Amorphous Interpretations: Diverse Narrative Generation for Open-Ended Simulation Environments*

Dipika Rajesh[1], Julian Togelius[2] and M Charity[3]

[1]*University of California Santa Cruz, Santa Cruz, CA, USA*
[2]*New York University, Brooklyn, NY, USA*
[3]*University of Richmond, Richmond, VA, USA*

Abstract

We introduce Amorphous Interpretations – an evolutionary narrative generation system capable of interpreting events in an open-ended simulation to create thematically diverse and cohesive stories. We use the event log files from four human-designed environments from the abstract Amorphous Fortress engine as our domains to generate artificially "interpreted" stories. The raw event logs are sifted to extract significant narrative events that are then reduced to a grammar to systematically modify thematic elements of the story. By applying two different Quality-Diversity algorithms, and by restricting the modification and randomization of certain story elements, we discover that this system is capable of generating a wide range of stories from just a single event log.

Keywords

artificial life, narrative generation, simulations, open environments, quality diversity algorithms

1. Introduction

Often with simulation environments, we as humans invent "stories" for the agents and characters within them. Under the hood, the bits and pixels of the characters and their actions only have meaning with respect to the set of rules and mechanics defined by the environment.

As we interact with these simulations, each person has the ability to interpret their own version of the same story that is both unique and contextually relevant to the events that transpire in the simulation. Each person, in turn, provides their own perspective or viewpoints that make for interesting or novel re-tellings. Is it therefore possible to explore and evaluate the expressive range and quality of the themes created from artificially generated stories in open-ended environments?

This paper introduces Amorphous Interpretations – an evolutionary generative system capable of creating unique stories from a simulation environment's event log. We use the Amorphous Fortress environment, modeled after video game mechanics and non-playable character interactions, to generate interpretations of human-designed but AI-controlled environments. We find that the system is capable of creating thematically diverse and coherent narratives, automatically assigning new and diverse thematic meanings to fortresses that don't necessarily contain human-designed themes. This system is also capable of creating new stories from environments that already contain an established theme, thereby creating an "alternate universe" of the same world. With this system, our goal is to artificially generate new and interesting stories from a simulation environment in order to attain a human level of fluidity and creativity for a story retelling or interpretation.

Joint AIIDE Workshop on Experimental Artificial Intelligence in Games and Intelligent Narrative Technologies, November 10-11, 2025, Edmonton, Canada.

dirajesh@ucsc.edu (D. Rajesh); togelius@gmail.com (J. Togelius); mcharit2@richmond.edu (M. Charity)
https://julian.togelius.com/ (J. Togelius); https://mastermilkx.github.io/ (M. Charity)

2. Background

2.1. Simulations and Emergent AI Behavior

Simulation environments allow for the observation of interactions between artificial agents that simulate real-world scenarios and events. In games and AI research, simulations and Artificial Life [1] have been extensively studied to discover connections and patterns in AI behaviors. Charity et al.'s SimSim environment [2] evolves furniture arrangements in houses based on the Sims games to find novel designs while keeping the player alive and satisfied. Suarez et. al's NeuralMMO [3] environment evolved reinforcement learning (RL) agents to survive in an environment modeled after massively multiplayer online role-playing games (MMORPGs.) OpenAI's game environment [4] trained agents to adversarial compete in a game of hide and seek. Park et. al's Smallville environment [5] allowed for the simulation of human behavior and multi-agent interaction and communication in a small virtual community.

All of the systems also led to unexpected and novel behaviors from the agent interactions that were not originally defined for the system and emerged over time: SimSim generating houses consistently replacing the beds with coffee machines; developed agents in NeuralMMO learning to seek out and "hunt" weaker agents; agents in the hide and seek environment learning to glitch the environment to successfully beat the adversarial team; and agents forming relationships, making plans, and gossiping with each other over time in the Smallville environment. However, these discoveries were annotated by the authors and developers of the environments. There is a lack of academic work of the system itself automatically provide reasoning or form a narrative about these behaviors.

2.2. Story Sifting

Non-deterministic simulations, by their very nature, create a large number of event logs that must be used to generate narratives. Story Sifting [6] explores *sifting* as a process of combing through a large number of logs to extract "interesting" ones - carefully selecting a meaningful subset of the logs that carry narrative significance. We are interested in Story Sifting approaches as we need to systematically comb through large logs of abstract representations that cannot be extrapolated into narratives directly. Kremenski et al. [7] explore a domain-specific system that automatically synthesizes sifting patterns based on user-provided events.

However, these approaches benefit from interpretable and explainable simulation mechanics with clear narrative significance. For example, Centrifuge [8] proposes a visual Story Sifting system that matches queries against a simulated world to identify interesting characters and events, such as finding characters explicitly tagged as "villains." This is possible because the simulation provides interpretable character representations, in contrast to the more abstract representations used in Amorphous Fortress. With Amorphous Interpretations, we automate the Story Sifting process by using the inherent game mechanics defined by the Amorphous Fortress engine, without the presence of explicit labels.

2.3. Narrative Generation

There has also been extensive research on generating narratives using a number of different methodologies. Talespin [9] is a seminal project in this space where the characters in the simulation pursue goals using plans that are built by the narrative generation. A significant area of research in Narrative Generation explores Narrative Planning [10], which uses causal links and domain knowledge to explore character motivations. When it comes to abstract simulation environments, some Narrative Generation systems are formed from human annotations. Hjaltason et al.'s experiment [11] asked participants to annotate the interactions between agents and form a narrative based on a custom made action game environment. This system was inspired by Heider and Simmel's experiment [12] which asked participants to form a narrative after watching a stop-motion animation video that used simple shapes. This produced narratives with human-like attributes and behaviors applied to the shapes. Other works have formed narratives using evolutionary methods. McIntyre and Lapata's system [13] which generated stories and plots by evolving grammars and replacing entities from a corpus of children's stories. García et al.'s [14] system evolved grammars and finite-state machines defining character backstories and behaviors to interact with each other in a simulation environment to see if a particular story structure could be found from the event logs. Similarly, Caves of Qud [15] uses a character history generator that uses replacement grammar and state machines in order to generate historical events in the game and then interpret them. Our work seeks to automatically evolve and discover a diversity of stories, plots, and themes from a log simulated agent interactions in an environment that carry no inherent human-interpretable meaning.

3. Methods

3.1. Amorphous Fortress Environment

For this experiment, we use the Amorphous Fortress (AF) simulation environment [16] as the testbed for this story generation system. Entity classes in the Amorphous Fortress environment are defined using finite state machines – where nodes define the action space of an entity from the class and edges define transition conditions. The possible entity action nodes are defined in Table 1. Running a fortress, or contained simulation environment containing these agents, in the Amorphous Fortress engine produces an event log of interactions between entities in the system. These events are produced based on the action node definitions an agent performs at a given timestep. We use the event logs produced by the fortresses to form the core foundation of the narrative generation system.

Action Node	Definition
idle	*the entity remains stationary*
move	*the entity moves in a random direction (north, south, east, or west)*
die	*the entity is deleted from the fortress*
clone	*the entity creates another instance of its own class*
push (c)	*the entity will attempt to move in a random direction and will push an entity of the specified target character into the next space over (if possible)*
take (c)	*the entity removes the nearest entity of the specified target character*
chase (c)	*the entity will move towards the position of the nearest entity of the specified target character*
add (c)	*the entity creates another instance from the class of the specified target character*
transform (c)	*the entity will change to a different entity class - thus altering its FSM definition entirely*
move_wall (c)	*the entity will attempt to move in a random direction unless there is an entity of the specified class at that position - otherwise it will remain idle*

Table 1
Entity FSM action node definitions. These actions were modeled after actions commonly performed by characters in video games.

3.2. Log Sifting

In order to create a narrative that is easy to follow, we intended to capture a relatively short subset of the logs. For these experiments, we limited the number of logs to 30. However, it takes several time steps for the entities in the fortresses to take a wide variety of actions that capture a narrative. Since each action at a particular time step generates a log, the raw event logs generated by the fortresses are often very long. Instead of randomly selecting 30 logs, we designed a Log Sifting system that extracts narratives in a meaningful way.

The system iterates through the raw event logs generated by the fortresses and systematically selects logs based on patterns, rare events, and random chance, prioritized in that order. The patterns are hand-authored based on the mechanics offered by the AF simulation system. Three main patterns were identified after analyzing the logs and the system mechanics: Predator-Prey, Gatherer, and Treasure Hunter. The Predator-Prey pattern identifies situations where one entity *chases* another entity and sometimes *takes* them. The Gatherer pattern identifies situations where one entity *takes* multiple other entities. Finally, the Treasure Hunter pattern identifies situations where one entity *moves* around considerably and *takes* only one other entity.

The Log Sifting system chooses the method of selecting logs probabilistically. The pattern extraction system has the highest probability of 0.5 to give preference to the patterns that are afforded by the system's mechanics. The rare events are identified by creating a weighted probability distribution over the actions in the raw AF log, where less frequent actions are assigned higher weights. For example, the *move* action happens most frequently in the logs and is therefore assigned the least weight. The Log Sifting system is least likely to pick the log with the *move* action if it is sifting based on rare events. The probability of rare events being extracted by the system is 0.3. Finally, the Log Sifting system has a 0.2 chance of selecting the logs entirely randomly, one at a time. This process is repeated until the desired number of logs is reached and the final sifted logs are ordered by time step at which they occur during the simulation.

3.3. Grammar Extraction and Word Replacement

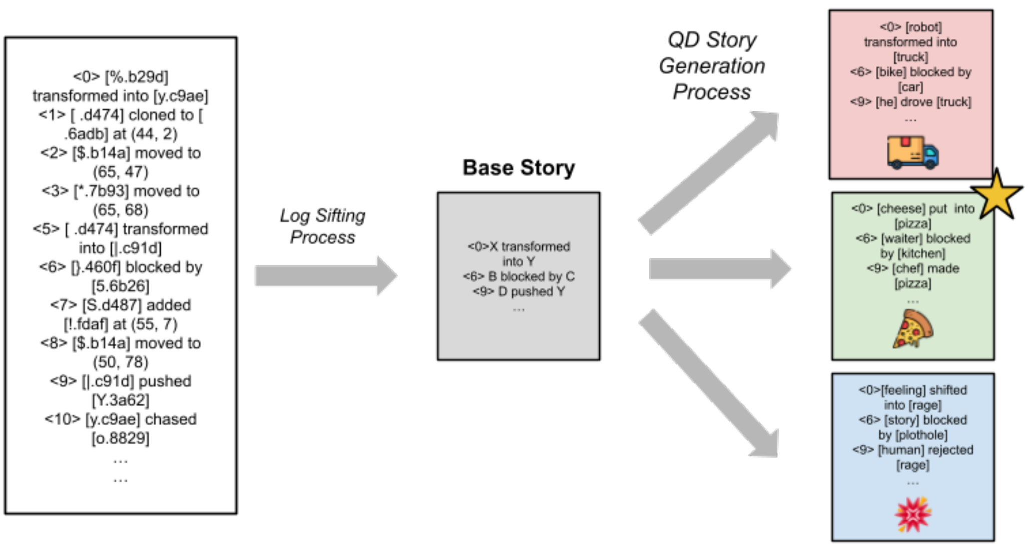

```
<0> [%.b29d]
transformed into [y.c9ae]
<1> [ .d474] cloned to [
.6adb] at (44, 2)
<2> [$.b14a] moved to
(65, 47)
<3> [*.7b93] moved to
(65, 68)
<5> [ .d474] transformed
into [|.c91d]
<6> [}.460f] blocked by
[5.6b26]
<7> [S.d487] added
[!.fdaf] at (55, 7)
<8> [$.b14a] moved to
(50, 78)
<9> [|.c91d] pushed
[Y.3a62]
<10> [y.c9ae] chased
[o.8829]
...
...
```
```
<0>X transformed
into Y
<6> B blocked by C
<9> D pushed Y
...
```
```
<0> [robot]
transformed into
[truck]
<6> [bike] blocked by
[car]
<9> [he] drove [truck]
...
```
```
<0> [cheese] put into
[pizza]
<6> [waiter] blocked
by [kitchen]
<9> [chef] made
[pizza]
...
```
```
<0>[feeling] shifted
into [rage]
<6> [story] blocked
by [plothole]
<9> [human] rejected
[rage]
...
```

Figure 1: A diagram of the conversion from the original Amorphous Fortress logs, to the sifted form, then finally to 3 generated stories.

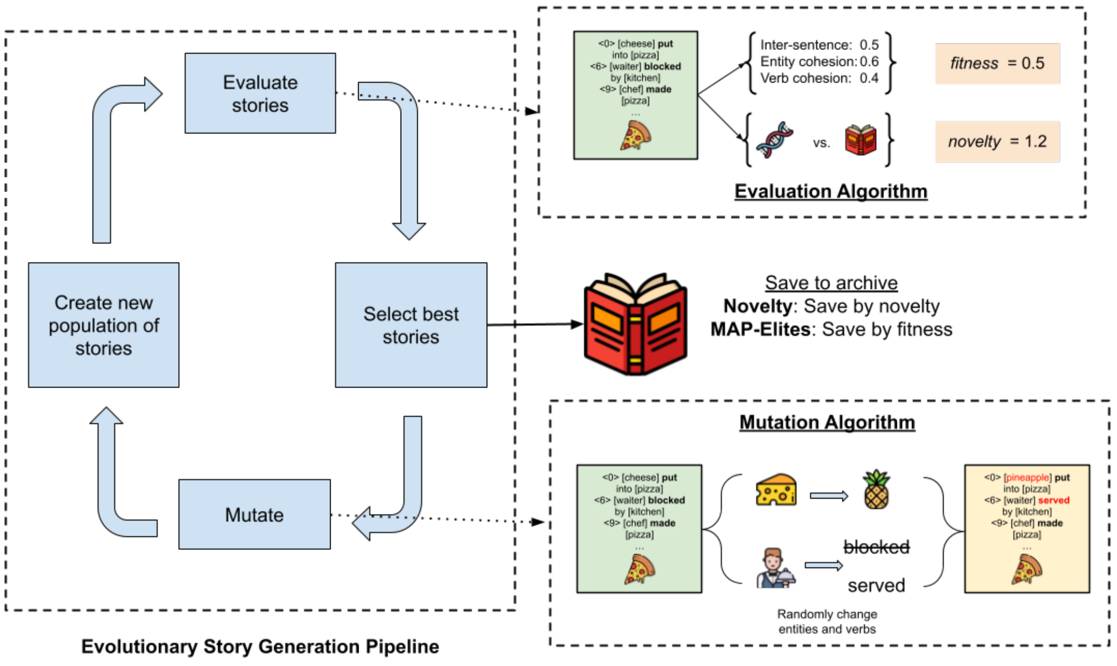
The sifted event logs from the fortress simulations are reduced to a grammar format for simple replacement of entities and their actions. Entities, represented in the log as their entity class ASCII symbol followed by an entity ID number, are replaced as generic nouns. Their actions, defined from their FSM action node, are replaced as generic verbs. Using this grammar of the event logs, the system replaces the nouns and verbs with relationship definitions taken from the ConceptNet semantic network[1]. For this experiment, we reduced the entire ConceptNet database to only use NOUN-VERB-NOUN relationships to match the grammar structure of the logged events. We use a grammar format to maintain the already abstract nature of the Amorphous Fortress environment. The pool of nouns and verbs were extracted using the 'UsedFor', 'CapableOf', and 'ReceivesAction' graph connections along with the spaCy library [2] for identifying nouns, compound nouns, proper nouns, and verbs. All verbs were also changed to past-tense form using spaCy to create a consistent and grammatically correct story-telling narrative

[1]https://conceptnet.io/
[2]https://spacy.io/

from the generation process. Figure 1 illustrates the process of converting the original Amorphous Fortress log file, into the base grammar form, sifted, and then into 3 different randomly made stories using the NOUN-VERB-NOUN relationships in the modified ConceptNet database.

3.4. Story Generation

This experiment uses the evolutionary algorithm to generate unique and “interesting“ stories from the grammar-reduced log files. Figure 2 shows the entire genreation process in the experiment.



Evaluate stories
Select best stories
Mutate
Create new population of stories
Evolutionary Story Generation Pipeline
<0> [cheese] put into [pizza] <6> [waiter] blocked by [kitchen] <9> [chef] made [pizza] ...
Inter-sentence: 0.5
Entity cohesion: 0.6
Verb cohesion: 0.4
vs.
fitness = 0.5
novelty = 1.2
Evaluation Algorithm
Save to archive
Novelty: Save by novelty
MAP-Elites: Save by fitness
Mutation Algorithm
~~blocked~~
served
<0> [pineapple] put into [pizza] <6> [waiter] served by [kitchen] <9> [chef] made [pizza] ...
Randomly change entities and verbs

Figure 2: A diagram of the evolutionary pipeline and story generation for the Amorphous Interpretations system.

First a *main character* is chosen for the story. This main character is defined based on the entity with the most interactions in the log original file. All other entities in the fortress are themed around this main character by choosing associated nouns defined from the custom ConceptNet dataset. For example if the main character is chosen to be a *cat*, the other entities may be defined as *dog*, *mouse*, *human*, *milk*, *bird*, or *house*. If there are multiple entities from the same class definition (i.e. share the same symbol in the log), the other entities are defined with the same noun but with numbers after the name (e.g. *mouse*, *mouse2*, *mouse3*, etc.) For each entity (including the main character entity) their grammar verbs are replaced with verbs associated with the entity. Using the previous example, the *dog* entity may have the possible (past-tense) verbs of *ran*, *barked*, *ate*, *buried*, *chased*, or *bit* for its interactions.

3.5. Evolutionary Search

As defined for most evolutionary algorithms, the evolved artifact for this system is the story generated from the base grammars extracted from an Amorphous Fortress log.

3.5.1. Genome Representation

The genome representation for the stories is the vectorized representation of the list of noun-entities and order of the verbs used as associated with the event logs in the story. We use “all-MiniLM-L6-v2“ model from the HuggingFace sentence transformer library[3] to encode this list. This is to encourage thematic diversity when generating the stories from the same bsae log file.

[3]https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2

3.5.2. Mutation

The mutation function for each artifact story selects each noun-entity and with some random chance changes the representation. If the noun-entity is selected to be changed, all associated verbs are changed as well. If the entity has multiple instances (e.g. mouse2, mouse3,) those instances are changed as well to maintain consistency. The new noun-entity is chosen at random from a list of available unused nouns associated to the main character. If there are no other nouns left, a noun is chosen at random from the ConceptNet dataset. Verbs are also assigned at random based on the subject noun's association list.

3.5.3. Fitness Evaluation

Evaluation for an artifact story is done in three parts. The first part evaluates *inter-sentence cohesion* and averages the semantic closeness and relevance between consecutive story log entries with the assigned nouns and verbs (e.g. *[cat] hunted [mouse]* | *[mouse] ate [cheese].*) This metric is evaluated using cosine similarity scores between the log entries vector embedding from the sentence transformer. The second part of the fitness evaluates the semantic relatedness between the nouns of the story. Ideally, all of the nouns are chosen based on the relevancy to the chosen *main character*, however, if there are more entities than there are related nouns, additional nouns to replace the entities are chosen at random from the ConceptNet dataset. This incentivizes the evolutionary algorithm to select main characters from the dataset with many associations. The last part evaluates the semantic similarity between the replacement story verb and the verb used in the original Amorphous Fortress fortress log entry. This is to incentivize consistency with the game action and the generated verb. For example, generating *[cat] hunted [mouse]* would most likely have a higher similarity score to the original AF log entry *[c.21b3] chased [@.231f]* than an alternative generation of *[cat] licked [mouse]*. The final fitness score for the artifact takes a weighted average of the three individual score parts equally to achieve a float value between 0-1 with 1 being the highest fitness value possible. The heighest weight is given to *inter-sentence cohesion* in order to promote thematic cohesion within the generated narrative, followed by entity-similarity and finally, verb similarity.

4. Experiments

We conduct three different experiment setups for this system and evaluate the generated stories and two different quality diversity algorithms for a total of 6 experiments.

4.1. Quality Diversity Algorithms

We test the system using the *Novelty Search with Local Competition* (NS-LC) algorithm [17] and the *MAP-Elites* algorithm [18] to generate new and diverse stories from the evolutionary system. Novelty is determined by the cosine similarity between story genome vectors (refer to the subsection on Genome Representation.) Artifacts with a minimum fitness value of 0.3 and a minimum novelty score of 1 are stored in the archive. The MAP-Elites algorithm sorts the generated artifacts based on the individual fitness components: intra-sentence cohesion, inter-sentence cohesion, main character semantic similarity, and verb semantic similarity (refer to the subsection on Fitness Evaluation.) The top 5 artifacts with the highest fitness in each of these categories is maintained in the map archive. For both algorithms, new samples for the 90% of the next population generation are taken from these archives. The other 10% is made up of randomly generated artifacts to avoid having a local maxima from being reached during evolution.

4.2. Experiment Setup

The three experiments conducted (all using the 2 quality diversity experiments) explore the different kinds of stories that can be generated using different control variables and parameters. Each experiment is run for a 100 generations with a population size of 100 individuals.

The first experiment, titled **Purely Random**, does not choose associated nouns or verbs for the generated stories. The main character, the other noun-entities, and the verbs for log entry are all chosen completely at random using the custom ConceptNet database. This experiment is intended to act as a baseline comparison for the other 2 experiments. It is also intended to show the thematic range and chaotic lack of cohesion of the possible story outputs.

The second experiment, titled **Story Tropes**, randomly selects a main character but uses associated noun selections for the other entities (e.g. cat is the main character, then dog, mouse, human could be associated nouns.) The verbs are also associated to the subject nouns for each entry log. This is based off of the original description of the system as outlined by the previous sections. With this experiment, the intention is to show a wide range of stories that are thematically cohesive.

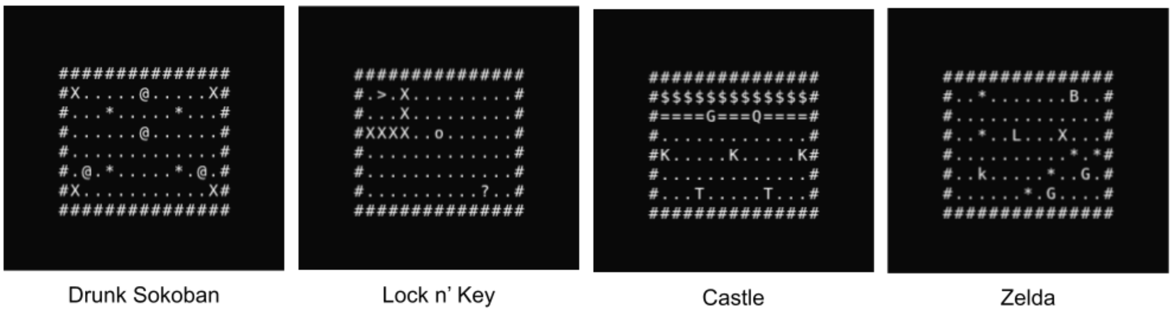
The third experiment, titled **Alternate Universe**, uses a consistent main character for each story artifact. The other noun-entities are still associated but the ordering and potential extra nouns are assigned at random. The verb selection also remains random but associated. This is to evaluate how cohesive or interesting a story could become if the same character is used for generation. This experiment makes the generated stories feel like a narration or retelling of the events in the fortress. We run this experiment with two different pre-selected main characters for each story with both algorithms.

With two algorithms, four stories and three experiments (the third having two sub experiments), we ran a total of 32 experiments.

4.3. Fortress Environments

We use one log each from four different fortresses as the base log files for these experiments. These fortresses were each manually designed in the Amorphous Fortress Online system [19] and given themes but were chosen to provide a more direct thematic comparison between designer intent and the generated stories.

The first fortress, *Drunk Sokoban*, features agents that randomly move around the map and can "push" crate objects onto goal positions. These crates will be destroyed and the goal will transform if successfully placed on top. The second environment, *Lock n' Key* contains a character that needs to collect an object in order to pass through immovable walls and reach the exit area. The third environment *Castle* features 2 "guard knights" characters, 3 "thief" characters, and a "king" and a "queen" who stand in front of a sectioned room full of "gold". The thieves will randomly move and assassinate the king and/or queen if next to them and collect the gold. However, the knights will hunt and kill the thieves if they are within range. The last fortress, *Zelda*, is modeled after characters from The Legend of Zelda series. A character "Link" will pick up "rupees" dropped by a "korok" character. Another character the "Bokoblin" will chase and kill Link if he is within range. A third character "Goron" will push and move "boulders" around the map that can't be passed by any character. Starting images for these 4 fortresses are shown in Figure 3.



```
###############
#X.....@.....X#
#...*.....*...#
#......@......#
#.............#
#.@.*.....*.@.#
#X...........X#
###############
```

```
###############
#.>.X.........#
#...X.........#
#XXXX..o......#
#.............#
#.............#
#..........?..#
###############
```

```
###############
#$$$$$$$$$$$$$#
#====G===Q====#
#.............#
#K.....K.....K#
#.............#
#...T.....T...#
###############
```

```
###############
#..*.......B..#
#.............#
#..*..L...X...#
#..........*.*#
#..k.....*..G.#
#......*.G....#
###############
```

Figure 3: Starting renders of the fortresses used in the experiment.

Again, these thematic meanings and assignments to the agents and entities in these selected fortresses were originally given by the designers. The purpose of this current system is to automatically generate new thematic meaning to fortresses or assign meanings to fortresses in the AF system that do not contain these human-designed themes or narratives.

5. Results and Discussion

The MAP-Elites algorithm was able to generate stories with consistently high fitness, while the Novelty Search algorithm produced a wide variety of "novel" stories, as reflected in archive size. By examining the fitness graphs, archive sizes, and best-fitness stories for each experiment, alongside exploratory analysis, we were able to make several qualitative observations.

Experiment 1: Purely Random As expected, the Pure Random experiment generated the most diverse stories, many of which lacked thematic cohesion. Serving as a baseline, this experiment intentionally avoided using semantic word associations to replace words in the grammar. As a result, it produced a wide variety of characters and situations, not all of them coherent. Nevertheless, some stories displayed unexpected coherence, with related entities emerging even though the fortress does not explicitly encode such connections. For instance, one run produced a narrative centered on "cranberries and marmalade" (Table 2), which, while structurally simple, demonstrates how unguided randomness can surface surprising thematic links.

Experiment 2: Story Tropes By introducing semantic associations between entities, the Story Tropes experiment improved the narrative coherence of the generated stories, which in turn led to higher fitness scores (Table 2). However, this came at the cost of novelty: the size of the archive decreased, as the system produced fewer unexpected combinations once associations were imposed. Interestingly, two different fortresses converged on a story with "pilots" as the main character, suggesting that the algorithm may be exploiting structural affordances in the system's design (see Tables 5 and 6, Appendix).

Experiment 3: Alternate Universe Fixing the main character improved the thematic focus of the generated stories, although the quantitative results were mixed (Table 3). However, if the goal is to generate stories centered on a specific theme or idea, the results of this experiment suggest that the Amorphous Interpretations system is capable of producing cohesive narratives around a central theme. For example, the "explorer" story generated in the Lock n' Key domain demonstrates a focused narrative and strong thematic cohesion (Table 7, Appendix) - the story suggests an expanding land that turns into a continent and an explorer who attempts to conquer and journey through it. This is only one interpretation of this story, as the grammar-based system offers room for a variety of interpretations.

Domain Complexity While the system remains largely independent of the context of the domain, the number of entities and the variety of entity groups affects the fitness and diversity of generated stories. We observed that domains with fewer number of entities (Lock n' Key) were most conducive to this system by producing consistently high fitness scores and large archive sizes (as observed in Tables 2 and 3). This indicated that having a smaller search space improves thematic cohesiveness and diversity in interpretations.

Quantitative vs. Qualitative Evaluation We observed that stories with lower fitness scores could still be cohesive and compelling. For instance, the story generated by the Story Tropes experiment in the Zelda domain (see Table 6, Appendix) produced a meaningful narrative involving airplanes, pilots, and jets, whereas the Pure Random experiment yielded stories such as the "cranberries and skaters" narrative (see Table 5, Appendix), which achieved higher fitness but was far less engaging. This contrast highlights the need for qualitative, human-centered evaluation, or a mixed-methods approach to better assess narrative quality.

Generated Story Themes Outside of the best selected stories, the archives generated from these quality diversity experiments contained a variety of weird, interesting, cohesive stories from just a single log file. The Castle fortress, for example, was originally themed around thieves vs. a guard of royal knights trying to steal gold, the system made interpretive stories such as prisoners vs. jailers, hockey players vs. ice skating, hunters vs. animals, and raindrops vs. water. While these stories were different thematically, they still followed the narrative story structure sifted from the log file of the fortress.

	Experiment 1: Random				Experiment 2: Story Tropes			
	Drunk Sokoban	Lock n' Key	Castle	Zelda	Drunk Sokoban	Lock n' Key	Castle	Zelda
Novelty Search	9	120	6	15	8	35	2	4
MAP-Elites	0.57	0.64	0.58	0.53	0.62	0.75	0.66	0.55

Table 2
Results from Experiments #1 and #2. First row shows the maximum archive size for the novelty search experiments. Second row shows the highest fitness for the MAP-Elites experiments.

	Experiment 3: Alternate Universe (person)				Experiment 3: Alternate Universe (fortress-specific)			
	Drunk Sokoban	Lock n' Key	Castle	Zelda	Drunk Sokoban [worker]	Lock n' Key [explorer]	Castle [guard]	Zelda [fighter]
Novelty Search	3	4	2	2	2	8	5	5
MAP-Elites	0.5	0.73	0.55	0.5	0.55	0.6	0.59	0.56

Table 3
Results from Experiments #3. First row shows the maximum archive size for the novelty search experiments. Second row shows the highest fitness for the MAP-Elites experiments.

6. Future Work

We would like to further develop the generative system to include co-creative story telling and narrative creation from open-ended environments. In this co-creative system, users would be able to directly annotate events or characters in the story while the generative Amorphous Interpretations system would build a narrative around it. This can allow for more exploration into thematic narrative construction or variance in story creation while maintaining cohesion given the user's feedback and direction.

Likewise, a formal user study could be performed to evaluate the quality and selection of the generative system as compared to human evaluation of a story. Feedback from the user study could be used to design a supervised model that could distinguish between "interesting" and "boring" stories. This would allow us to improve the system's current evaluation metric so that it more closely aligns with human preferences and taste.

We would also like to examine how large language models (LLMs) such as ChatGPT and Gemini can be used to augment the grammar-based generated stories into prose-like or fairy-tale styled stories. The LLMs would be able to translate from the simulation's log format, to the thematic interpretation, and finally into a more human-readable and fluid story.

7. Conclusion

With Amorphous Interpretations, we introduce an interpretable narrative system for open-ended simulations. We use two Quality-Diversity algorithms to explore the thematic diversity and cohesion of the narratives generated. We found that the novelty search algorithm generated a large archive of thematically diverse stories for each fortress log while the MAP-Elites algorithm generated stories with more narrative cohesion. We look forward to exploring human evaluation of these artificially interpreted stories and extending the system to include a co-creative narrative design pipeline to allow for even more diverse and fluid interpretations.

Acknowledgments

The graphic icon images used in the diagram are designed by Freepik[4] and taken from the FlatIcon.com website.

[4]https://www.flaticon.com/authors/freepik

Declaration on Generative AI

The author(s) have not employed any Generative AI tools in the writing of this paper.

References

[1] C. G. Langton, Artificial life: An overview (1997).

[2] M. Charity, D. Rajesh, R. Ombok, L. B. Soros, Say "sul sul!" to simsim, a sims-inspired platform for sandbox game ai, in: Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, volume 16, 2020, pp. 182–188.

[3] J. Suarez, Y. Du, P. Isola, I. Mordatch, Neural mmo: A massively multiagent game environment for training and evaluating intelligent agents, arXiv preprint arXiv:1903.00784 (2019).

[4] B. Baker, I. Kanitscheider, T. Markov, Y. Wu, G. Powell, B. McGrew, I. Mordatch, Emergent tool use from multi-agent autocurricula, in: International conference on learning representations, 2019.

[5] J. S. Park, J. O'Brien, C. J. Cai, M. R. Morris, P. Liang, M. S. Bernstein, Generative agents: Interactive simulacra of human behavior, in: Proceedings of the 36th annual acm symposium on user interface software and technology, 2023, pp. 1–22.

[6] J. Ryan, Curating simulated storyworlds, University of California, Santa Cruz, 2018.

[7] M. Kreminski, N. Wardrip-Fruin, M. Mateas, Toward example-driven program synthesis of story sifting patterns., in: AIIDE Workshops, 2020.

[8] S. Johnson-Bey, M. Mateas, Centrifuge: A visual tool for authoring sifting patterns for character-based simulationist storyworlds., in: AIIDE Workshops, 2021.

[9] J. R. Meehan, Tale-spin, an interactive program that writes stories., in: Ijcai, volume 77, 1977, pp. 91–98.

[10] M. O. Riedl, R. M. Young, Narrative planning: Balancing plot and character, Journal of Artificial Intelligence Research 39 (2010) 217–268.

[11] K. Hjaltason, S. Christophersen, J. Togelius, M. J. Nelson, Game mechanics telling stories? an experiment., in: FDG, 2015.

[12] F. Heider, M. Simmel, An experimental study of apparent behavior, The American journal of psychology 57 (1944) 243–259.

[13] N. McIntyre, M. Lapata, Plot induction and evolutionary search for story generation, in: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, 2010, pp. 1562–1572.

[14] R. García, P. García Sánchez, A. Mora, J. Merelo, My life as a sim: evolving unique and engaging life stories using virtual worlds, in: Artificial Life Conference Proceedings, MIT Press One Rogers Street, Cambridge, MA 02142-1209, USA journals-info ..., 2014, pp. 580–587.

[15] J. Grinblat, C. B. Bucklew, Subverting historical cause & effect: generation of mythic biographies in caves of qud, in: Proceedings of the 12th International Conference on the Foundations of Digital Games, 2017, pp. 1–7.

[16] M. Charity, S. Earle, D. Rajesh, M. Wilson, J. Togelius, Amorphous fortress: Exploring emergent behavior and complexity in multi-agent 0-player games, in: 2024 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2024, pp. 01–10.

[17] J. Lehman, K. O. Stanley, Evolving a diversity of virtual creatures through novelty search and local competition, in: Proceedings of the 13th annual conference on Genetic and evolutionary computation, 2011, pp. 211–218.

[18] J.-B. Mouret, J. Clune, Illuminating search spaces by mapping elites, arXiv preprint arXiv:1504.04909 (2015).

[19] M. Charity, M. Wilson, S. Lee, D. Rajesh, S. Earle, J. Togelius, Amorphous fortress online: Collaboratively designing open-ended multi-agent ai and game environments, arXiv preprint arXiv:2502.05632 (2025).

Original AF Story	Sifted Story	Best Generated Story
<1>[@.46ae] moved to 7,2	<8>[@.46ae] pushed [*.66a4]	<8>[cranberries2] completed [skater]
<1>[@.1a7f] moved to 7,2	<54>[@.bf99] pushed [*.efb1]	<54>[cranberries] learned [skater2]
<1>[@.aece] moved to 1,5	<82>[@.aece] pushed [*.f6c2]	<82>[cranberries3] backed [skater3]
<1>[@.bf99] moved to 11,5	<93>[@.bf99] moved to 11,1	<93>[cranberries] deliberated to 11,1
<2>[@.46ae] moved to 6,2	<186>[@.bf99] moved to 6,2	<186>[cranberries] categorized to 6,2
<2>[@.1a7f] moved to 6,2	<250>[@.aece] moved to 5,3	<250>[cranberries3] nested to 5,3
<2>[@.bf99] moved to 11,6	<343>[@.bf99] moved to 4,5	<343>[cranberries] ingested to 4,5
<3>[@.46ae] moved to 7,2	<458>[@.bf99] pushed [*.7dbb]	<458>[cranberries] spread [skater4]
<3>[@.1a7f] moved to 5,2	<459>[@.bf99] pushed [*.7dbb]	<459>[cranberries] hollowed [skater4]
<3>[@.aece] moved to 2,5	<462>[@.bf99] pushed [*.7dbb]	<462>[cranberries] flowered [skater4]
<3>[@.bf99] moved to 10,6	<505>[@.1a7f] moved to 4,3	<505>[cranberries4] skilled to 4,3
<4>[@.46ae] moved to 8,2	<558>[@.aece] pushed [*.7dbb]	<558>[cranberries3] depth [skater4]
<4>[@.1a7f] moved to 5,3	<559>[*.7dbb] took [X.a0f1]	<559>[skater4] beheaded [marmalade]
<4>[@.aece] moved to 3,5	<560>[*.7dbb] transformed into [O.48e7] at 1,6	<560>[skater4] wondered into [gem] at 1,6
<5>[@.46ae] moved to 9,2	<648>[*.66a4] took [X.7597]	<648>[skater] explored [marmalade2]
<5>[@.1a7f] moved to 6,3	<649>[*.66a4] transformed into [O.c1ca] at 13,1	<649>[skater] diverted into [gem2] at 13,1
<5>[@.aece] moved to 3,6	<703>[@.bf99] moved to 11,1	<703>[cranberries] journeyed to 11,1
<5>[@.bf99] moved to 11,6	<720>[@.aece] pushed [*.efb1]	<720>[cranberries3] donated [skater2]
<6>[@.46ae] moved to 8,2	<759>[*.efb1] took [X.e61b]	<759>[skater2] tallied [marmalade3]
<6>[@.1a7f] moved to 5,3	<760>[*.efb1] transformed into [O.c7ff] at 1,1	<760>[skater2] streaked into [gem3] at 1,1
<6>[@.aece] moved to 4,6	<812>[@.1a7f] moved to 3,4	<812>[cranberries4] circled to 3,4
<6>[@.bf99] moved to 10,6	<874>[@.1a7f] pushed [*.f6c2]	<874>[cranberries4] pressed [skater3]
<7>[@.46ae] moved to 9,2	<962>[@.1a7f] moved to 12,2	<962>[cranberries4] helped to 12,2
<7>[@.1a7f] moved to 6,3	<1004>[@.1a7f] moved to 11,1	<1004>[cranberries4] introduced to 11,1
<7>[@.bf99] pushed [*.f6c2]	<1073>[@.aece] moved to 13,2	<1073>[cranberries3] inflicted to 13,2
<8>[@.46ae] pushed [*.66a4]	<1116>[@.46ae] moved to 9,1	<1116>[cranberries2] clouded to 9,1
<8>[@.1a7f] moved to 6,4	<1196>[@.bf99] moved to 5,5	<1196>[cranberries] reinforced to 5,5
<8>[@.bf99] pushed [*.f6c2]	<1201>[@.bf99] moved to 5,4	<1201>[cranberries] threw to 5,4
<9>[@.46ae] pushed [*.66a4]	<1202>[*.f6c2] took [X.700b]	<1202>[skater3] assimilated [marmalade4]
...	<1203>[*.f6c2] transformed into [O.ba86] at 13,6	<1203>[skater3] kidnapped into [gem4] at 13,6
...		

Table 4
Story log comparisons for the **Drunk Sokoban** fortress. The best story was taken from Experiment 1: Random Story using Novelty Search.

Original AF Story	Sifted Story	Best Generated Story
<1>[T.73ab] moved to 3,6	<1>[T.76a3] moved to 6,6	<1>[pilots2] posted to 6,6
<1>[T.76a3] moved to 6,6	<1>[T.73ab] moved to 3,6	<1>[pilots] flew to 3,6
<1>[T.d89f] moved to 10,5	<4>[T.d89f] moved to 11,5	<4>[pilots3] learned to 11,5
<2>[T.76a3] moved to 7,6	<6>[K.fc7c] chased [T.d89f]	<6>[helicopter] injured [pilots3]
<2>[T.d89f] moved to 10,6	<6>[T.76a3] moved to 6,5	<6>[pilots2] enriched to 6,5
<3>[T.73ab] moved to 4,6	<7>[T.d89f] moved to 11,4	<7>[pilots3] mined to 11,4
<3>[T.d89f] moved to 11,6	<9>[T.d89f] blocked by [=.23f6]	<9>[pilots3] shopped by [jet]
<4>[T.73ab] moved to 4,5	<10>[T.73ab] took $[.1a6e]$	<10>[pilots] flew [airplane]
<4>[T.d89f] moved to 11,5	<10> [G.3305] died	<10>[plane] gained
<5>[T.73ab] moved to 5,5	<12> [K.fc7c] chased [T.d89f]	<12>[helicopter] injured [pilots3]
<5>[T.76a3] moved to 6,6	<13> [T.76a3] moved to 5,5	<13>[pilots2] tried to 5,5
<5>[T.d89f] moved to 12,5	<13> [T.73ab] took $[.6d7b]$	<13>[pilots] went [airplane2]
<6>[K.fc7c] chased [T.d89f]	<15>[K.fc7c] took [T.d89f]	<15>[helicopter] injured [pilots3]
<6>[T.73ab] moved to 5,4	<20>[K.7479] chased [T.76a3]	<20>[helicopter2] alphabetized [pilots2]
<6>[T.76a3] moved to 6,5	<21>[T.73ab] took $[.9fe4]$	<21>[pilots] went [airplane3]
<6>[T.d89f] moved to 12,4	<22> [T.73ab] blocked by [=.2486]	<22>[pilots] flew by [jet2]
<7>[T.73ab] moved to 5,3	<24> [K.7479] chased [T.76a3]	<24>[helicopter2] overcame [pilots2]
<7>[T.76a3] moved to 6,6	<25> [T.73ab] took $[.1b0e]$	<25>[pilots] flew [airplane4]
<7>[T.d89f] moved to 11,4	<26>[T.73ab] blocked by [=.0cc4]	<26>[pilots] flew by [jet3]
<8>[K.fc7c] chased [T.d89f]	<26>[K.7479] chased [T.76a3]	<26>[helicopter2] radiated [pilots2]
<8>[T.73ab] moved to 5,2	<29>[T.73ab] took $[.0bdf]$	<29>[pilots] flew [airplane5]
<8>[T.76a3] moved to 6,5	<29> [K.7479] took [T.76a3]	<29>[helicopter2] filled [pilots2]
<8>[T.d89f] moved to 11,3	<32> [T.73ab] moved to 8,1	<32>[pilots] went to 8,1
<9>[T.73ab] moved to 5,1	<38> [T.73ab] took $[.6666]$	<38>[pilots] flew [airplane6]
<9>[T.76a3] moved to 6,6	<41>[T.73ab] took $[.6924]$	<41>[pilots] went [airplane7]
<9>[T.d89f] blocked by [=.23f6]	<43> [T.73ab] took $[.db0a]$	<43>[pilots] went [airplane8]
<10>[G.3305] died	<45>[T.73ab] took [$.4709]	<45>[pilots] flew [airplane9]
<10>[K.fc7c] chased [T.d89f]	<49>[T.73ab] moved to 11,1	<49>[pilots] went to 11,1
<10>[T.73ab] took [$.1a6e]	<50>[K.fc7c] chased [T.73ab]	<50>[helicopter] injured [pilots]
...	<53>[K.fc7c] took [T.73ab]	<53>[helicopter] injured [pilots]

Table 5
Story log comparisons for the **Castle** fortress. The best story was taken from Experiment 2: Trope Story using Novelty Search.

Original AF Story	Sifted Story	Best Generated Story
<2>[G.4709] moved to 10,6	<6>[B.5b49] moved to 12,1	<6>[plane] took to 12,1
<3>[G.6924] moved to 12,6	<11>[X.1a6e] added [$.3305] at 10,2	<11>[pilots] flew [jet] at 10,2
<3>[L.6d7b] moved to 5,3	<15> [L.6d7b] chased [$.3305]	<15>[helicopter] injured [jet]
<5>[G.4709] moved to 10,5	<15>[G.6924] moved to 10,4	<15>[airplane] traversed to 10,4
<5>[L.6d7b] moved to 5,2	<21>[X.1a6e] added [$.6f30] at 8,1	<21>[pilots] flew [jet2] at 8,1
<5>[X.1a6e] moved to 9,3	<21> [B.5b49] chased [L.6d7b]	<21>[plane] went [helicopter]
<6>[B.5b49] moved to 12,1	<22> [B.5b49] took [L.6d7b]	<22>[plane] gained [helicopter]
<7>[G.4709] moved to 11,5	<27> [G.4709] moved to 8,4	<27>[airplane2] annoyed to 8,4
<7>[L.6d7b] moved to 5,3	<29> [G.4709] pushed [*.6666]	<29>[airplane2] favored [kite]
<9>[G.4709] moved to 12,5	<31> [X.1a6e] added [$.ab76] at 8,3	<31>[pilots] flew [jet3] at 8,3
<9>[G.6924] moved to 11,6	<35>[G.6924] pushed [*.6666]	<35>[airplane] lifted [kite]
<9>[L.6d7b] moved to 5,4	<45>[X.1a6e] moved to 6,1	<45>[pilots] flew to 6,1
<9>[X.1a6e] moved to 9,2	<49>[G.4709] moved to 9,2	<49>[airplane2] repaid to 9,2
<11>[G.4709] moved to 11,5	<51>[B.5b49] moved to 10,3	<51>[plane] gained to 10,3
<11>[G.6924] moved to 11,5	<57>[G.6924] moved to 10,2	<57>[airplane] crashed to 10,2
<11>[L.6d7b] moved to 6,4	<61>[X.1a6e] added [$.8a85] at 5,1	<61>[pilots] flew [jet4] at 5,1
<11>[X.1a6e] added [$.3305] at 10,2	<71> [X.1a6e] added [$.3c08] at 6,2	<71>[pilots] went [jet5] at 6,2
<13> [G.4709] moved to 11,6	<81>[X.1a6e] added [$.97eb] at 7,1	<81>[pilots] flew [jet6] at 7,1
<13> [G.6924] pushed [*.9fe4]	<91> [B.5b49] moved to 8,3	<91>[plane] travelled to 8,3
<13> [L.6d7b] moved to 7,4	<93> [X.1a6e] moved to 7,1	<93>[pilots] flew to 7,1
<13> [X.1a6e] moved to 9,1	<93> [G.4709] pushed [*.6666]	<93>[airplane2] warned [kite]
<15> [G.6924] moved to 10,4	<116> [B.5b49] moved to 7,2	<116>[plane] travelled to 7,2
<15> [L.6d7b] chased [$.3305]	<121> [G.4709] moved to 7,1	<121>[airplane2] shifted to 7,1
<16>[L.6d7b] chased [$.3305]	<121> [X.1a6e] added [$.e832] at 5,2	<121>[pilots] went [jet7] at 5,2
<17> [G.6924] moved to 11,4	<121>[G.6924] moved to 12,4	<121>[airplane] travelled to 12,4
<17> [L.6d7b] chased [$.3305]	<141>[X.1a6e] added [$.d0a5] at 3,2	<141>[pilots] went [jet8] at 3,2
<18>[L.6d7b] chased [$.3305]	<151>[B.5b49] moved to 7,3	<151>[plane] broke to 7,3
<19>[G.4709] moved to 10,6	<155>[G.6924] moved to 12,6	<155>[airplane] crashed to 12,6
...	<161>[G.4709] moved to 8,4	<161>[airplane2] beat to 8,4
...	<161>[G.6924] moved to 12,5	<161>[airplane] visited to 12,5

Table 6
Story log comparisons for the **Zelda** fortress. The best story was taken from Experiment 2: Trope Story using Novelty Search. Interestingly enough, also using pilots as the main character although randomly selected.

Original AF Story	Sifted Story	Best Generated Story
<1>[o.4114] moved to 7,4	<1>[o.4114] moved to 7,4	<1>[land] grew to 7,4
<2>[o.4114] moved to 6,4	<2>[o.4114] moved to 6,4	<2>[land] grew to 6,4
<3>[o.4114] moved to 7,4	<3>[o.4114] moved to 7,4	<3>[land] grew to 7,4
<4>[o.4114] moved to 8,4	<4>[o.4114] moved to 8,4	<4>[land] grew to 8,4
<5>[o.4114] moved to 8,3	<5>[o.4114] moved to 8,3	<5>[land] grew to 8,3
<6>[o.4114] moved to 8,2	<6>[o.4114] moved to 8,2	<6>[land] grew to 8,2
<7>[o.4114] moved to 8,3	<7>[o.4114] moved to 8,3	<7>[land] grew to 8,3
<8>[o.4114] moved to 9,3	<8>[o.4114] moved to 9,3	<8>[land] grew to 9,3
<9>[o.4114] moved to 9,4	<9>[o.4114] moved to 9,4	<9>[land] grew to 9,4
<10>[o.4114] moved to 9,5	<10>[o.4114] moved to 9,5	<10>[land] grew to 9,5
<11>[o.4114] chased [?.e7b5]	<11>[o.4114] chased [?.e7b5]	<11>[land] grew [continent]
<12>[o.4114] chased [?.e7b5]	<12>[o.4114] chased [?.e7b5]	<12>[land] grew [continent]
<13>[o.4114] chased [?.e7b5]	<13>[o.4114] chased [?.e7b5]	<13>[land] grew [continent]
<14>[o.4114] took [?.e7b5]	<14>[o.4114] took [?.e7b5]	<14>[land] grew [continent]
<15>[o.4114] transformed into [O.c31a]	<15>[o.4114] transformed into [O.c31a]	<15>[land] grew into [explorer]
<16>[O.c31a] chased [>.34c6]	<16>[O.c31a] chased [>.34c6]	<16>[explorer] kept [territory]
<17>[O.c31a] chased [>.34c6]	<17>[O.c31a] chased [>.34c6]	<17>[explorer] discovered [territory]
<18>[O.c31a] chased [>.34c6]	<18>[O.c31a] chased [>.34c6]	<18>[explorer] kept [territory]
<19>[O.c31a] chased [>.34c6]	<19>[O.c31a] chased [>.34c6]	<19>[explorer] treasured [territory]
<20>[O.c31a] chased [>.34c6]	<20>[O.c31a] chased [>.34c6]	<20>[explorer] journeyed [territory]
<21>[O.c31a] chased [>.34c6]	<21>[O.c31a] chased [>.34c6]	<21>[explorer] lost [territory]
<22>[O.c31a] chased [>.34c6]	<22>[O.c31a] chased [>.34c6]	<22>[explorer] kept [territory]
<23>[O.c31a] chased [>.34c6]	<23>[O.c31a] chased [>.34c6]	<23>[explorer] journeyed [territory]
<24>[O.c31a] chased [>.34c6]	<24>[O.c31a] chased [>.34c6]	<24>[explorer] journeyed [territory]
<25>[O.c31a] chased [>.34c6]	<25>[O.c31a] chased [>.34c6]	<25>[explorer] journeyed [territory]
<26>[O.c31a] chased [>.34c6]	<26>[O.c31a] chased [>.34c6]	<26>[explorer] journeyed [territory]
<27>[O.c31a] chased [>.34c6]	<27>[O.c31a] chased [>.34c6]	<27>[explorer] journeyed [territory]
<28>[O.c31a] chased [>.34c6]	<28>[O.c31a] chased [>.34c6]	<28>[explorer] journeyed [territory]
<29>[O.c31a] chased [>.34c6]	<29>[O.c31a] chased [>.34c6]	<29>[explorer] journeyed [territory]
<30>[O.c31a] died	<30>[O.c31a] died	<30>[explorer] journeyed

Table 7
Story log comparisons for the **Lock 'n Key** fortress. The best story was taken from Experiment 3: AU Story using Novelty Search and a set main character of 'explorer'. (Note: Log slightly modified to fit within paper margins)