Warm Rocks for Cold Lizards: Generating Meaningful Quests in *Caves of Qud*

Jason Grinblat, C. Brian Bucklew
Freehold Games
{jason,bbucklew}@freeholdgames.com

Dynamic quest generation is a topic of interest in the game AI community, with examples appearing in AAA games (Lenhardt 2012), indie games (Coxon 2016), and academia (Sullivan et al. 2012). In July 2018, we released an update to our simulative science fantasy roguelike *Caves of Qud* (Freehold Games 2015) that added its own take on dynamically generated villages and quests. These systems were built on top of a framework we previously developed for generating historical biographies (Grinblat and Bucklew 2017). Like that framework, our approach to dynamic quest generation—novel in the space as far as we know—was shaped primarily by two design forces at work in the project's development. The first is an architectural constraint; to support its large-scale game worlds, *Caves of Qud* uses a two-phase approach to world creation (Bucklew and Grinblat 2019) that delays the full fabrication of game "zones" until the player enters them. The quest generation system was necessarily designed to fit this two-tiered architecture. The second is an aesthetic choice of procedural content design; we tend to be nonprescriptive in deciding which of our content modules can be procedurally combined, preferring instead to "let our generators run wild," as we say, but ensure that they smooth the rough edges off their output. For our storyful generators, this means eschewing causal logic in favor of randomness that's parameterized by the richly narrative world the generators are embedded in. The resulting outputs are ripe for apophenic readings.

As mentioned, the quest generation work is split between the two phases of world creation, world-gen and zone-gen, plus some pre-authoring of templates. World-gen happens at the start of each game, where abstract representations of the game zones, their contents, and their relationships are generated. Zone-gen happens when a player enters a new zone and those abstractions are reified and fabricated as game objects. With this architectural context in mind, the components of the quest generation system are as follows:

1. A catalog of pre-authored quest templates (e.g. "DiscoverAnImportantLocation", "FindAnImportantItem")
2. An abstract DynamicQuestContext interface that zones implement in order to answer questions about themselves; those answers get used to populate details in the templates of quests generated inside the zones
3. A dynamic quest factory that, when asked for a quest originating in a zone, chooses a template and resolves its details with the context provided by that zone
4. A list of annotations that get sent to the zone building module for the requesting zone and any other zones that are affected by the quest
5. A replacement grammar for each quest template that, in conjunction with additional replacement rules for the fabricated objects, is used to generate quest names and dialog

Let's walk through an example. During world-gen, a village zone of moisture-farming reptiles wants to generate a dynamic quest for itself. It instantiates a class that implements DynamicQuestContext, where it specifies answers to relevant quest questions, like who are the potential quest givers (the most senior reptiles) and what's important to them (warm rocks). The quest factory randomly chooses the FindAnImportantItem template and populates the details with the reptile village's provided context, including the quest item (a quartz slab) and its location (another reptile village, randomly chosen from among nearby locations). An annotation is sent to the zone builder module that tells it to add quest-giving dialog to the quest givers, who are chosen from among the viable fabricated creatures during zone-gen. Another annotation is sent to the destination zone, telling the zone builder to make sure to place the quartz slab when it builds the zone. (This allows for a sequence break where the player arrives at the destination zone before having received the quest or even generating the source zone at all. The quartz slab is still there and retrievable.) Finally, when the player arrives at the source zone for the first time, the zone is generated, game objects such as the reptile villagers are fabricated, a quest giver is chosen based on the resolved quest context, and that quest giver is given appropriate dialog generated by the replacement grammar ("Hello, friend. We heard about this lovely warm rock in a neighboring village. Would you go retrieve it for us?").

References

Bucklew, C. B., and Grinblat, J. 2019. Math for Game Developers: End-to-End Procedural Generation in 'Caves of Qud'. https://www.gdcvault.com/browse/gdc-19/play/1026313.

Coxon, T. 2016. Quest Friends Forever. https://playstarbound.com/quest-friends-forever/.

Freehold Games. 2015. *Caves of Qud*. http://www.cavesofqud.com/.

Grinblat, J., and Bucklew, C. B. 2017. 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*.

Lenhardt, H. 2012. Bethesda's Nesmith reflects on the difficult birth of Skyrim's 'Radiant Story' system. https://venturebeat.com/2012/01/27/bethesdas-nesmith-reflects-on-the-difficult-birth-of-skyrims-radiant-story-system/.

Sullivan, A.; Grow, A.; Mateas, M.; and Wardrip-Fruin, N. 2012. The Design of *Mismanor*: Creating a Playable Quest-Based Story Game. In *Proceedings of the International Conference on the Foundations of Digital Games*.