Author Assistance Visualizations for *Ice-Bound*, A Combinatorial Narrative

Jacob Garbe, Aaron A. Reed, Melanie Dickinson,
Noah Wardrip-Fruin, and Michael Mateas
University of California Santa Cruz
1156 High St, Santa Cruz, CA
jgarbe@ucsc.edu, aareed@soe.ucsc.edu, mldickin@ucsc.edu,
nwf@ucsc.edu, michaelm@soe.ucsc.edu

ABSTRACT

As games increase in narrative complexity, challenges mount for their story designers. As authors, these designers are concerned with building an experience that responds to players in consistent and engaging ways, even when players make unpredictable story choices amongst a large group of options. Furthermore, they want those choices to have narrative impact. However, historically the task of authoring such narratives can prove daunting or infeasible. One approach to solving this problem is to use external tools to visualize story structure. This paper introduces a new set of design-time visualizations for combinatorial interactive narrative authoring. By using these visualizations during the creation of *Ice-Bound* (an interactive narrative iPad game) we were able to author content within a large combinatorial possibility space, and achieve both desired player freedom and content responsiveness. This generalized visualization strategy could also prove useful to future interactive narratives using combinatorial approaches.

Categories and Subject Descriptors

K.8 [**Personal Computing**]: Games; D.2.8 [**Metrics**]: Complexity measures

General Terms

Design, Measurement

Keywords

interactive narrative, visualization, authoring tools

1. INTRODUCTION

For modes of interactive narrative based on branching tree structures, where players choose actions from a short list, knowing when enough content has been authored to cover all sets of player interactions is as simple as verifying each tree node. However, for works which involve combinatorial narrative—or possible combinations of several different story parts—knowing when all sets of possible player interactions will still result in the display of appropriate content is a non-trivial problem. During the development of *Ice-Bound*, an iPad game where players use different combinations of story elements to create narratives, it quickly became necessary to visualize the content distribution in a way that was both intuitive and informative, allowing us to easily zero in on spaces needing more authoring.

Through the course of developing the visualization system, we learned it was necessary to fluidly move between its different information displays, and the game itself. This both provided fact-checking for assumptions made by the viz system, and helped provide different ways of looking at the content authoring challenge we faced. This assisted us in writing content efficiently, preserving both the quality and reactiveness we desired. Furthermore, it brought to light subtle issues arising from the combinatorial procedures of the system, that may have otherwise remained hidden. This suggests that tools like our visualization could generalize to other works, assisting authors in the creation of more deeply interactive narratives.

2. PREVIOUS WORK

Interactive narrative is an established form with a history that spans many different modalities and mediums. As such, writing for any possible state the player may explore in a given piece is an authoring problem that's been addressed in number of different ways as described below. However, combinatorial narratives such as *Ice-Bound* are still relatively rare, and as such require different visualizations to provide the same level of authorial overview.

The most prevalent visualization for interactive narratives is the simple graph (including trees), where nodes are individual story segments (or lexias) connected via the afforded actions of the player. Hyperfiction, for instance, typically uses the display of node connections to show the afforded action of clicking a link. These graph visualizations have been in place since the early days of non-linear storytelling with tools like Aquanet [10] and Storyspace [1], and continue to be used today in a variety of authoring tools, from Twine [5] to game modding tools like the Neverwinter Nights toolsets [4].

These sorts of graphs fill the most immediate need of authors: seeing the overarching structure of a piece, providing a way to see where paths of interaction dead end, where con-

centrations of links make certain content more likely to be displayed, or the lack of links makes it perhaps impossible. As said by Mark Amerika, author of *Grammatron*: "creating complex hypertext structures for the web is a nightmare because, after a certain point, one cannot visualize a cognitive mapping structure for a webwork that has literally thousands of screens and links" [2]. In this case, Storyspace's ability to provide a simple spatial representation of reader pathways proved invaluable to his efforts.

Another interesting aspect of tools such as Tinderbox [3] and Twine, is that the authoring interface can be heavily bound up in the visualization itself, where creating a new node in the visualization directly maps to creating a new node in the work. Semantic flags in the writing itself can create new nodes, which are immediately added to the visualization. In these sorts of environments, the authoring is done from within the data visualization of the media artifact rather than as a response to experiencing the game or text, which (through leveraging good UI design) can dramatically increase an author's ability to architect complex narrative structures.

Games that provide powerful tools for the creation of new narratives with game engines, such as the Neverwinter Nights toolsets, still hold to the graph structure for narrative visualization, if they provide one at all. The well-used tree map for branching dialogue with NPCs is still the standard, and well-suited to most forms of menu-driven interaction with game characters. But it also begs the question: with different, more sophisticated visualizations, what narratives could become feasible to create?

Another visualization tool for interactive narrative debugging is the IDE for Inform 7, a parser-based interactive fiction language, which features a view of possible traversals of the story called the "Skein." This mode also makes use of a graph metaphor, although with the graph now showing multiple *playthroughs* of the interactive narrative, which can be replayed upon changes to the narrative code to ensure they still produce the expected output. This opens up interesting diagnostic affordances to authors, allowing them to zero in on specific series of actions taken by players [7]. However, this visualization has trouble addressing IF's sheer combinatorial scale. The basic player affordance to interact via an arbitrary combination of nouns and verbs ("get lamp", "drop lamp", "light lamp") quickly becomes too large to be tenable for visualization, and would even perhaps not be considered useful. In general, it isn't sensible to author dedicated content for nonsensical parser commands, such as "eat lamp". Therefore, the easiest way to generate data for visualization falls back to playtraces, which can rely generally on players to engage in goal-directed play that avoids nonsensical parser command combinations.

Continuing in this vein, recent projects such as Playtracer [6] are making headway in providing representations that could conceivably be adapted to provide narrative diagnostic strategies for projects where readers have a high level of expressive affordance. Playtracer generates graph structures where certain nodes are defined as goal states, and each choice or state the player can induce in the system is a node connected by the action to the previous state. The distance of nodes from each other is calculated using an abstraction of the states' dissimilarity from each other. The size of nodes is the number of playtraces that involve that state.

There have also been playthrough visualizations used for interactive narratives to better understand the potential space explored through interaction. For *Prom Week*, a game making use of cutting-edge persistent social state modeling to create dynamic experiences for each player, visualization served a critical evaluative purpose in exposing how quickly individual playthroughs become unique [9]. These same techniques were used to demonstrate a similar property in *Façade*, an earlier work famous for its expressive player affordances.

But while these sorts of visualizations proved illuminating for certain aspects of these works, they were used after the authoring was completed. In contrast, *Ice-Bound*'s visualization system was created to provide a continuous form of feedback about the narrative state, helping to target authoring of content to cover as many possibilities as possible.

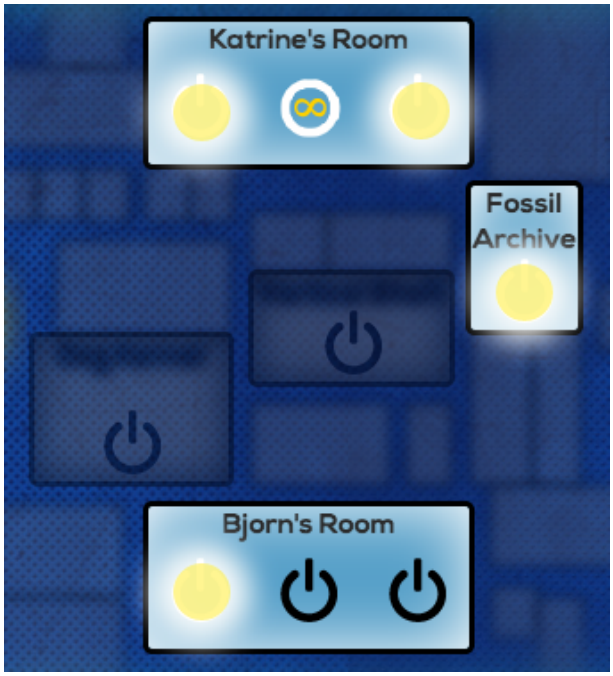

Figure 1: Each *Ice-Bound* level has around eight to twelve sockets and two to four movable lights. Here the player has activated four sockets by using all four lights, while a fifth socket (in the center of Katrine's Room) is always on regardless of the player's actions. Four other sockets are inactive.

3. LIVE DEMO

A live demo of the visualization system can be accessed at http://ice-bound.com/viz.

4. *ICE-BOUND*: THE NARRATIVE SYSTEM

A reader plays *Ice-Bound* by crafting sets of stories from component texts. Each story is presented as a set of ***sockets*** arranged on a map and corresponds to potential narrative ***symbols***, such as a character trait, or a dramatically charged location (Figure 1). By positioning a limited supply of ***lights*** on those sockets, the player activates some subset of the story sockets and sets the corresponding symbols to ***active***. The combination of active symbols triggers various ***events and endings*** which narrate a story arising from the symbols

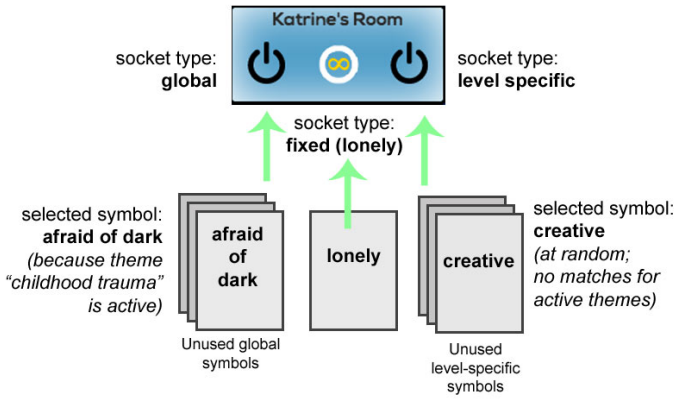

Figure 2: The build process for a room

chosen by the player. The player eventually selects one of the ***endings*** as the preferred one, locking in the story, and continuing on to the next level [8].

Each socket has a type which limits the pool of symbols that can be assigned to it. As seen in Figure 2, the socket on the left is global, meaning it takes symbols from the global symbol pool. The middle socket is fixed, requesting a specific symbol. The rightmost socket is level-specific, meaning only symbols in the pool authored for this particular story can appear here. In addition, the middle symbol is marked "always on," meaning it will always be active (which allows the author to express fixed components of the potential story; in this case, specifying that this story will always involve Katrine being lonely). If the player was given at least two lights, this room would allow for four possible combinations of active symbols: (lonely), (afraid of dark | lonely), (lonely | creative), and (afraid of dark | lonely | creative). The selection of certain symbols for the sockets is also subject to filters concerning those symbols' themes. This will be discussed later in Section 5.1: Theme-driven Filters.

5. SYMBOL COMBINATIONS AND POSSIBILITY SPACES

Ice-Bound's narrative possibility space is combinatorial in two distinct ways:

1. **Build-time combinations.** When a player starts a level, the system first builds it by assigning symbols to sockets from a wider pool of available content. The same level might be built with different symbols based on the way the player has resolved prior stories, or simple random variation. The total number of possible combinations for a level (given every possible build) with our current preliminary set of symbols ranges from 500 to 5000.

2. **Run-time combinations.** While playing a given build of a level, the player can explore the combinations of active symbols made possible by the number of lights they are given. The number of unique combinations the player can activate typically ranges from 30 to 70.

The large number of build-time combinations allow authors to make the selection of content for sockets sensitive to player choices through theme-driven filters. The number of run-time combinations means that players are presented with a wide array of choices, which gives them the ability to explore a highly varied story landscape and construct a fictional experience that resonates with the themes they find compelling in *Ice-Bound*.

Each of these plays an integral role in the richness and responsiveness of the game. However, two distinct authoring challenges arise from this structure: 1) ensuring all build-time combinations produce a satisfying narrative possibility space, and 2) ensuring most player-arranged run-time combinations produce a satisfying story.

The definition of "satisfying" is an author-selected metric based on playtesting feedback. The targeted goal for content authoring was that for any given combination using all available lights for the level, at least three events and two endings would be activated. Fewer than that, and the system feels unsatisfying to explore (and if no endings are present, stories become impossible to resolve).

In the process of authoring more symbols for global sockets, which radically increases the number of build-time combinations on every level of the game, it became apparent early in the design process that a visualization tool was sorely needed to keep on top of the many different combinations made possible by these additions. This assisted us in our efforts to ensure that there was always a "satisfying" number of events and endings triggered by them.

5.1 Theme-driven Filters

Theme-driven filters are the selection strategy that helps "cash out" the combinatorial richness of *Ice-Bound*. Each symbol, event, or ending in *Ice-Bound* is tagged with one or more of 27 descriptive themes. These themes are activated every time players confirm endings for stories they've formed from activated symbols and events on a given level. These activated themes affect symbol selection for the building of subsequent levels through theme-driven filters.

The intent with this strategy is to allow the system to react to player choice in order to more often offer story options the player wants to explore. By running possible level symbols through a theme-driven filter—such as one centering around "addiction", "self-realization", or "loss of innocence"—the build-time combinations will start to more prominently feature themes the player has demonstrated an interest in.

6. VISUALIZATION DESIGN GOALS

The goal of the visualization system is to highlight where symbol combinations do not trigger at least three events and two endings, our self-selected minimum requirement for content. Symbol combinations where this is not the case need more authoring. This metric is also parameterized, such that we can make it stricter later on in the authoring process, to provide finer feedback. When considering where new content would be most useful, the primary consideration is discovering a combination of symbols to be used as a precondition that fills existing holes in the possibility space. The visualization needed to provide information to allow authors to intuitively grasp how to accomplish that.

Ice-Bound's engine is written in Javascript, so the visualization tool also needed to run on the same code base. This was necessary to minimize errors that might be introduced

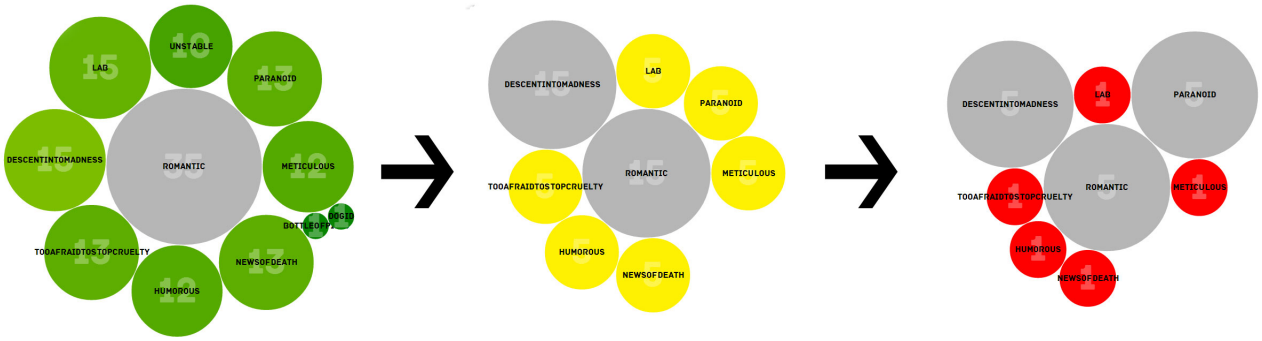

Figure 3: The combination browser, as the viewer progressively zeros in on problem combinations involving the symbols "romantic", "descentIntoMadness", and "paranoid".

through re-implementation of game procedures, and also to ensure any future changes introduced to those game procedures would be faithfully reflected in the visualization.

Ideally, the tool would both provide a window on the possibilities for an example level build (where all the symbols have been selected for sockets) and reveal possible level builds where there were not enough events or endings written or triggered, if certain symbols were involved. Furthermore, showing us which specific symbol combinations give rise to these states would give strong indicators for authoring appropriate preconditions to new events and endings in order to fulfill our goals.

7. DESCRIPTION OF SYSTEM

The system was designed in two parts: ***a combination browser*** and a ***level profiler***. The browser is concerned with broadly classifying combinations which need content, abstracted from a specific level build. The profiler gives a detailed view into the combinations within a specific level build. In operation, the combination browser is used to provide a list of explicit symbol combinations where content is sparse. The user can then click those combinations to build a level containing the given symbols, and through the level profiler, see how much content is needed to fix the scarcity issue.

7.1 Combination Browser

The combination browser engages all possible symbol activations for a level by permuting every unique combination of symbols from the global pool and level-specific pool, equal in length to the amount of available lights on the level. It then uses the game logic to simulate activating each symbol, rebuilding the level if necessary to contain those symbols. If the level cannot be built with a particular combination of symbols, it discards the combination as illegal. For example, some symbols have preconditions that prevent them being chosen if another symbol is present in the build. In this case, the system quickly discards the combination after the level building function determines that those symbols are impossible to have together. After activating the symbols on this simulated level, it records the number of events and endings activated by each symbol combination.

Once the system has constructed this exhaustive list, it decomposes the combinations such that it has a record of how many under-authored combinations each symbol is part of. For example, the problematic symbol combinations A, B, C and A, C, F would yield that both A and C are part of two problematic combos, while B and F are only part of one.

This analytical strategy was chosen because the solution to the problems highlighted by the viz is to write more events and endings, with preconditions containing specific symbols. Thus, a symbol being present in more than one combination that is not adequately covered by content is a good candidate for a precondition in future content.

This process of data formulation for the visualization happens quickly enough to make it useful for making small changes and seeing how that affects the possibility space. Furthermore, once formulated, the data in the visualization can be quickly manipulated, which greatly increases its utility.

Through a bubble cluster diagram (Figure 3) the browser displays each symbol. The size of the bubble corresponds to the number of problematic combinations it was involved in. Each bubble can be clicked to tell the system to apply the same visualization methods on the subset of combinations only involving the selected symbol. Doing this resizes the other bubbles in the visualization, so their new size corresponds to the number of times they are involved in a problematic combination with the new set of selected symbols. If a different bubble is clicked, the values are again updated. This allows the viewer to interactively evaluate and discover problematic symbol combinations.

There is a specific tension in authoring goals when trying to patch content holes. On one hand, the more combinations an authored event or ending hits, the more effective it is at patching a hole. However, if it is also being activated for combinations which do not need more events and endings, it is potentially diluting that space. In the game itself, if a combination triggers more than five events or three endings, we truncate the list in order to keep the scope of the stories presented to the player from becoming overwhelming. Therefore, the closer we can keep all combinations to that amount, the higher the perceived causal link for the player from their actions to the reaction of the system.

To reflect these tensions, a circle color scale from green to red was adopted to show the ratio of combos needing content versus not needing content. Red circles have the highest ratio of content-less combos, and green circles have the least. This is necessary in order to discourage the authoring of content with preconditions that trigger for combinations that don't need it. For example, while authoring an event

with preconditions that trigger for every combination in the game would technically address every combination needing content, it would also appear for many combinations that didn't need additional events and endings. Also, from a narrative design standpoint, the more targeted an event or ending is to its precondition symbols, the better. It leads to content that strongly correlates with the player's selection, communicating that their choices are having a real effect, and that they have agency over the story being formed. Events or endings that show up for many different combinations are also more likely to conflict in some way with other content that is selected.

This means that the author can quickly browse to symbol groups that both have a high content need in their combination sets (by selecting circles with redder color) as well as representing a large number of the total combinations needing content for the level (by selecting circles which are larger).

Because the specifics of this are difficult to grasp, there is an additional info panel that displays the same information as the color and size, using text and tooltips (Figure 4). The color, which is determined by the ratio of combos needing content, is reflected as a pie chart (C) with explicit text (B). The size of the circle is also shown numerically (D).

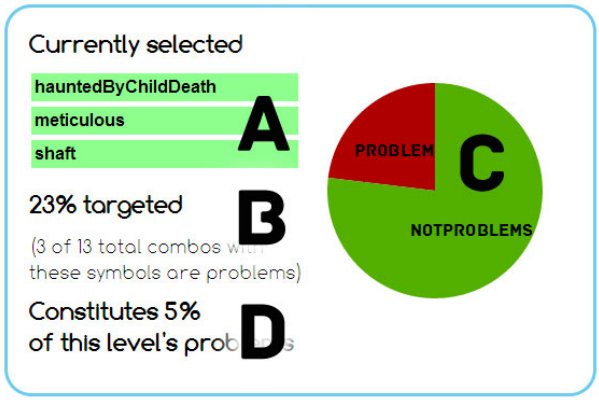

Figure 4: The detail panel in the combination browser, showing the selected symbols (A), and the percentage of their combinations which need content (B), also represented by the pie chart (C). (D) shows how many of the total combinations needing content the current selection is involved in.

While providing a good top-down view of the content distribution as a whole, the combination browser does not show the content distribution within run-time combinations of a particular level build. This gives rise to a blind spot in authoring considerations. If the combinations lacking content are concentrated in a few specific builds of a given level, that is a bigger problem than a low number of problems that persist across all possible level builds. For example, if only 1 out of 35 content-lacking symbol combinations is present in a level build, it isn't as glaring a problem as all 35 problems being present in one level build, due to the fact that a given level build can have around 30-70 combinations. In the first case, only 1.4 - 3% of the player's possible chosen combinations are lacking. In the second case, it's closer to 50 - 100%.

To address that, we needed a visualization dealing with specific level builds, accessible via the combination browser. This is accomplished through listing out the symbols currently being inspected by the viewer. If the symbols "romantic", "descentIntoMadness", and "paranoid" are selected, the list contains every content-needing combination containing those symbols, with a link to "build". When clicked, the system builds the level containing those symbols, and displays the second visualization tool: the level profiler.

7.2 Level Profiler

The level profiler (Figure 5) was the visualization most used during the initial round of authoring for *Ice-Bound*. It displays the run-time combinations of a given level build with a given set of symbols in the form of an annotated stacked bar graph. Each bar represents a particular set of active symbols, with the height corresponding to the total number of events and endings it activates.

The bottom portion of the bars represents events, and the upper portion endings. This provides an easy way to see which combinations do not trigger enough content. The coloration of these bars is either green, yellow, or red. This is dependent again on the two combination minimums we established for a "satisfying" story: a combination should trigger at least three events and two endings. If a combination satisfies both of those, it is green. If it only satisfies one, it is yellow. If it satisfies none, it is red.

Below the graph are a list of the symbols present in the level, as well as every event and ending that can be logically triggered by those symbols. Hovering over a given bar in the chart highlights the symbols, events, and endings activated in the lists below. Additionally, each item on the lists below the chart can force a re-sort of the chart upon being clicked. This means if authors want to see content authored for a specific symbol, event, or ending, they click on it, and the graph re-sorts so that all combinations involving that move to the left.

This proved enormously useful in highlighting problems that weren't readily apparent from the complex interactions of preconditions for various events and endings. Red bars indicating problem sets of active symbols can be examined to look for common elements, indicating there are not enough events and endings activated by those elements. Using this tool, one can quickly see what the combinations that need more content have in common, and get a feel for how much content is currently available to the reader at run-time.

8. DEVELOPMENT HISTORY

Of the two, the level profiler was actually developed first. Previous to this, we would play-test our levels after an arbitrary period of authoring, and add more events and endings if we seemed to be encountering content gaps consistently. But once this tool was completed, we were able to profile level builds and quickly cycle through them. This led to rapidly identifying several instances where there were too few events or endings triggering, which would have taken an infeasible amount of time to find by hand. This led to the authoring of more needed events and endings, and a much more well-rounded content base.

However, since the build-time combinations of symbols can number in the thousands, that means statistically we would have to re-build using the level profiler an infeasible number of times to be completely certain we had seen the total possibility space with all possible symbols. Therefore,

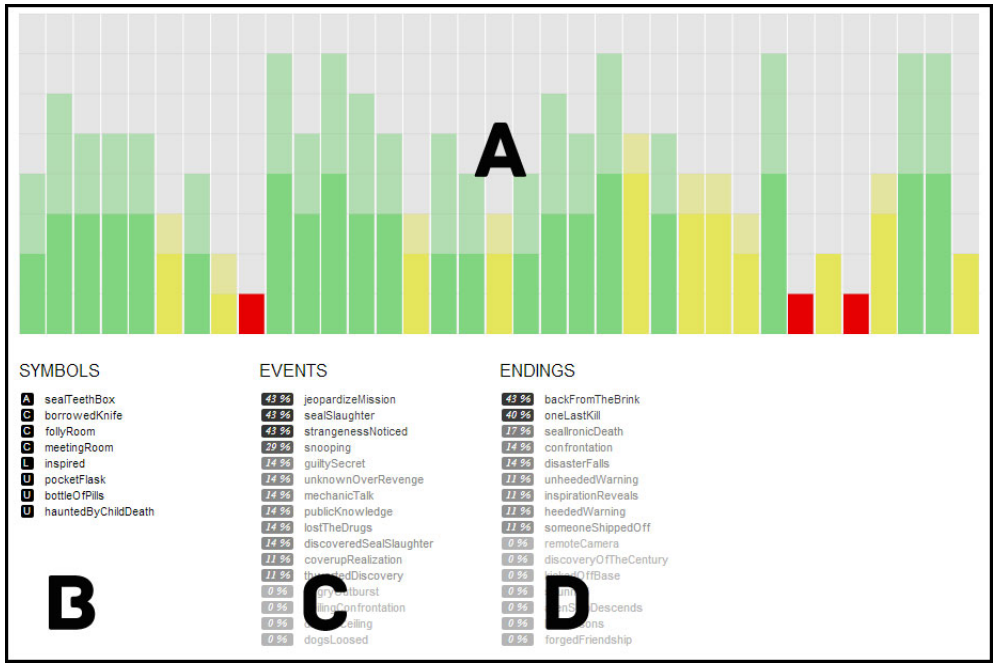

Figure 5: The level profiler for Ice-Bound, showing all possible run-time combinations of active symbols for a single build of a level. Red bars indicate content-sparse combinations.

the combination browser was created to provide a much-needed higher level view, balancing the level profiler, which grounds out those findings with specific level builds.

9. DESIGN CHALLENGES

The biggest technical challenge in designing our tool was ensuring the simulated builds of the levels were in fact going through all the steps the real game used, in order to reduce false positives. Because of *Ice-Bound*'s precondition complexity, tight integration between the combination browser, the level profiler, and the game itself were needed to provide oversight for the visualization data.

A large conceptual hurdle in the design of this visualization was how to simply communicate the complex interplay between gaps in the possibility space, and the delicate balance between specificity and effectiveness in authoring future content. In earlier versions of this visualization we provided explicit numbers and percentages inside each circle, but found that it was visually overwhelming, and slowed the process of comparison when trying to determine which circles represented symbols needing the most content.

A later iteration also translated the pie chart in the detail panel (Figure 4) to the circles themselves, creating a sort of "pie chart cluster diagram", but in trying to present slight differences it proved ineffective, and still overwhelming. Thus, color was substituted for that metric. With this change it proved much easier to distinguish fine differences, while reserving the pie chart as a further detail allowed the communication of more specific data if needed.

10. FUTURE WORK

With the level profiler alone, we were able to significantly reduce content-sparse combinations down to roughly 3%. However, content authoring for *Ice-Bound* is still underway, with two of the proposed eight levels finished, and completion estimated by summer of 2014. So far, there are thirty symbols, forty events, and thirty endings between the two levels. The largest build-time combination space so far is roughly five thousand, but it is expected to radically increase as more levels are created. It was this combinatorial expansion that prompted the creation of the the combination browser, so that as the project continues we can expect to be able to keep on top of problematic combinations.

Given this, there are a series of improvements planned for the tool to increase its utility for future content creation. For the combination browser, there is currently no way to see how many events or endings each symbol combination is lacking, only whether or not it falls outside our specified parameters. If we incorporated this information into the tool, it would more faithfully reflect the authorial work needed. Combinations currently represented by small circles may actually need more content, and large circles may not need that much more content to be acceptable. This raises the question too of whether an additional visual strategy could be used to highlight combinations which can be "easily" solved through the addition of a few events or endings, versus combinations which need many. The ability to filter the display to focus on events vs. endings could also help focus the visualization to the particular authorial task the viewer is interested in undertaking.

Also, compared to a level-specific symbol, a given global symbol has a much lower probability of being cast in a level, due to the relatively larger pool it is chosen from during the level build. In comparison, level-specific symbols have a higher likelihood, due to their consistently smaller pool. If this disparity was factored in to the visualization, one would see level-specific symbol problems emphasized over global symbol problems, which would help prioritize authoring. This would be yet another analytical lens, which would

allow us to more accurately fix content gaps that are statistically more likely to turn up than others.

A feature currently finishing development is the ability to quickly add virtual events and endings from within the viz, so that authors can try out a variety of preconditions that address content holes, but still satisfy the artistic goals for player experience. This would move it closer to the idealized state of the tool, where structurally informed authoring can simply take place within the tool itself, while providing real-time feedback on how the symbols, events, and endings are affecting the narrative space.

A second authoring concern needing to be addressed is the incorporation of symbol combinations that are *over-authored*, or trigger more than five events and / or three endings. Currently the game selects a maximum of five events and three endings from those triggered by symbol combinations, based on our authoring metric. The system will select events and endings that are more thematically apropos to the player, yet it is still an issue (albeit a more hidden one) if a certain symbol combination activates many more events and endings than that. Providing a way to show those sorts of problem combinations could help us zero in on potential situations where the story may seem to lack cohesion or react too generically to player choices.

11. IMPLICATIONS FOR OTHER WORK

This visualization strategy could be useful for other narrative games in development as well, in order to expose unintended consequences from combinations of assets, be those characters, story events, or procedures. The "symbols" as used in *Ice-Bound* to create the circles could potentially be replaced with plot items in the player's inventory, active quests, or conversation points with plot characters. With further game content operating off those types of precondition triggers, it would allow authoring of narratives that can incorporate these complexities into their structure, and allow authors to accurately diagnose both the scope of the authorial burden, and what game states still require authoring.

12. CONCLUSION

The process of authoring *Ice-Bound* so far has driven home the importance of having a robust visualization tool to assist in giving creators an accurate picture of content distribution from the outset of a project. These sorts of strategies are needed to stay on top of the possibility space, as well as prototype the effect of adding certain content. Without such tools, creating a high-quality narrative becomes quickly infeasible, with much wasted effort and mis-directed energy.

It is our opinion that experiences like *Ice-Bound*, which bind the story tightly to the mechanics of the game itself, are an important direction to pursue in games. But these sorts of works will only become more prevalent if more sophisticated visualizations are available to augment authorial tools and assist in their writing. The visualization tool for *Ice-Bound* provides a window into a possible form that they may take, and the sorts of narrative authoring potentially enabled, by increasing the ability and reach of the authors.

13. ACKNOWLEDGMENTS

Ice-Bound was developed during the 2013 SPIN Studio, with generous support from the UC Santa Cruz Center for Games and Playable Media.

14. REFERENCES

[1] M. Bernstein. Storyspace, June 2007. `http://www.eastgate.com/storyspace/index.html`.

[2] M. Bernstein. Storyspace and the making of grammatron, June 2007. `http://www.eastgate.com/storyspace/writing/Amerika.html`.

[3] M. Bernstein. Tinderbox, Dec. 2013. `http://www.eastgate.com/Tinderbox`.

[4] Bioware. Guide to the foundry- the official neverwinter wiki, Dec. 2013. `http://nwn.wikia.com/wiki/Toolset`.

[5] C. Klimas. Twine, Dec. 2013. `http://twinery.org`.

[6] Y.-E. Liu, E. Andersen, R. Snider, S. Cooper, and Z. Popović. Feature-based projections for effective playtrace analysis. In *Proceedings of the 6th International Conference on Foundations of Digital Games*, pages 69–76. ACM, 2011.

[7] A. Reed. *Creating Interactive Fiction with Inform 7*. Course Technology PTR, Boston, 2011.

[8] A. Reed, J. Garbe, N. Wardrip-Fruin, and M. Mateas. Ice-bound: Combining richly-realized story with expressive gameplay. In *Foundations of Digital Games*. FDG, 2014.

[9] S. Sali. *Playing With Words: From Intuition to Evaluation of Game Dialogue Interfaces*. PhD thesis, UC Santa Cruz, January 2012.

[10] F. M. Shipman III and C. C. Marshall. Spatial hypertext: an alternative to navigational and semantic links. *ACM Computing Surveys (CSUR)*, 31(4es):14, 1999.