators that read. Proceduralizing various approaches to reading may prove a successful strategy for the discovery of new approaches to generation. Moreover, deep engagement with a particular theory of reading may enable deeper analysis of existing generators that process complex inputs.

It is also important to note the double meaning of the term "reading" as it is commonly understood: the same term applies both to the process of interpretation and to the concrete interpretations that are produced through the application of this process. Reading a text produces a particular reading of the text in question, and a reading of a text may be examined, understood, or interpreted as a concrete artifact or text in and of itself. Therefore, when a generator reads an input text, it may be useful to consider the reading it produces as an artifact that merits examination, even if this reading is not intended to be directly consumed or experienced by the generator's audience at the end of the generative process. In the following section, we further examine the implications of this view.

4 GENERATIVIST READINGS

By analogy to proceduralist readings, we propose the notion of *generativist readings*. A generativist reading is an interpretation of a text consisting of a set of rules for generating artifacts similar to or based on the text. Much like a proceduralist reading of an interactive text focuses on deriving meaning from the rules or procedures within the text, a generativist reading attempts to answer the question of what this text can tell us about how to produce more similar texts.

Generativist readings need not be exclusively constructed by mechanical processes. Oftentimes, when we create generators to produce types of artifacts that were previously exclusively handmade, we essentially find ourselves manually conducting a generativist reading of a corpus of examples. For instance, if a human reader was to read *Moby-Dick* and handcraft a Tracery [6] grammar that utilizes vocabulary and sentence structures drawn from the book to produce sentences that sound plausibly as though they could be drawn directly from the source text, the resulting grammar would constitute a generativist reading of the text. Similar practices are not uncommon among Twitter bot creators, who may often begin by writing out the source text that a generator will try to imitate and then recursively substitute grammar rules in place of concrete words, gradually sublimating the text itself into a statistical model

of the text. Manual generativist readings may even be used as an instrument of critique: consider Umberto Eco's proposal of algorithms for plot generation in the style of various filmmakers as a way of parodying those filmmakers' styles. [11]

However, in practice, many generativist readings *are* constructed by mechanical processes. Mechanical processes of generation that rely on generativist readings typically begin by conducting one or more generativist readings of an input text or corpus. The generator then queries or manipulates these readings to produce individual output artifacts. For instance, text generation with Markov chains follows a two-step process. First, the computer conducts a generativist reading of the source text by moving over the text and tracking the overall frequency with which each word it encounters follows each other word. Then, the process of generation employs the statistical model created through reading to write new texts that imitate the read text. This same structure can be observed in many forms of generation, especially in procedural content generation via machine learning [37], which hinges entirely on the construction of generative models from which individual output artifacts can then be sampled.

Some practitioners in the generative art world recognize the reading process as an intrinsic part of generation. For example, in the view of everest pipkin:

> When I say that the creative act is the reader's, I imply the creator as well as the audience. When working with generative text, it is impossible not to read. One has to look for bodies of text that can function as useful sources for tools; big enough, or concrete enough, or with the right type of repetitive structure; learnable. And then one has to read the output of such machines, refining rules and structures to fix anything that breaks that aura of the space one is looking for. In this, we are not unlike the medieval scholar who studies holy verse to become fluent enough in that space that it becomes building block. [29]

Generators with more complexity stem from reading with more sophistication: compared with a Markov chain, the better performance of Long Short Term Memory neural network architecture [16] can be partially attributed to a more in-depth reading process that takes into account correlations that are more complex than the short horizon a reasonable Markov chain can remember.

While reading is an intrinsic part of machine learning, it is not confined to neural networks. Procedural generation algorithms like WaveFunctionCollapse [19] also depend on reading. WaveFunctionCollapse performs a generativist reading on the images it uses as input and translates them into a model that can in turn be used to generate new examples that imitate structures it has recognized in the input.

Not every generator that makes use of reading as part of the generative process necessarily conducts a generativist reading. Gemini [36], a generator of simple abstract games based on Martens et al.'s proceduralization of proceduralist readings [22], conducts proceduralist readings on the games it generates in order to determine whether or not they can be interpreted in a way that matches the arguments the user has specified they want to make. However, this reading occurs only internally—the things it reads are the incomplete games that it has itself generated—and it does not build a generative model of these games based on its reading, but merely uses its reading to direct its search within the possibility space.

In contrast to Gemini's proceduralist reading approach, the Ludi game generation system reads abstract games in terms of their rules, as formatted in a game description language. Ludi reads game rules through a process of analyzing self-play simulations. Ludi then evaluates their fitness as defined by a set of algorithmically-measured aesthetic criteria [4]. The writing process combines the read game rules into new mixtures of rules. These are added to the collection of game descriptions to create the next generation of games and the reading and writing process repeats. It has been suggested to us that Ludi uses a form of generativist reading: the analysis and evolving process is aimed at generating new game rules that are similar to the game descriptions it read as input, and it builds up a generative model that attempts to describe the possibility space of aesthetically interesting game descriptions.

5 CASE STUDY

As an illustration of the potential importance of reading to the generative process, we now present a brief comparison of two similar erasure poetry generators: *The Deletionist* [3] and *blackout* [20]. Both of these generators are packaged as browser bookmarklets and present themselves as ways of turning arbitrary web pages into poetry by erasing most of a page's text. Moreover, the processes by which these generators write their modifications to the target page are both straightforward and nearly identical. The difference between these two generators thus lies almost entirely in how each generator reads or interprets a page's text prior to modification.

The Deletionist interprets each webpage as a single unit, considering all the text on the page at once rather than breaking it up into smaller pieces for analysis. It decides which words to erase deterministically, such that running it repeatedly on the same webpage will produce the same result every time. Its selection of which parts of the text to retain is based on one of several regular expression patterns, many of which use either the start or end of words to determine whether some or part of the word should be retained. In some cases, for instance, it will choose to retain primarily words beginning with the letter M, while in other cases, it will choose to retain words ending with a period. It also frequently retains certain common whitelisted words, such as "from" and "like". Once it has decided which parts of the text to retain, all other parts are erased.

blackout takes a markedly different approach. Rather than treating the whole page as a single unit, it reads each paragraph in isolation and makes no attempt to coordinate its reading of one paragraph with its reading of the next. It reads nondeterministically, such that running it repeatedly on the same webpage will typically produce different results from one run to the next. Its selection of which words to retain, meanwhile, makes use of part-of-speech tagging and probabilistic fuzzy matching of valid sequences of parts of speech, recognizing simple declarative sentences that could be formed by omitting some or all of the words in a paragraph and selecting some valid sentence for each paragraph.

Figure 2: An erasure poem generated by running *blackout* on one paragraph of the 2019 PCG Workshop call for papers, taking the form of a simple declarative sentence.

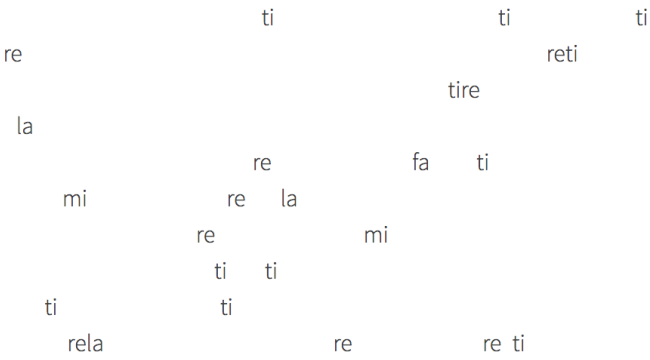

Figure 3: An erasure poem generated by running *The Deletionist* on the same input, taking the form of a series of syllables corresponding to musical notes.

A side-by-side comparison of the outputs that these two generators produce when run on the same input page will further confirm that the differences between them lie largely in terms of how they read the input they are provided. This serves to illustrate an important principle: for generators that take nontrivial input (and that can thus can meaningfully be said to consist of a distinct "reading" and "writing" component), it is possible to alter or replace either the reading or the writing part without changing the other component and still get interestingly different results.

Both generators are equipped with a variety of patterns, each of which is tested against the input before the generator makes a final determination about how to generate the output poem. This can be seen as leveraging a form of internal multivocality, embracing the ambiguity of the reading process by engaging with a variety of interpretations of the same input. Moreover, in the case of *blackout*, the generator creatively misapplies a part-of-speech tagging algorithm taken from a natural language processing package to deliberately preserve ambiguities of interpretation in the source text. Rather than flattening ambiguous words (which could be parsed as having several distinct and mutually incompatible parts of speech) into a single most likely interpretation, as in the typical application of similar algorithms, *blackout* avoids this flattening by treating words as having arbitrarily many distinct part-of-speech tags until the final poem is rendered.

6 CONCLUSIONS

As we have seen, reading is a useful analytic lens we can use to better understand how generators process input, including context sensitive generation, chaining generators together into pipelines, and furthering mixed-initiative co-creativity.

By treating the problem of extracting machine-usable meaning from complex input as a form of reading, we suggest that—when crafting a generator that reads—it is often desirable to preserve ambiguity and embrace the possibility of incorrectness, rather than attempting to read correctly and unambiguously. Interpretation is essentially a kind of creative act, and different approaches to interpretation may inform the development of new approaches to generation. A terrain generator that models a physical process reads its model in a different way than a tree generator that curves roots around nearby stones. The idea of a proceduralist reading is an already accepted methodology in game studies, and, in part, inspires our introduction of generativist readings.

A generativist reading is an interpretation of a text into a generator of similar texts. The existing practice of many generative artists and bot creators can be considered as generativist readings, while mechanical generativist readings are foundational to many existing approaches to PCG, including procedural content generation via machine learning. The greater the complexity of the generator, the more sophisticated the reading: breakdowns in generative pipelines can be caused by a mismatch in complexity between the input and the output, as with Compton et al.'s example of complex input from a Kinect being effectively reduced to button-presses by a badly designed pipeline [8].

As a brief illustration, we contrasted two erasure poetry generators, *The Deletionist* [3] and *blackout* [20] and demonstrated that the differences between them rests almost entirely on the different approaches to reading the input.

Many generators have interesting approaches to reading. However, the way generators read is less discussed than how they write, and seldom separated into a subject worthy of discussion on its own. Despite this, we have demonstrated several ways in which particular generators cannot be understood without first examining how they read. Therefore, when we discuss generators, we should go beyond discussing what the output looks like and consider including a clear separation in our discussion: between the ways in which a generator writes and the ways in which it reads.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous referees for their valuable comments and helpful suggestions.

REFERENCES

[1] Alan H. Barr. 1991. Teleological Modeling. In *Making Them Move*, Norman I. Badler, Brian A. Barsky, and David Zeltzer (Eds.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 315–321. http://dl.acm.org/citation.cfm?id=111154.111171

[2] Gabriella Alves Bulhoes Barros, Michael Green, Antonios Liapis, and Julian Togelius. 2019. Who killed Albert Einstein? From open data to murder mystery games. *IEEE Transactions on Games* (2019).

[3] Amaranth Borsuk, Jesper Juul, and Nick Montfort. 2013. The Deletionist. https://thedeletionist.com. (June 2013).

[4] C. Browne and F. Maire. 2010. Evolutionary Game Design. *IEEE Transactions on Computational Intelligence and AI in Games* 2, 1 (March 2010), 1–16. DOI: http://dx.doi.org/10.1109/TCIAIG.2010.2041928

[5] Harry Caplan. 1929. The Four Senses of Scriptural Interpretation and the Mediaeval Theory of Preaching. *Speculum* 4, 3 (1929), 282–290. http://www.jstor.org/stable/2849551

[6] Kate Compton, Ben Kybartas, and Michael Mateas. 2015. Tracery: An Author-Focused Generative Text Tool. In *Interactive Storytelling*, Henrik Schoenau-Fog, Luis Emilio Bruni, Sandy Louchart, and Sarune Baceviciute (Eds.). Springer International Publishing, Cham, 154–161.

[7] Kate Compton and Michael Mateas. 2017. A generative framework of generativity. In *Experimental AI in Games Workshop 2017, at the Thirteenth Artificial Intelligence and Interactive Digital Entertainment Conference*. The AAAI Press, Palo Alto, California, Snowbird, Little Cottonwood Canyon, Utah USA. https://aaai.org/ocs/index.php/AIIDE/AIIDE17/paper/view/15896

[8] Kate Compton, Edward Melcer, and Michael Mateas. 2017. Generominos: Ideation Cards for Interactive Generativity. In *Experimental AI in Games Workshop 2017, at the Thirteenth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*. AAAI Press, Snowbird, Little Cottonwood Canyon, Utah USA. https://aaai.org/ocs/index.php/AIIDE/AIIDE17/paper/view/15898

[9] Kate Compton, Johnathan Pagnutti, and Jim Whitehead. 2017. A shared language for creative communities of artbots. In *Proceedings of the 2017 Co-Creation Workshop*. Eighth International Conference on Computational Creativity, Atlanta, Georgia, USA.

[10] Jonathan Culler. 2015. *Theory of the Lyric*. Harvard University Press, Cambridge, MA, 6.

[11] Umberto. Eco. 1993. Make Your Own Movie. In *Misreadings*. Harcourt Brace & Co., San Diego, 145–155.

[12] Arnaud Emilien, Adrien Bernhardt, Adrien Peytavie, Marie-Paule Cani, and Eric Galin. 2012. Procedural generation of villages on arbitrary terrains. *The Visual Computer* 28, 6-8 (2012), 809–818.

[13] P. Ffrench. 2012. Text. In *The Princeton Encyclopedia of Poetry and Poetics : Fourth Edition*, Roland Greene, Stephen Cushman, Clare Cavanagh, Jahan Ramazani, Paul F. Rouzer, Harris Feinsod, David Marno, Alexandra Slessarev, and Inc. ebrary (Eds.). Princeton University Press, 41 William Street, Princeton, New Jersey 08540, 1425–1426.

[14] Jason Grinblat and Brian Bucklew. 2019. Math for Game Developers: End-to-End Procedural Generation in 'Caves of Qud'. https://www.gdcvault.com/play/1025914/Math-for-Game-Developers-End. In *Game Developer's Conference 2019*. San Francisco, CA USA.

[15] Matthew Guzdial, Nicholas Liao, Jonathan Chen, Shao-Yu Chen, Shukan Shah, Vishwa Shah, Joshua Reno, Gillian Smith, and Mark Riedl. 2019. Friend, Collaborator, Student, Manager: How Design of an AI-Driven Game Level Editor Affects Creators. In *Proceedings of ACM SIGCHI Conference on Human Factors in Computing Systems (CHI)*.

[16] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9, 8 (1997), 1735–1780. DOI: http://dx.doi.org/10.1162/neco.1997.9.8.1735 arXiv:https://doi.org/10.1162/neco.1997.9.8.1735

[17] R. Hollander. 2001. *Dante: A Life in Works*. Yale University Press.

[18] Stefan Jänicke, Greta Franzini, Muhammad Faisal Cheema, and Gerik Scheuermann. 2015. On close and distant reading in digital humanities: A survey and future challenges. In *Eurographics Conference on Visualization (EuroVis)-STARs. The Eurographics Association*, Vol. 2. 6.

[19] Isaac Karth and Adam M. Smith. 2017. WaveFunctionCollapse is Constraint Solving in the Wild. In *Proceedings of the 12th International Conference on the Foundations of Digital Games (FDG '17)*. ACM, New York, NY, USA, Article 68, 10 pages. DOI: http://dx.doi.org/10.1145/3102071.3110566

[20] Max Kreminski. 2017. blackout. https://mkremins.github.io/blackout. (March 2017).

[21] Yuri Lotman. 1977. *The Structure of the Artistic Text*. University of Michigan: Department of Slavic Languages and Literature, Ann Arbor, Michigan.

[22] Chris Martens, Adam Summerville, Michael Mateas, Joseph Osborn, Sarah Harmon, Noah Wardrip-Fruin, and Arnav Jhala. 2016. Proceduralist readings, Procedurally. In *Proceedings of the Twelfth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*. https://www.aaai.org/ocs/index.php/AIIDE/AIIDE16/paper/view/14061

[23] Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *CoRR* abs/1301.3781 (2013). arXiv:1301.3781.

[24] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and Their Compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2 (NIPS'13)*. Curran Associates Inc., USA, 3111–3119. http://dl.acm.org/citation.cfm?id=2999792.2999959

[25] Tomas Mikolov, Wen tau Yih, and Geoffrey Zweig. 2013. Linguistic Regularities in Continuous Space Word Representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT-2013)*. Association for Computational Linguistics. https://www.microsoft.com/en-us/research/publication/linguistic-regularities-in-continuous-space-word-representations/

[26] Alexander Mordvintsev, Christopher Olah, and Mike Tyka. 2015. Inceptionism: Going deeper into neural networks. (2015).

[27] F. Moretti and A. Piazza. 2005. *Graphs, Maps, Trees: Abstract Models for a Literary History*. Verso.

[28] F. Kenton Musgrave. 2003. 14 - A brief introduction to fractals. In *Texturing and Modeling (Third Edition)* (third edition ed.), David S. Ebert, F. Kenton Musgrave, Darwyn Peachey, Ken Perlin, Steven Worley, William R. Mark, John C. Hart, F. Kenton Musgrave, Darwyn Peachey, Ken Perlin, and Steven Worley (Eds.). Morgan Kaufmann, San Francisco, 428 – 445. DOI: http://dx.doi.org/10.1016/B978-155860848-1/50043-7

[29] everest pipkin. 2016. A Long History of Generated Poetics: cutups from Dickinson to Melitzah. (2016). https://medium.com/@everestpipkin/a-long-history-of-generated-poetics-cutups-from-dickinson-to-melitzah-fce498083233 Archived by WebCite at http://www.webcitation.org/76fwxfAz5.

[30] Herman Rapaport. 2011. *The Literary Theory Toolkit: A Compendium of Concepts and Methods*. Wiley-Blackwell, Chichester, West Sussex, United Kingdom.

[31] B. M. Reed. 2012. Poetics, Western. In *The Princeton Encyclopedia of Poetry and Poetics : Fourth Edition*, Roland Greene, Stephen Cushman, Clare Cavanagh, Jahan Ramazani, Paul F. Rouzer, Harris Feinsod, David Marno, Alexandra Slessarev, and Inc. ebrary (Eds.). Princeton University Press, 41 William Street, Princeton, New Jersey 08540, 1058–1064. http://ebookcentral.proquest.com/lib/ucsc/detail.action?docID=913846

[32] Christoph Reinfandt. 2009. Reading Texts after the Linguistic Turn: Approaches from Literary Studies and Their Implications. In *Reading Primary Sources: The Interpretation of Texts from Modern History*, Benjamin Ziemann and Miriam Dobson (Eds.). Routledge, London, UK, 37–54.

[33] Christoph Salge, Michael Cerny Green, Rodgrigo Canaan, and Julian Togelius. 2018. Generative Design in Minecraft (GDMC): Settlement Generation Competition. In *Proceedings of the 13th International Conference on the Foundations of Digital Games (FDG '18)*. ACM, New York, NY, USA, Article 49, 10 pages. DOI: http://dx.doi.org/10.1145/3235765.3235814

[34] Rob Saunders and John S Gero. 2001. The digital clockwork muse: A computational model of aesthetic evolution. In *Proceedings of the AISB'01 Symposium on Artificial Intelligence and Creativity in Arts and Science*, Vol. 1. Citeseer, University of York, Heslington, York, YOlO 5DD, England, 12–21.

[35] Gillian Smith, Jim Whitehead, and Michael Mateas. 2010. Tanagra: A mixed-initiative level design tool. In *Proceedings of the Fifth International Conference on the Foundations of Digital Games*. ACM, 209–216.

[36] Adam Summerville, Chris Martens, Ben Samuel, Joseph Osborn, Noah Wardrip-Fruin, and Michael Mateas. 2018. Gemini: Bidirectional generation and analysis of games via ASP. In *Proceedings of the Fourteenth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE 2018)*. AAAI Press.

[37] Adam Summerville, Sam Snodgrass, Matthew Guzdial, Christoffer Holmgård, Amy K Hoover, Aaron Isaksen, Andy Nealen, and Julian Togelius. 2018. Procedural content generation via machine learning (PCGML). *IEEE Transactions on Games* 10, 3 (2018), 257–270.

[38] Mike Treanor, Bobby Schweizer, Ian Bogost, and Michael Mateas. 2011. Proceduralist Readings: How to Find Meaning in Games with Graphical Logics. In *Proceedings of the 6th International Conference on Foundations of Digital Games (FDG '11)*. ACM, New York, NY, USA, 115–122. DOI: http://dx.doi.org/10.1145/2159365.2159381

[39] Georgia Warnke. 2016. Hermeneutics. (Nov 2016). DOI: http://dx.doi.org/10.1093/acrefore/9780190201098.013.114 Published Online. Accessed 2019 April 17.

[40] Georgios N Yannakakis, Antonios Liapis, and Constantine Alexopoulos. 2014. Mixed-initiative co-creativity. In *Proceedings of the 9th International Conference on the Foundations of Digital Games*. Society for the Advancement of the Science of Digital Games.