Subverting Historical Cause & Effect: Generation of Mythic Biographies in *Caves of Qud*

Jason Grinblat
Freehold Games
Berkeley, California
jason@freeholdgames.com

C. Brian Bucklew
Freehold Games
Walkerton, Indiana
bbucklew@freeholdgames.com

ABSTRACT

Procedurally generating history is a daunting task. The process of proceduralization requires codifying relationships into rules, and real-life histories are tangled networks of people, places, and events whose complexities obscure their relational mechanics. The task is further complicated by the fact that history serves a rhetorical function, i.e., historical accounts are used to promote certain cultural narratives that can belie the facts they purport to narrativize (this function is routinely neglected in video games' treatment of history). This paper describes a novel approach for procedurally generating coherent biographical narratives for historical figures, as implemented in our far-future roguelike game *Caves of Qud.* Using a state machine and replacement grammar as procedural tools, we construct a system that subverts the logic of cause and effect in favor of a process that first generates historical events and rationalizes them ex post facto.

CCS CONCEPTS

•**Applied computing** → **Computer games;** *Arts and humanities;*
•**Computing methodologies** → Modeling and simulation;

KEYWORDS

Qualitative procedural generation; history; myth; narrative generation

ACM Reference format:
Jason Grinblat and C. Brian Bucklew . 2017. Subverting Historical Cause & Effect: Generation of Mythic Biographies in *Caves of Qud.* In *Proceedings of FDG'17, Hyannis, MA, USA, August 14-17, 2017,* 7 pages.
DOI: 10.1145/3102071.3110574

1 INTRODUCTION

Procedural generation in games continues to transcend its origins as a tool to create physical spaces and veer into the qualitative realms of history, religion, language, and culture [8]. Simulation genre paragon *Dwarf Fortress* [1] already features an extensive historical simulation with many interesting results [5]. But there's a distinction to be made between history as a *process* and history as an *artifact.* The former can be conceptualized as the playing out of rules and relationships that continually produce the present. To simulate this process, we might seek to reproduce its logic. On the other hand, the latter is a constituent of the present, something we engage with through a contemporary lens and whose complexities may be obscured by that flattening of perspective. Only historical accounts are accessible to us in the present, and the further back we look, the greater our uncertainty becomes about their veracity. To generate this type of history—that is, to generate historical accounts—we might also seek to recreate the logic that produced it, or we might invent a new logic that reproduces it altogether.

In this paper, we present an approach for procedurally generating historical and mythic narratives that circumvents the recreation of historical logic in favor of a process of historical *rationalization.* Our system first decides that a particular historical event will occur, then it rationalizes a cause based on the states of the historical entities at play. The concept is related to the narrative principle of *Schrödinger's Gun*, which states that a fictional world is constrained only by what's been revealed to the audience, and work done with plan-based narrative generation to mediate resolution when player actions create conflicts [11]. Instead of using rationalization to account for player agency, however, we use it to produce a variety of curated historical events without having to simulate their underlying logic.

We situate our approach among the constellation of works that simulate history, explore its rhetorical function, and use replacement grammars for qualitative procedural generation. In particular, both *Dwarf Fortress* and *Bad News* [12] take a deeply simulative approach to history generation. Though it doesn't employ procedural generation, *Opera Omnia* [9] argues that from the perspective of the present, historical logic is malleable and serves political ends. Finally, *Voyageur* [4] makes extensive use of replacement grammars to describe extraterrestrial landscapes, biomes, and cultures while maintaining a consistent voice.

We begin by establishing the context and introducing our far-future roguelike game, *Caves of Qud* [6], in which we implemented this system. We articulate the design goals that motivated our particular approach, and we describe the resultant system in some detail, including its interaction and generation models. Then we discuss the system's salient features, including the lack of expected causality, the parameterization of historical events, the emergence of coherent narratives, event branching, and the use of a replacement grammar to generate historical descriptions. We continue by illustrating how the generated histories impact the game world and the experience of its players. Finally, we discuss the system's successes and the possibility of future work.

2 *CAVES OF QUD*

Caves of Qud is a science fantasy roguelike game currently in development by the authors. It's set thousands of years in the future among the geologically reclaimed ruins of a vast arcology. While we identify *Caves of Qud* with the post-apocalypse genre, the game complicates the genre's usual preoccupation with the aesthetics of decay by focusing as much of its attention on the richness of the descendent cultures as the departed ones. It draws inspiration from a variety of sources, including pen & paper RPG *Gamma World,* "Dying Earth" genre fiction like Clark Ashton Smith's *Zothique* and Gene Wolfe's *The Book of the New Sun*, the subversive works of New Wave sci-fi authors like Ursula K. Le Guin, historical texts like Edward Gibbon's *The Decline and Fall of the Roman Empire*, and roguelike predecessors *Dwarf Fortress* and *Ancient Domains of Mystery (ADOM)*.

We use a variety of techniques to construct the world of *Caves of Qud*. In many ways, the game is a hybrid of handcrafted and procedural systems. In contrast to most roguelikes, it features a static, handcrafted narrative of the sort you might expect to encounter in a traditional open-world RPG. The world map is also static, but individual areas are highly procedural and vary in detail from game to game. Many physical systems are simulated, environments and bodies are mutable, and there's a rich social and political landscape for players to navigate.

Caves of Qud's themes are manifold. By letting players customize characters via a system of body and mind-altering mutations, it aims to explore our relationship with our morphology. By procedurally situating players in a variety of circumstances with technological artifacts, sometimes as their masters and sometimes their victims, it aims to depict our complicated interdependence with technology. Most relevantly to this paper, by embedding players in a layered labyrinth of lost and contemporary civilizations, it argues that our insights into the past are always filtered through the lens of the present.

Development on *Caves of Qud* began in 2007, and in 2010 we released a freeware beta version as Freehold Games, LLC. In July 2015, *Caves of Qud* launched on Steam Early Access[1]. At the time of this paper's authoring, over 30,000 Steam users have purchased *Caves of Qud*, steamdb.info ranks it as the 44th highest-rated game on Steam[2], and Steam lists it as the highest-rated post-apocalyptic game on their platform[3].

On December 23, 2016, we released an update that included the history generation system discussed in this paper. On May 5, 2017, we released a further update that lets players browse historical texts they've discovered via an in-game journal. The game's full release is slated for 2018.

3 MOTIVATION AND DESIGN GOALS

The idea that the present filters our insights into the past played a major role in motivating our design for a generative history system. In particular, we were primarily interested in players' engagement with history as an artifact—that is—as a diegetic experience in the game world's present. Accordingly, we didn't aim to explicitly simulate historical logic, just as most procedural terrain generators don't aim to simulate water erosion. This approach stands in contrast to genre exemplar *Dwarf Fortress*, which allows players to interrupt its simulated history at any point to start play, and so must remain agnostic as to what's past and what's present.

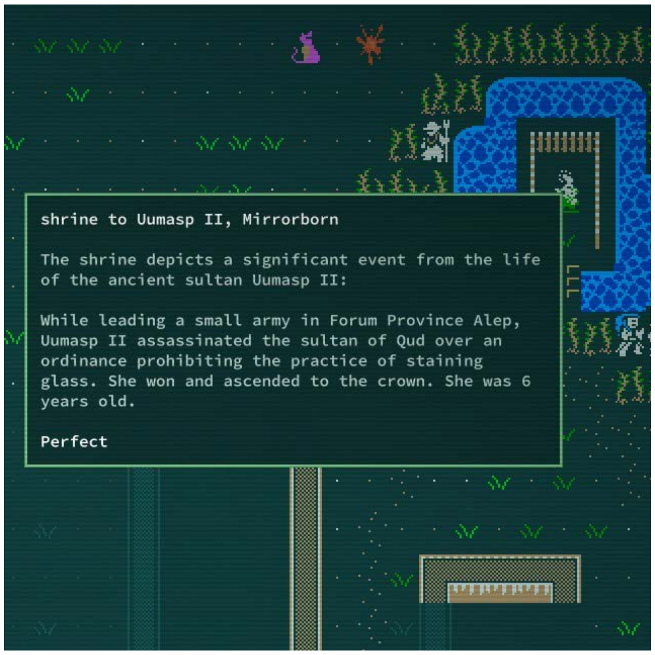
Put another way, our goal was to generate history in order to simulate historical *accounts*. These accounts would be relayed through cultural artifacts or diegetic history texts, and like their real-life counterparts, would originate from inherently limited perspectives and represent particular points of view.

Caves of Qud's state at the time of this system's development provided several constraints that guided our design. For one, because the system would serve in a subordinate position to the major systems already at play, we couldn't devote the resources to research and develop a generative system whose complexity matched the historical and cultural simulations of qualitative PCG exemplars *Dwarf Fortress* and *Ultima Ratio Regum* [7]. Secondly, the generated histories would have to mesh with the handcrafted lore and narratives that had already been developed for the game. These two constraints lead to the decision to focus the generated histories around the mythic lives of five significant rulers—called *sultans*—from a diegetic age known as the sultanate, in which one of Qud's early advanced cultures thrived and before which much of the handwritten lore took place.

From this point, we decided on some aesthetic principles. Firstly, we wanted to generate structured text that matched the voice of the game's handwritten text. Secondly, because we highly value novelty in our output, we sought to avoid prescribing a morphology for the sultans' lives. That is, we didn't set out to generate life events along a narrative arc, preferring instead to give the generator room to trace atypical life paths for the sultans and allowing players to employ apophenia, the human tendency to perceive patterns, in their interpretations. This last point was especially important to us. Players would be engaging with the sultan narratives in a game world full of rich narrative context. The sultans would be part of a cultural tapestry that included the ruined environments of the diegetic past, the NPCs and cultures of the present, and the aggregated experiences of their myriad player characters. Each of these would act as a cultural touchstone that colors their interpretations of the sultans' lives. This would allow us to aim for evocative biographical narratives—rather than meticulously detailed ones—that leverage our players' apophenia. Because we intended to generate several sultans per instantiation of the game world, and because, as part of a tertiary game system, the details of any individual sultan's life would have a limited effect on the player's critical path, we afforded ourselves this freedom.

Finally, in order to confer both a characteristic personality and a mythic quality to each sultan, we decided to assign each one an archetypal unit of culture we call a *domain*. Types of domains include culturally resonant, physical objects and phenomena such as *glass*, *jewels*, *ice*, and *stars*, as well as abstract ideas such as *might*, *scholarship*, and *chance*. Sultans would be assigned a single domain when they were initialized but could accrue others over the course their lives (between 0 and 2 in practice).

[1]http://store.steampowered.com/app/333640/Caves_of_Qud/
[2]https://steamdb.info/stats/gameratings/
[3]http://store.steampowered.com/search/?sort_by=Reviews_DESC&tags=3835



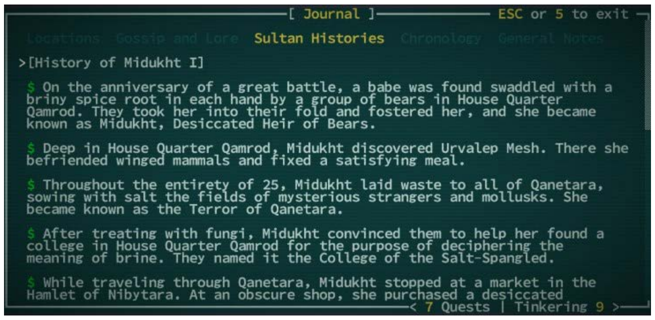
```
shrine to Uumasp II, Mirrorborn

The shrine depicts a significant event from the life
of the ancient sultan Uumasp II:

While leading a small army in Forum Province Alep,
Uumasp II assassinated the sultan of Qud over an
ordinance prohibiting the practice of staining
glass. She won and ascended to the crown. She was 6
years old.

Perfect
```

Figure 1: A shrine depicting a historical event from the life of Uumasp II, a procedurally generated sultan of ancient Qud. (©Freehold Games)



```
[ Journal ]                ESC or 5 to exit
Locations  Gossip and Lore  Sultan Histories  Chronology  General Notes
>[History of Midukht I]
 $ On the anniversary of a great battle, a babe was found swaddled with a
 briny spice root in each hand by a group of bears in House Quarter
 Qamrod. They took her into their fold and fostered her, and she became
 known as Midukht, Desiccated Heir of Bears.

 $ Deep in House Quarter Qamrod, Midukht discovered Urvalep Mesh. There she
 befriended winged mammals and fixed a satisfying meal.

 $ Throughout the entirety of 25, Midukht laid waste to all of Qanetara,
 sowing with salt the fields of mysterious strangers and mollusks. She
 became known as the Terror of Qanetara.

 $ After treating with fungi, Midukht convinced them to help her found a
 college in House Quarter Qamrod for the purpose of deciphering the
 meaning of brine. They named it the College of the Salt-Spangled.

 $ While traveling through Qanetara, Midukht stopped at a market in the
 Hamlet of Nibytara. At an obscure shop, she purchased a desiccated
                                        < 7 Quests | Tinkering 9 >
```

Figure 2: Several gospels from a sultan's life appear in the player's journal.

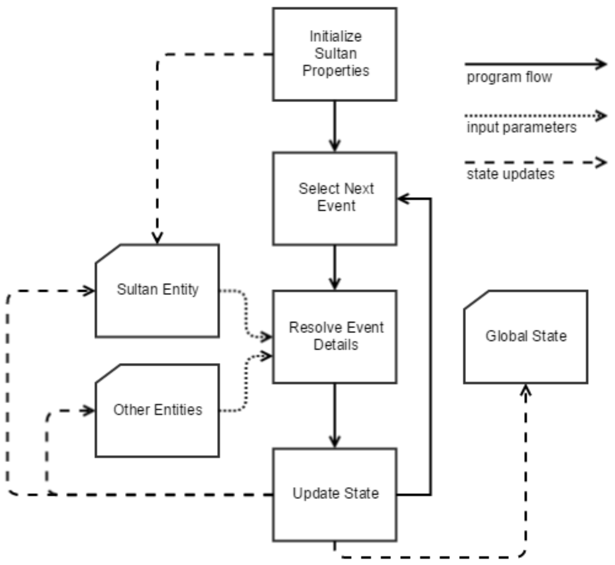


Figure 3: Flow diagram for the generation of a sultan's history.

4 GENERATED HISTORIES

4.1 Gospels and Cultural Artifacts

When players start a new game of *Caves of Qud*, the world-creation engine generates a unique history in five periods, each one centered around a generated sultan. Each period is comprised of several historical events, and for each event, a descriptive text snippet is generated. Internally, we call these single-event text snippets *gospels*. Histories are generated one gospel at a time, and so gospels form the basic discrete unit of history in *Caves of Qud*. In the system's current implementation, each historical event manifests a single gospel.

During play, players engage with the generated history via gospels shared with them by NPCs or appearing in the descriptions of cultural artifacts—namely, shrines, paintings, and engravings—encountered in the game world. These cultural artifacts are generated dynamically as players explore new areas, and each one depicts a single gospel from a sultan's life, chosen randomly from the generated history. Figure 1 shows an example of a sultan shrine, including the gospel for the historical event it depicts.

Players can encounter these historical snippets in any order. As they do, they are inscribed in a journal and sorted chronologically (see Figure 2). Players can swap snippets with NPCs through a custom of cultural exchange called the *water ritual* that's mediated by a currency of reputation with various in-game factions. Over time, players accrue more and more of the gospels from a sultan's life, and a biographical narrative coheres.

4.2 Generative Model

Fundamentally, our system models history as the interplay between *historical entities*—places, items, and sultans represented as data structures—and *historical events* that modify the properties of those entities. In order to engender coherence, the events themselves are parameterized by the properties of existing historical entities, including the very ones they modify.

At a high level, the system generates an individual sultan's history via the following process, illustrated in Figure 3. First, it instantiates a historical entity that represents the sultan in an initial state described by a few core properties (name, pronouns, birth year, birth region, location in birth region, and domain). Then, it randomly chooses a life event—for example, *sieges a city*—from a pool of such events. Based in part on the sultan's state and in part on random branching, the event resolves an outcome, and then

Number	Gospel
0	(sultan initialization; name set to ***Antixerpur***, pronouns set to ***she/her***, birth region set to ***the Philosophers' Quarter of Shaneruk***, and domain set to ***ice***)
1	*At daybreak on the first day of summer*, a ***geologist*** found a babe with a freezing icicle in each hand outside *her* dig site. *She* and *her* fellow geologists adopted the babe and named her Antixerpur.
2	In 230, Antixerpur assassinated the sultan of Qud over an ordinance prohibiting the practice of encasing things in ice. She won and **ascended to the throne**. She was 21 years old.
3	Acting against the prohibition on the practice of taking a spiritual trek through the tundra, Antixerpur led an army to the gates of ***Minekesh***. She *sacked* Minekesh *and persecuted its citizens, forcing them to change its name to* **Antixerpurplatz**.
4	Acting against the prohibition on the practice of encasing things in ice, Antixerpur led an army to the gates of ***Darchesh***. She *sacked* Darchesh *and slaughtered its citizens, forcing them to change its name to* **Antixerpurabad**.
5	After *treating with* ***cats***, Antixerpur convinced them to help her found a **dig site** in the Philosophers' Quarter of Shaneruk for the purpose of excavating ancient blocks of ice. They named it the **Freezing Dig Site**.
6	At the Battle of ***Old Teggash***, Antixerpur fought to liberate cats. She wielded a frosty ***hammer*** with such prowess, that it became forever known as **Frostycus Catsfriend**.
7	*Deep in the wilds of* the Philosophers' Quarter of Shaneruk, Antixerpur stumbled upon a clan of **bears** performing a secret ritual. Because of her reputation for murdering someone with a dagger made of rime, they *furiously rebuked her and* ***declared her a villain to their kind***.
8	*While traveling near* Old Teggash in the Philosophers' Quarter of Shaneruk, Antixerpur was captured by bandits. She *languished in captivity for 7 years, eventually escaping to* ***Urashur***.
9	*While wandering around* the Philosophers' Quarter of Shaneruk, Antixerpur discovered the ***Shrine at Mailimrod***. There she befriended ***highly entropic beings*** and calculated the distance to a nearby star.
10	In 300, Antixerpur won a decisive victory against the combined forces of ***the Jewelers' Quarter of Biilitum*** at the bloody Battle of ***Tappa Cave***, though she lost her prized **Frostycus Catsfriend** during the course of the conflict. As a result of the battle, Tappa Cave was so devastated by icy winds that it was renamed the **Freezing Marsh**.
11	In 302, Antixerpur won a decisive victory against the combined forces of ***Suppir*** at the bloody Battle of ***Miarravah***. As a result of the battle, Miarravah was so devastated by icy winds that it was renamed **Freezingmoor**.
12	*Deep in* ***the Historians' Quarter of Tunepad***, Antixerpur discovered ***Tarchenna***. There she befriended ***mysterious strangers*** and dug into the earth's strata.
13	In 306, Antixerpur, the Untitled, died of natural causes. She was 97 years old.

Table 1: Text representing a change in state is bolded. Text generated according to existing state is underlined. Text generated via random branching or synonymization is *italicized.*

it modifies the sultan's properties and the properties of any non-sultan historical entities that are consequently affected. A gospel for the event is generated via a Tracery-like [3] replacement grammar whose rules map sultan properties to text fragments for a variety of narrative circumstances. As a result of the event, some global properties of the history are also updated, such as the current year. The system then serially chooses and processes about twelve more events in the same fashion, each one parameterized by the sultan's state at the start of the event, and each one transitioning the sultan to a new state by the end of the event. Finally, if the sultan is still alive, the system selects a generic death event that results in the sultan's death. Taken in aggregate, the gospels for the events in the sultan's life form their biography.

4.3 Causality

Note an important characteristic of the system: there's no logic behind the choice of events. Historical cause and effect aren't intrinsic. Events themselves are chosen at random, though their gospels often profess causes. These rationalizations are generated inside the events and are mediated by the sultan's state. As an example, consider the *sieges a city* event, whose gospel starts with the following pattern:

> Acting against *#injustice#*, *#sultanName#* led an army to the gates of *#location#*.[4]

When determining how to replace *#injustice#*, the event examines the sultan's properties for meaningful state. If, for instance, the sultan has any allied animal factions—say, frogs—*#injustice#* may be replaced with "the persecution of frogs". In this way, the event's effect precedes its cause. When the sultan's state fails to produce a suitable cause, the event can even create one by, say, altering the sultan's allied factions property to include frogs and performing the substitution described above. In these cases there's a full reversal of the expected causality; the effect causes the cause.

4.4 Parameterization and Narrative Coherence

Though there's no intrinsic causality to the series of events in a sultan's life, the event parameterization promotes player-interpreted causalities that give rise to coherent biographical narratives. The sultan's shared state acts as a glue that holds the disjointed events together. Domains play an especially important role in the parameterization of historical events. In the context of the replacement

[4]This example uses Tracery's syntax.

grammar, almost all gospel patterns include symbols that represent domains, meaning that once the replacement is complete, the generated text frequently includes narrative references to the sultan's domains. The effect is the production of a narrative coupling between these domains and the sultan's personality. The domains act as narrative threads that tie together the events of a sultan's life.

Consider the typical example in Table 1, which includes the full set of gospels for the life of the sultan Antixerpur with text fragments marked up according to how they were generated. In gospel 5, Antixerpur treats with cats and convinces them to help her found an excavation site. Then, in gospel 6, she liberates cats at the Battle of Old Teggash. Though Antixerpur's rendezvous with cats played no role in the system's decision to have her initiate the Battle of Old Teggash, the gospel claims otherwise. When put to the task of rationalizing the battle, the event logic repurposed Antixerpur's affiliation with cats, which had just been created as a result of gospel 5. With the aid of apophenia, players can draw narrative conclusions about the deal Antixerpur struck with cats. Because of the limited number of sultan properties shared across many events, emergent micro-narratives like this one are quite common.

Antixerpur's domain of ice, chosen during the pre-historic initialization step, was also repeatedly incarnated in the gospels, coupling her with the archetypal phenomenon. Miraculously, she's found as a babe with icicles in her hands (gospel 1). This occurs as the result of a birth event that functions as an orphaned hero *mytheme*, or discrete unit of myth [10]. Early in her life, she fights against the prohibition on ice-associated practices: encasing things in ice (gospels 2 and 4) and taking spiritual treks through the tundra (gospel 3). Later, she founds a dig site to excavate blocks of ice (gospel 5), wields a frosty hammer during a momentous battle (gospel 6), and historically devastates two sites with icy winds in her wake (gospels 10 and 11). These ice-flavored text fragments are manifested by the replacement grammar according to Antixerpur's initialized domain, and collectively they act as narrative force that pushes through the aggregated events of her life.

4.5 Event Branching and Text Replacement

To understand the inner workings of the system, it's instructive to examine how an individual event's logic plays out, updating state and generating a gospel via the replacement grammar during its resolution. Consider gospel 2 from Table 1, reproduced below.

> In 230, Antixerpur assassinated the sultan of Qud
> over an ordinance prohibiting the practice of encasing things in ice.
> She won and **ascended to the throne**. She was 21 years
> old.

This gospel was generated by a *challenge sultan* event. This event's logic contains a significant branch depending on whether the (would-be) sultan has already become the sultan at the start of the event. If they have, then the event generates a gospel about a challenger who defies the sultan for some generated reason. If they have not, as is the case with Antixerpur at the start of gospel 2, then the would-be sultan assumes the role of the challenger. As the challenger, the would-be sultan always wins and takes the crown, and their state is updated accordingly.

Let's look at how the event rationalized Antixerpur's challenge. Each event contains logic controlling the types of properties that could determine its outcome and get reflected in its gospel text. For the *challenge sultan* event, the candidate properties are *allied factions*, *profession*, and *domains*. The event first randomly determines which property to use, rerolling if it selects a property for which the sultan doesn't have a value. For example, if *allied factions* is chosen but the sultan has no allied factions, the result is rerolled. If no valid value is found, some events create new values by altering one of the candidate properties, thereby inventing a cause (as described in 4.3). In the case of the *challenge sultan* event, the value instead defaults to *domains*, which is guaranteed to have at least one value since a domain is set for the sultan in the initialization step. For gospel 2, the value is *ice*.

The pattern of the gospel segment representing *challenge sultan*'s rationalization differs based on which property is chosen. For domains, the pattern is "over an ordinance prohibiting the practice of *#practice#*". To replace *#practice#* with text, the event looks up the appropriate grammar rule, stored in a static, nested JSON file, by using a query language we developed for this purpose. Within the grammar, there are a variety of rules that map keys representing narrative concepts to arrays of text fragments for each domain; *practices* is one such key. Put another way, each domain's JSON representation contains a practices rule. In our query language, *#practice#* takes the following form:

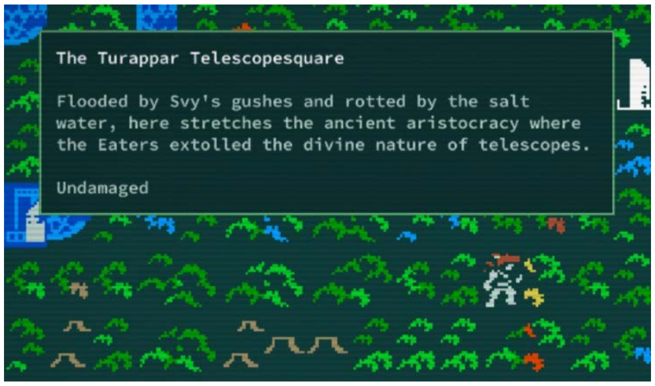
$< domains.sultan\$domains[random].practices.!random >$

In the case of gospel 2, the syntax $domains.sultan\$domains[random]$ resolves to *ice*. The resultant transformed query indicates where to find the appropriate practices rule in the JSON structure, with the $!random$ syntax signifying a random choice from among that rule's array of text fragments. For gospel 2, the query referenced the *practices* rule contained in *ice*'s JSON object and randomly selected the text "encasing things in ice". Other possible values included "ice sculpting" and "taking a spiritual trek through the tundra". If Antixerpur were associated with a different domain—say, *time*—*#practice#* would instead be replaced with a random text fragment from the *practices* rule contained in *time*'s JSON object. In that case, the query would have returned a text fragment such as "watching sand sift through an hourglass".

The grammar file contains rules for several sultan properties in addition to general rules that produce curated synonyms for common words and phrases in the game's voice. Query statements can be embedded in the grammar rules themselves, enabling arbitrary levels of nesting and allowing us to generate the variety of text seen in Antixerpur's gospels.

4.6 Impact on the Generated World

Although the primary means of engagement with the generated histories in *Caves of Qud* is knowledge exchange with NPCs and the examination of cultural artifacts, historical events also shape the generated game world. Places that the sultans visit are modeled as historical entities and instantiated as historic sites during world generation (see Figure 4). Items named during the course of life events, such Frostycus Catsfriend, are also modeled as historical entities. If a sultan leaves a named item at a location as the


The Turappar Telescopesquare

Flooded by Svy's gushes and rotted by the salt water, here stretches the ancient aristocracy where the Eaters extolled the divine nature of telescopes.

Undamaged

Figure 4: A historic site generated in the world history, instantiated in the game world, and procedurally described.

result of a life event, as Antixerpur lost Frostycus Catsfriend during the Battle of Tappa Cave (gospel 10), that item is instantiated according to its historical properties at the appropriate historic site. In the case of Frostycus Catsfriend, the instantiated item is a hammer that deals frost damage and grants bonus reputation with cats. Players engage with the histories through cultural artifacts that tell the sultans' stories, and this historical knowledge acts as a vector for players to find and engage with the material remnants of the sultans' lives.

5 EVALUATION

Because the system described in this paper has been in the hands of players for six months, we're in a unique position to evaluate its effectiveness, both in terms of our own assessment and the responses of our players. The fact that the system plays a subordinate role complicates any quantitative analysis of game sales or active players, so we stick to a mostly qualitative approach. However, as a brief note, we did experience a significant uptick both of sales and active players following the update that introduced the system. With some confidence, we attribute most of the uptick to the fact that we crossed a review threshold into the "Overwhelmingly Positive" category on Steam, probably increasing our baseline visibility. That being said, the update was likely responsible—at least in part—for the spate of reviews that pushed us into the new category.

Now for the qualitative analysis. As for the goal of producing novel output in the voice of the game's handwritten text, we believe the system is quite successful. An illustrative example is the child crowned sultan in the gospel of Figure 1. In testing our output, we were surprised by the phenomenon of children accomplishing extraordinary feats, a consequence enabled by the lack of a prescribed ordering or categorization of events. Fortuitously, extraordinary children fit into the mythic mold we wanted to cast for sultans. In fact, the last line of this gospel, in which the sultan's age is revealed, was added after we noted the phenomenon and realized it would make a powerful narrative impression on players.

In particular, the replacement grammar was remarkably successful at enabling us to generate text in the game's voice. *Caves of Qud* relies heavily on its corpus of text—over 40,000 words—to establish its voice, and via the replacement rules we were able to codify our diction and repackage it procedurally to great effect.

By our evaluation, players reacted positively to individual gospels. As of the authoring of this paper, gospels are featured in 14 of the 150 most popular screenshots uploaded to the *Caves of Qud* steam community page in the last six months[5]. Several screenshots featuring other aspects of the system, such as historic site or item descriptions, also appear in the top 150. Considering the subordinate role the system plays, we believe these results point to a positive reception.

As for the goal of producing narrative coherence, we believe the system is successful, but the results are less definitive for two main reasons. Firstly, we only recently added the in-game journal to track and juxtapose multiple gospels. Before this update, players would have to mentally juxtapose several gospels from a sultan's life for the narrative to cohere. Secondly, by the nature of the game's permadeath mechanic, only a small subset of player characters persist long enough to accrue several gospels from a single sultan. This means fewer candidate players to manifest their approval or disapproval.

Our debugging tools afford us a perspective unavailable to our players, namely, the ability to generate several sultans quickly and read through their gospels. Our satisfaction with the degree of narrative coherence can largely be attributed to time spent honing it through these tools. However, it's worth noting that no players have explicitly expressed dissatisfaction with the coherence of the generated narratives. Though this fact's contribution to an argument of positive reception may be limited, it does stand in contrast to some of our other procedural systems, whose results were explicitly criticized by our players. Finally, we include a reference to one standout player interpretation of a sultan's life [2]. The full text is omitted here due to its length, but it serves as an exceptional example of narrative coherence resulting from evocative output and the employment of apophenia.

6 FUTURE WORK

A few extensions to the system are obvious to us. Most immediately, the variety of output would be improved by expanding the set of domains (currently numbering ten) and the suite of historical events (currently numbering nineteen). In fact, the system was designed with this modularity in mind, and we've added new domains and events since the system's release. Secondly, the core presentation of history could be expanded beyond biographies by introducing events that affect other historical entities, such as locations and items, and generate corresponding gospels. This approach could be employed to construct a richer representation of history that extends beyond the lives of a few storied rulers. Thirdly, in further exploring the idea of history's rhetorical function, events could generate multiple, conflicting gospels from different perspectives. The history of a city's siege, for example, might read quite differently from the perspective of the sieged as it would from the perspective of the sieging party.

More broadly, we hope this approach gives a new perspective on historical modeling to both researchers and practitioners seeking

[5]http://steamcommunity.com/app/333640/screenshots/?p=1&browsefilter=trendsixmonths

to generate histories. By privileging historical accounts over the processes that produce them, the designers of history generation systems gain access to a whole space of generative methods. In particular, the technique of using existing state to rationalize narrative events after the fact may have application across the landscape of qualitative procedural generation.

7 ACKNOWLEDGEMENTS

We'd like to thank Gillian Smith for her preliminary review of this paper and Adam Summerville for his formatting guidance.

REFERENCES

[1] Tarn Adams and Zach Adams. 2006. Slaves to Armok: God of Blood Chapter II: Dwarf Fortress. Bay 12 Games.

[2] Angry Diplomat. 2017. Caves of Qud: My Axe Turtle's Tongue Rotted After Time in Golgotha's Cloaca. (2017). https://forums.somethingawful.com/showthread.php?threadid=3739217&pagenumber=141&perpage=40#post471677689

[3] Kate Compton. 2015. Tracery. (2015). https://github.com/galaxykate/tracery

[4] Bruno Dias. 2017. Voyageur. Failbetter Games.

[5] Joshua L. Diaz. 2009. Dwarf Fortress gathers at the statue and attends a party. (2009). http://dspace.mit.edu/handle/1721.1/54502#files-area

[6] Jason Grinblat and Charles B. Bucklew. 2010. Caves of Qud. Freehold Games.

[7] Mark R. Johnson. 2012. Ultim Ratio Regum.

[8] Mark R. Johnson. 2016. Towards Qualitative Procedural Generation. Computation Creativity and Games Workshop 2016.

[9] Stephen Lavelle. 2009. Opera Omnia. Increpare Games.

[10] Claude Lévi-Strauss. 1955. The Structural Study of Myth. Journal of American Folklore, Vol. 68, No. 278.

[11] Justus Robertson and Robert M. Young. 2014. Finding Schrödinger's Gun. Proceedings of Artificial Intelligence and Interactive Digitial Entertainment.

[12] Ben Samuel, James O. Ryan, and Adam J. Summerville. 2015. Bad News. Expressive Intelligence Studio.