Visualizing Redundant Paths Through Hypertext Narratives

Stacey Mason
University of California, Santa Cruz
1156 High St
Santa Cruz, CA 95064
1 (831) 459-0111
stcmason@ucsc.edu

ABSTRACT
Visualization of interactive narrative structures remains a difficult problem: while authoring systems have developed robust ways to visually communicate narrative structures, comparatively little work has been done towards visualizing the output of playthroughs, namely story metrics and redundant content paths. This paper explores an interactive visualization that quickly helps authors spot redundant paths in interactive narratives made with the Twine authoring tool. By treating branching narratives as directed graphs and walking the graph of each unique path through a story, the visualization collects all possible paths through a narrative, using force-directed graph layouts to visualize the narrative structure and increasing size of the node thickness of the paths through the narrative to indicate nodes and links visited on different possible paths. The outcome is a clear indication of redundant paths and most-visited nodes.

Categories and Subject Descriptors
[D.2.2 Design Tools and Techniques] User Interfaces [Hypertext/Hypermedia]: Visualization. I.7.2 [Document Preparation]: hypertext/hypermedia

General Terms
Documentation, Design, Human Factors

Keywords
Visualizations, Hypertext, Twine, Interactive Fiction Interactive Narrative

1. INTRODUCTION
With the growing popularity of tools for novice game-makers to create text games, interactive narratives have seen a recent explosion in creators interested in hypertext narrative authorship [1]. While many of the authoring tools available offer creators the chance to visually enhance the authoring process with adjustable visualization tools, comparatively fewer tools visualize the results of the authors’ work, that is the finished narrative. Fewer still create an easy way to integrate visualizations with the tools authors are already using.

This paper outlines a visualization tool for interactive narratives created with the Twine hypertext authoring tool. The tool takes the HTML file output by the Twine system, parses it to obtain the link structure of the narrative, and visualizes that structure using force-directed graph techniques. It uses a depth-first search to approximate all possible paths through the narrative and adjusts the sizes of the nodes to suggest nodes that will be visited across multiple paths. The outcome is a clear indication of redundant paths and most-visited nodes.

2. MOTIVATION
Conventional approaches to authoring interactive narratives still rely on rigorous playtesting and intuition to guide their creation. While some companies are experimenting with embedded data tools to drive ongoing narrative content [6], these methods are still not widely adopted.

Some of the most popular tools for interactive narrative creation, including Twine, Inform, Tinderbox, and Undum[4,8], leave much to be desired in terms of information conveyed about the stories created with them. While some of the more flexible visual tools like Tinderbox or Twine provide authoring interfaces that convey
information about the structure of the authoring process—e.g. robust graph layouts—they offer comparatively fewer visual representations for how the narrative might actually be experienced or which metrics the authors might consult to improve the story.

Since “improving the story” is a subjective undertaking, the best a visualization tool can hope to offer is a concise way to convey information, leaving the authors to adjust their stories to their own aesthetic goals. Still, the visualization of story metrics is lacking in most of the major systems: number of endings, paths to those endings, redundancies in paths, and narrative-state tracking are all underdeveloped in these tools.

3. RELATED WORK

Many comparisons have been made between interactive digital narratives and their analog counterparts in modernist literature. A popular example often used to explain interactive narrative experiences is the Choose Your Own Adventure book series popularized by Bantam Books [2].

Christian Swinehart created some of the most impressive and robust visualizations of interactive narrative structures in his visualization of different paths of Choose Your Own Adventure Books (CYOAs) by generating all possible outcomes of playthroughs and visualizing the CYOAs as directed graphs [3]. In these graphs, nodes represent pages of narrative content and edges represent jumps and choice points in the text, such as “To attack the dragon, turn to page 5. Or to pet the dragon turn to page 20” (Fig. 1). A path from the “Start” node to a leaf is considered a possible path through the story (a playthrough).

By configuring the structure of the story as a directed graph, Swinehart is then able to generate all possible paths through the narrative using a depth-first search technique. In this way, Swinehart visualizes each possible path through a CYOA by increasing the size of nodes and thickness of edges in relation to how many times the links are traversed on various playthroughs.

While this methodology worked to great success in Swinehart’s examples for interactive analog narratives, interactive digital narratives present additional challenges. For example, Swinehart’s source material contained no cycles, no states to track, and no recurses. Still, the prospect of visualizing possible endings and repeated passages by generating all possible play-throughs of a narrative is interesting and worth further exploration.

In terms of authoring tools, Eastgate Systems’ hypertext tool, Tinderbox, [1] largely believed to be one of the most robust authoring tools for interactive narrative in terms of visualization capabilities, provides many dimensions of visual signifiers for authors to utilize on individual nodes and edges—size, shape, color, opacity, drop shadow color and distance, etc.—but does not provide information on the possible outcomes of playthroughs including whether a certain node will be visited on successive plays.

Chris Klimas’s open-sourced authoring environment, Twine, provides a more minimal authoring aesthetic, omitting colors, shapes, and other visual signifiers offered by Tinderbox to instead offer only boxes for nodes and arrows to signify text passages and links.

While Tinderbox offers some of the most robust visual features in hypertext authoring, due to price and its steep learning curve, Twine is by far the most widely-adopted hypertext authoring environment. Twine outputs HTML that is—for the most part—easy to work with. For its wide adoption, generally non-technical user base, Twine is an excellent candidate for incorporation into an interactive tool that can provide some of the visual features Twine itself lacks.
4. APPLYING ANALOG TECHNIQUES TO DIGITAL NARRATIVES

In this project, I hoped to apply Swinehart’s techniques to the interactive digital narrative space by creating a visualization that will show naive likelihood of endings and repeated narrative passages. While digital narratives introduced new difficulties to the problem—the growing convention toward using cycles in narratives, keeping track of state, and so forth—that lie outside the scope of this paper, the application of Swinehart’s visual techniques to Twine narratives proved fruitful.

4.1 Hypertext Narratives as Graphs

Just as Swinehart approached Choose Your Own Adventure books, hypertext narratives are best understood as directed graphs in which the reader (or “player” if the hypertext piece is a game with a desired goal endstate) begins the narrative at one node (“Start”) and ends the piece at a designated ending, which could be among many and could be desirable or not.

4.2 Parsing Twine Narratives

Because Twine files are designed for novice creators to be able to make interactive stories and publish them to the Web, all of Twine’s data about the path through its narrative are encoded in the single HTML file it outputs. While visual stylings such as animation, and computational game effects created in Javascript are minified and stored into the HTML file in ways that are difficult to recover for a simple parser, the information about link structures are stored in special tags in the HTML output file called `<tw-passagedata>`.

Passage data tags contain the name of the node, an id, the node’s story text, coordinate data from the Twine authoring environment, and outgoing link destinations.

In the final tool, users will be able to upload their own Twine HTML output file for parsing into a visualization (see Future Work). This initial prototype captures that use case by parsing the data from the `<tw-passagedata>` tags in a PHP script as a file is uploaded. The script then converts the relevant graph information into a JSON file that contains objects for text nodes and the outgoing links from those nodes needed to reconstruct the graph. This information currently needs to be manually saved into `data.json`, but in the final version of the tool will be moved there automatically.

---

1 The distinction between “reader” and “player” is an ongoing discussion in the field of interactive narrative. While generally games are thought of as systems of rules that yield either a desirable or undesirable outcome [7], many works of interactive narrative do not fit neatly unto this category, but still maintain the interactivity and investment in outcome that we might see in a game. For the purposes of this paper, I will use the two terms interchangeably, since hypertext narratives may take the form of games with a win/loss condition or literary works with endings that do not fall squarely into a win or loss.
4.3 Visualizing Redundancies
Replay is a goal for many hypertext authors, and understanding which nodes are revisited and which links are traversed from one playthrough to the next is incredibly useful to authors who hope to foster replay in their works.

This system visualizes redundant content by visualizing the graph structure of the piece with larger nodes for those revisited on multiple playthroughs, and thicker lines to denote links traversed on multiple playthroughs. Having a quick way to spot redundancies can help authors alter their content in the ways they find most appropriate.

4.3.1 Parsing Redundancies
My system uses a post-order depth-first search algorithm to find leaf nodes (story endings) and calculate a count of each node using the outbound links in the JSON file to traverse the graph. The descendant count of each node is then used to scale the size of the graph visualization (see 4.3.2).

Because the traversal algorithm relies on moving on to the next node once it encounters a node it has already visited, it does not detect cycles very well, and in certain cyclical cases returns false negatives and under-represents path traversal. While this might seem like a system failing at first, it actually prevents the problem of a cycle producing nodes that grow infinitely large, since some possible playthrough yields a path that traverses the cycle ad infinitum.

While some authors use cycles to aesthetic ends, many do not, and this first-pass prototype is still useful in conveying overall structure and visualizing redundancies in common cases, proving especially useful in cases of highly-connected graphs (e.g. highly non-linear narratives).

4.3.2 Displaying Redundancies
Once the graph is parsed and the correct counts obtained for each node are obtained, we pass the JSON to the Sigma.JS library, which handles rendering the graph for WebGL. The scale the size of the node according to its descendant count on a force-directed graph. This scaling means that no matter how the structure of the story looks, the Start node will always be the largest and no matter how the narrative branches, the start is always highly visible. While that might seem obvious in mostly linear narratives, which establish something like a long-tail visual, narratives from which many different paths diverge from the Start node might look more like a spider or spoke, and the highly-visible Start is indeed helpful.

4.4 Other Visual Information
In addition to size, this system relies on a few other visual markers to convey information about the narrative structure. The force directed layout offers a different visual layout to the human-authored layout of the Twine tool, and a short animation when the tool loads conveys the movement of the force-direction. The repulsion factors also mean that stories that may seem highly connected in the Twine authoring environment might be spotted in this system as mostly linear with a few branching leaf endings (compare the Twine version of the narrative in Fig 6 to the same narrative in this system in Fig 7).

Figure 6. “Not Knowing When the Dawn Will Come, I Open Every Door” by Patrick Fox [5] in Twine 2 authoring environment.

This is a case of a particularly well-laid out Twine structure; the author has clearly put great care into the visual design, and it still conveys more connectivity among nodes than the story seems to actually allow. One can imagine a narrative in which the author is not so careful and the messes than can—and do!—result from disorganized authoring.

Figure 7. “Not Knowing When the Dawn Will Come, I Open Every Door” by Patrick Fox [5] in our visualization tool.
Figure 8. Obvious endings

Figure 9. Non-obvious endings

Labels that display the name of nodes are drawn based on the current viewing size of the nodes, so they render for large nodes at default size, and are visible on smaller nodes upon zooming in. Node names may also be obtained by hovering a cursor over the nodes.

Figure 9. Node hover labels (cursor could not be captured by my capture tool and was simulated in this shot).

5. LIMITATIONS

While this project represents a helpful step toward a useful tool for Twine authors, the system in its current form has several limitations. As previously mentioned, it does not always detect cycles, and in cases of cyclical graphs may give too small of a size scale to descendants beyond the cycle.

The system also cannot handle cases of highly computational “combinatorial” narratives. Twine offers the default ability for the text of one node to be displayed in another node through the <display> macro. For example, an author could construct a narrative experienced by a player inside a single page (node) in which the text is actually combined smaller passages from many other nodes. These kinds of “include-links” are rendered as edges in the Twine interface, but are not contained in tag data in Twine’s HTML data, and thus are not easy to parse. Thus for this iteration of the project, include-links are not visualized. In cases of highly-combinatorial narratives, that is narratives utilizing many include-links, the system cannot detect children or how often a passage is experienced, thus will render the nodes as default size with no edges.

6. FUTURE WORK

While the system is still very much a prototype, several obvious next steps are readily apparent: future work hopes to incorporate parsing of include-links to more accurately depict more highly combinatorial narratives.

Adding arrows to denote link direction is also planned, but was unwieldy during this project due to poor documentation of the Sigma.js libraries. Adjusting edge thickness was difficult for similar reasons, but is planned for future implementation.

In cases where cycles cause false negatives in the descendant count, adding a factor that takes in-degree into account could also issues where a leaf is not scaled because it doesn’t have descendents even though it is accessed on multiple playthroughs (e.g. Fig 9).

Once the project is more usable to a novice user, user tests will be essential to judging its effectiveness and how the visualizations are received.

7. CONCLUSIONS

This project created a tool to more easily visualize the output of the interactive narrative authoring process, namely the structure of hypertext stories. By using force-directed layout techniques to visualize the same structure in a way different from its authoring environment, patterns in the narrative structures emerged and were more readily apparent. In my limited cases, how linear or non-linear a narrative is is immediately more visually apparent than in the Twine authoring tool, even with careful layout in Twine on the part of the author.

The size scaling of this system was effective for conveying which nodes lie on multiple paths (usually more “main” paths), but adjusting edge thickness would make the path traversal aspect more immediately apparent.

The tools has not been tested on an output file with more than 50 nodes, though hypertext narratives rarely contain more than a few hundred nodes and edges.

8. ACKNOWLEDGMENTS

I would like to thank Alex Pang for innumerable helpful suggestions and valuable feedback. I would also like to acknowledge Michael Brich, Melanie Dickinson, and Jacob Garbe who offered helpful directions and tips on resources and libraries. And finally, I would like to thank Michael Thomet and several hypertext authors in the Interactive Fiction Database community for providing test files for the tool.
REFERENCES


[5] Fox, P. 2015 “Not Knowing When the Dawn Will Come, I Open Every Door”. IFID: FC79DF8C-E3BC-45E2-8F3D-C66A530F264E http://ifdb.tads.org/viewgame?id=n5w4w7cok3a64pl


