Generating Side Quests from Building Blocks

Tomáš Hromada, Martin Černý (✉), Michal Bída, and Cyril Brom

Department of Software and Computer Science Education
Charles University in Prague, Czech Republic
`gyfis@seznam.cz, cerny.m@gmail.com, michal.bida@gmail.com`

Abstract. Computer games are an important application area for interactive storytelling. In a large subset of games, quests — tasks that the player is assigned to complete — are the primary driving forces of the storyline. The main storyline is usually accompanied by a number of optional side-quests. We present a system for generating side-quests based on chaining simple building blocks, akin to the branching narrative approach to interactive storytelling. Our primary interest was how far can we get with such a simple approach. The simplicity of our system also lets game designers retain more control over the space of possible side-quests, making the system more suitable for mainstream computer games. We implemented the system in an experimental game and compared the quests generated by the system with hand-picked and random sequences of building blocks. We performed two rounds of player evaluation ($N_1 = 21$, $N_2 = 12$), which has shown promising results.

Keywords: quests · computer games · role-playing games

1 Introduction

Open-world games (OWGs) form a large and popular subset of computer games. While there is not a universally accepted definition of OWGs, one of their defining properties is freedom: the actions of the player are relatively unconstrained, she may traverse the virtual world in a way she founds the most pleasant and has a (relatively) large pool of possible interactions with non-player characters (NPCs) and other entities inhabiting the game world. The high level of freedom has direct consequences for the way the game story is conveyed to the player. While some OWGs focus on sandbox-like experience with little or no story overarching the player's interaction with the game world, the typical solution is to structure the story into quests. Quest, in this context, is a task the player has to fulfill, usually assigned to the player by an NPC.

Structuring the main story into quests increases the feeling of agency as the player is free to choose when to attend to a specific quest and may spend a prolonged time exploring the world in between the quests. Most games have only one or very few possible main storylines. To make sure the game can be finished, the game does not usually let the player to reject most of the main quests and failure in any of the main quests makes the player lose the game.

To balance the reduced freedom of the main storylines, OWGs usually feature a large amount of optional *side-quests*. Side-quests give the player purpose to explore the world beyond the main storyline and as they are optional, the player may choose the side-quests that best match her playing style.

Generating side-quests is an interesting challenge for interactive storytelling (IS) and also a practical consideration from game development perspective. Many games choose to generate side-quests simply to reduce the workload, as a large number of side-quests are often required. In all cases we are aware of, side-quest generation in commercial games is a form of template instantiation: a premade scene is placed on a chosen location in the game world, possibly with some assets replaced (e.g., differently looking NPCs taking part in the crime).

In a more complex template setup, as in the S.T.A.L.K.E.R. series [8], the whole game world is continuously simulated and filled with NPCs that make their own decisions and the quest templates are instantiated in response to events emerging in the simulation. While this increases the diversity of possible situations the quests are executed in, the quests themselves are still instances of very simple templates (e.g., help a friendly group in a fight against an enemy group).

Commercial games do not use any complex forms of quest generation (e.g., planning-based IS systems), partly because those decrease the control the designers have over the system and thus increase the probability that the generated side-quests will be inappropriate to their context or interfere with the main storyline. For this reason, our focus is on simplicity of the system.

In this paper we present a system based on composing side-quests from premade building blocks, akin to the branching narrative approach to IS. Each building block represents a short event that takes place in the quest, including all in-game assets needed to enact the event and conditions for the player to successfully progress through the event. We argue that our system is a possible improvement over simple template based systems of the contemporary OWGs by having greater expressive power than templates while keeping tight designer control over the side-quest space. We evaluate the system by letting players play side-quests generated by an implementation of the system in a simple role-playing game (RPG). Although our work is intended for side-quests, we will speak generally about quests in the remainder of the paper for the sake of brevity.

2 Related Work

Multiple automated quest generating systems were described in the literature. The GrailGM system [16] uses a forward chaining inference engine to recombine quest elements given by a designer to form quests with multiple solutions and to guide the player's progression through the game. Petri Nets have also been proposed as models for generating quests [10].

In [5], authors examine 3000 quests from commercial games and present a generative grammar that covers most of the analyzed quests. This system was later connected to the game Everquest [6], although no evaluation has been performed so far. Another approach connected to an actual game is *Charbitat* where simple key-lock quests are integrated into a procedurally generated landscape [2].

In [11] authors present a planning-based approach to quest generation. The system is shown to be able to generate quest variations for a 2D RPG, but it is evaluated only with respect to speed.

There are also works on creating interactive experiences that were not intended to be quests, but could be adapted to become quests of a specific kind. This includes generating murder mysteries [1] or adventure-game puzzles [4].

Our approach is simpler than all of the aforementioned works. We consider this to be an advantage for practical game development as simplicity promotes designer control and the system can be quickly implemented. We are also not aware of any other work that evaluated generated quests on actual players.

The building blocks in our system roughly correspond to scenes as discussed by Mateas and Stern [13], or the dramatic beats in Façade [14]. Multiple works have investigated representation of stories as plans, where each action in the plan corresponds to a scene or event being enacted (e.g., [12, 15]). The blocks in our approach are similar to narrative actions in the plan-based approach to IS and our formalism could be reformulated as an extremely simple planning problem.

3 Building Blocks

One of our primary goals was to keep the system simple and controllable, while expanding the expressive power of template-based systems. To achieve both we decided to split quests into *blocks*. Each block represents a short section of a quest (e.g. killing a monster to obtain an item). The blocks are arranged into a directed acyclic graph (DAG) where edges connect each block to its possible successors. The DAG may be represented explicitly (as in our implementation) or it may be given implicitly by conditions. Blocks that have no incoming edges are called *start blocks* and those that have no outgoing edges are *end blocks*. Choosing blocks to form a quest is now simply the matter of choosing a random path from a start block to an end block. It is up to the designer to ensure that all paths in the DAG correspond to valid quests. Our experience however indicates that this is achievable with reasonable effort.

Once the blocks are chosen, the system lets the individual blocks to read data from other blocks in the sequence to make the quest coherent. For example a start block where an NPC asks the player to fetch a valuable item reads the name and location of the item from its successor block. The actual structure of the DAG has to ensure that for every possible path through the DAG, the necessary data are always present, but this can be easily checked automatically offline. To further increase variability, the individual blocks could be turned into templates (e.g., varying the specific type of a monster that protects the item).

Each block is also directly connected to assets in the game world, forming self-contained collections of in-game objects and behaviors they exhibit akin to smart objects [9] or smart areas [3]. This encapsulation further eases debugging as the blocks can be tested independently.

4 Our Implementation

To test the proposed approach we have created a simple first-person RPG in the Unity game engine[1]. Screenshots of the game are presented in Figure 1. Each run of the game presents the user with a single quest generated by our method. The game features some of the typical RPG mechanics: simple ranged combat, picking up items and giving them to NPCs and dialogues.[2] We implemented a total of 26 blocks, of those 9 are start blocks and 9 are end blocks. On average, there are four outgoing edges from each block. The DAG is defined explicitly by connecting each block to possible successors. Each block is also responsible for instantiating the objects it needs for execution in the game.

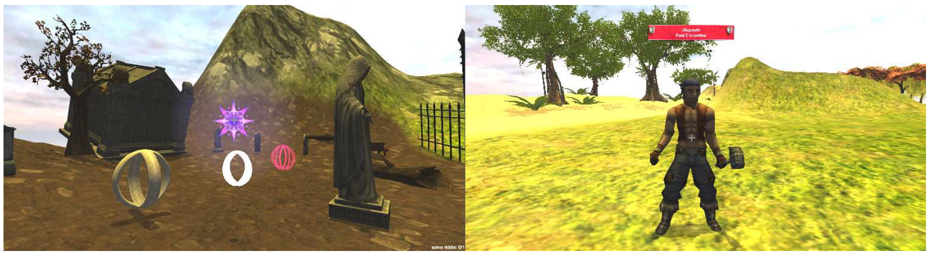

Fig. 1. Screenshots of the game we used for evaluation.

In our implementation, the start blocks represent exposure to the quest — there are multiple NPCs that give various motivations and context for the quests (helping the world, personal enrichment, . . .) and a surprise attack by skeletons. The end blocks represent conclusion of the quest (player returning an item to an NPC, player being betrayed and ambushed, player meditating and reflecting on the past experience, . . .). The blocks that are used in between represent the core of the tasks assigned to the player (finding items, chasing an NPC, fighting enemies, gathering information, . . .).

The blocks have several attributes (location, action, subject and object) that can be accessed by other blocks in the quest. In our implementation, these attributes serve only as strings that are inserted into the NPC's dialogue. Let us show this on an example of a generated quest: the start block shows a priest that has lost a valuable item and asks the player to bring it back to him. The successor that was chosen is a block where the player is asked to find diamonds scattered in a maple grove. In the chosen end block a mage asks the player to destroy an item she has brought, because the mage discovers it is cursed. Now the first blocks reads the object and location attributes of the second block ("diamonds" and "maple grove") and inserts them into a sentence describing what

[1] `https://unity3d.com`

[2] A playable version of the system can be downloaded at `http://gyfis.itch.io/side-quest-generating-system`

did the priest lost and where. It further reads the location and subject attributes of the last block ("mage Aaron" and "near the cemetery") and inserts them into a sentence describing where the player should deliver the items, once she founds them. The last block further reads the object attribute of the second block ("diamonds") and inserts it into a sentence thanking the player for bringing the item and into a sentence where the mage describes his discovery that it is cursed.

Our DAG represents 187 possible quests. A fully interconnected DAG with the same number of start, end and intermediate blocks would allow for 729 quests ($9 \times 8 \times 9$ of length three and 9×9 quests of length two), while combining the same amount of blocks into fixed quests would result in only 8 quests of length three or 13 quests of length two. This signals that the DAG structure does not overly limit the number of possible generated quests and that the system greatly increases the amount of quests over traditional methods. Since our blocks are not simple variants of a single concept, but truly different encounters for the player with different dialogues, game mechanics involved and motivations presented to the player, the template-based techniques for generating quests can be directly applied at block level to further increase variety.

5 Evaluation

We performed two rounds of evaluation. The first round engaged 21 high school students aged 15 — 22 (14 male, 7 female) in playing 8 quests in our game each. Five of those quests were generated (Gs), while three were hand-picked sequences of blocks (HPs) that were manually tweaked for maximal consistency. The Gs were different for each participant, while the HPs were the same.

After analyzing the first round, HPs showed to be a problematic baseline — they provided only an upper estimate of quality and did not let us asses the magnitude of the difference between HPs and Gs. So we performed a second round of evaluation with a random baseline. In the second round, 12 players aged 16 – 30 (7 male, 5 female) played 7 quests. Two of those were generated, two were HPs (randomly chosen from the three HPs from the first round) and three were randomly chosen sequences of a start block, an intermediate block and an end block that could not be generated by the system (Rs). The Gs and Rs were different for each participant.

For both rounds, the order of the quests was randomized. In total, 252 quest instances were played by the users. Due to bugs in the system, 7 invalid quest instances were removed from the results. In both rounds, players took 20 — 60 minutes to complete all quests.

After playing each quest, the participants were asked to evaluate the following statements on a 5-point Likert scale (strongly disagree to strongly agree): Q1) "I liked this quest", Q2) "The story was clear", Q3) "I knew what to do all the time during this mission" and Q4) "This quest was similar to another one I have already played" (during this session).

For all questions, the participants in the second round of evaluation viewed the system more positively — they were more likely to agree with the first 3 questions and disagree with question 4, but the differences were only slight. Thus for our analysis, we treat the two evaluation rounds as a single dataset, although the results for the random baseline should be interpreted with caution as they represent only 12 participants playing 3 quests each. The overall results for those questions are shown in Figure 2.

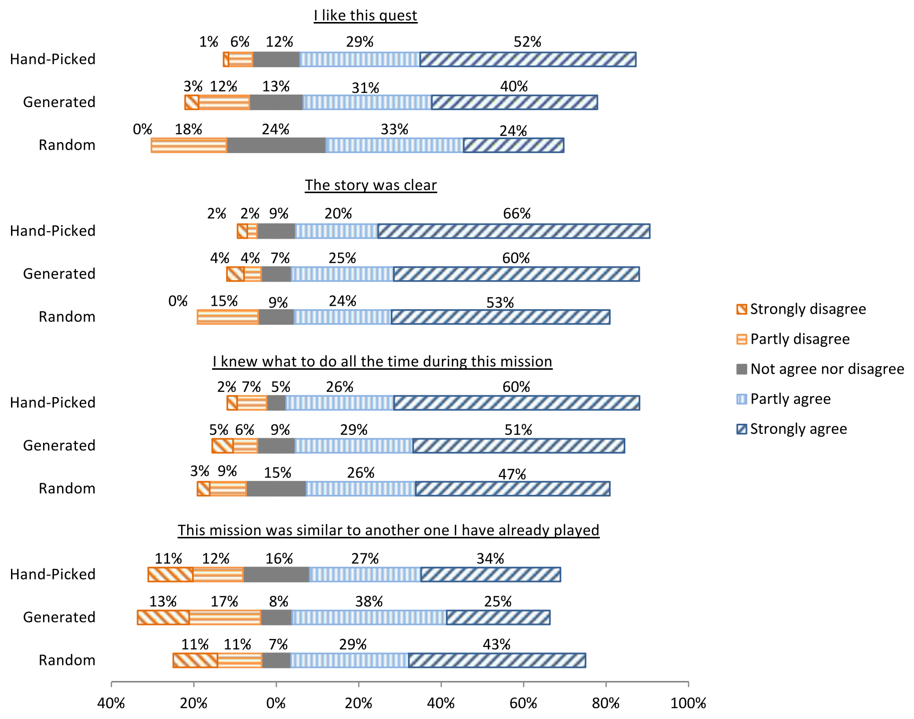

Fig. 2. Answers to the main questions of the poll, summarized over all quests. The percentages are rounded to nearest integer and thus do not always add up to 100%.

The answers to Q1 show a clear ordering. As expected, HPs were considered better than Gs ($p = 0.04$, Wilcoxon rank-sum test), while Rs were rated worse than Gs, although the difference is not significant ($p = 0.1$, Wilcoxon rank-sum test). The difference between HPs and Gs and between Gs and Rs is of approximately equal magnitude. For each player we further performed a pairwise comparison of ratings for Q1 for all quests she played. HPs were rated better or equal than Gs in 81% of the HP-G pairs and strictly better in 36% of the HP-G pairs, Gs were rated better or equal than Rs in 71% of the G-R pairs and strictly better in 44% of the G-R pairs. This indicates that HPs were indeed some of the best possible block combinations available, while Gs are slightly worse but still better than Rs. The differences are however small.

Answers to Q2 show that the differences in story coherence were even smaller (and not statistically significant), but retained the same ordering. Q3 lets us estimate the amount of noise introduced by players having trouble with the specific game mechanics of the game. For all categories, less than 12% of answers indicate some issues. The answers to Q4 (similarity to previous quests) are interesting, because the ratio of positive and negative answers is similar for Gs and HPs, but Gs have fewer neutral answers, although the link between the type of quest (HP or G) and the number of neutral answers is not statistically significant ($p = 0.09$, Pearson's χ^2). Further caution should be made when interpreting answers to Q4, as the participants initially reported low similarity, then their rating grows quickly and highest similarity was reported after completing the 4th quest and from then on, the players perceive the following quests as less and less similar. This could simply be noise — the correlation between answers to Q4 and the quest number is not significant, ($p = 0.06$, Pearson's χ^2). It is also possible that the perception of similarity evolves as players get familiar with the environment.

The answers to Q1 also evolve over time ($p = 0.049$, Pearson's χ^2). This is possibly related to quest similarity because participants' enjoyment of the game (Q1) was lowest when playing the first quest, possibly because some of the participants had problems understanding the mechanics, and peaked at 6th quest, which corresponds to the moment when, statistically, players start to see a lot of the content for the second time.

Further details on experiment design and source code can be found in the BSc. thesis of the first author [7]. Complete experimental results are also available.[3]

6 Conclusion and Future Work

We have presented a system for generating side-quests from premade building blocks. We have shown that although the system is simple, it can introduce a lot of variety to the input data and players rate the resulting side-quests positively and better than a random baseline but lower than a hand-picked baseline. Simplicity of the system is desirable as it promotes designer control and allows easy adoption for an existing game. We think that the accessibility of the system and the results presented in the paper show the system's potential to improve contemporary OWGs and pave the way for more massive usage of interactive storytelling techniques in the gaming industry. A shortcoming of our evaluation is that we did not evaluate how often a completely implausible quest is produced by the system, which is a very important metric in practice.

The system could be further improved by preferring blocks not yet seen by the player, or by using templates at the block level. Evaluating the system from the authoring perspective (e.g., comparing efficiency and quality of results authors achieve with our system and alternative quest generation paradigms) would make an interesting future work.

[3] `http://popelka.ms.mff.cuni.cz/~cerny/papers/sidequests-evaluation.zip`

Acknowledgments. This research is partially supported by the student grant GA UK No. 559813/2013/A-INF/MFF and by SVV project number 260 224.

References

1. Arinbjarnar, M.: Dynamic plot generation engine. In: Proceedings of the Workshop on Integrating Technologies for Interactive Stories. pp. 1–5. ICST, Brussels (2008)
2. Ashmore, C., Nitsche, M.: The quest in a generated world. In: Proceedings of the Digital Games Research Association Conference. pp. 503–509 (2007)
3. Cerny, M., Plch, T., Marko, M., Ondracek, P., Brom, C.: Smart areas: A modular approach to simulation of daily life in an open world video game. In: Proceedings of ICAART 2014. pp. 703–708. SCITEPRESS, Portugal (2014)
4. Dart, I., Nelson, M.J.: Smart terrain causality chains for adventure-game puzzle generation. In: Computational Intelligence and Games. pp. 328–334. IEEE Press (2012)
5. Doran, J., Parberry, I.: A prototype quest generator based on a structural analysis of quests from four MMORPGs. In: Proceedings of the 2nd International Workshop on Procedural Content Generation in Games. pp. 1–8. ACM (2011)
6. Doran, J., Parberry, I.: A server-side framework for the execution of procedurally generated quests in an MMORPG. Tech. rep., Department of Computer Science and Engineering, University of North Texas, USA (2015), available at `http://larc.unt.edu/techreports/LARC-2015-01.pdf`, Last Checked 2015-09-08
7. Hromada, T.: Generating Side Quests for RPG Games From Hand-Crafted Building Blocks. BSc. thesis, Charles University in Prague (2015)
8. Iassenev, D., Champandard, A.: A-Life, emergent AI and S.T.A.L.K.E.R. AIGameDev.com (2008), `http://aigamedev.com/open/interviews/stalker-alife/` Last checked 2015-06-02
9. Kallmann, M., Thalmann, D.: Modeling behaviors of interactive objects for real-time virtual environments. Journal of Visual Languages & Computing 13(2), 177–195 (2002)
10. Lee, Y.S., Cho, S.B.: Dynamic quest plot generation using Petri net planning. In: Proceedings of the Workshop at SIGGRAPH Asia. pp. 47–52. ACM (2012)
11. Soares de Lima, E., Feijó, B., Furtado, A.L.: Hierarchical generation of dynamic and nondeterministic quests in games. In: Proceedings of the 11th Conference on Advances in Computer Entertainment Technology. pp. 24–33. ACM (2014)
12. Magerko, B., Laird, J., Assanie, M., Kerfoot, A., Stokes, D.: AI characters and directors for interactive computer games. In: Proceedings of the 16th conference on Innovative applications of artifical intelligence. pp. 877–883. AAAI Press (2004)
13. Mateas, M., Stern, A.: Towards integrating plot and character for interactive drama. In: Socially Intelligent Agents, pp. 221–228. Springer, Heidelberg (2002)
14. Mateas, M., Stern, A.: Structuring content in the façade interactive drama architecture. In: Proceedings of the First Artificial Intelligence and Interactive Digital Entertainment Conference. pp. 93–98. AAAI Press (2005)
15. Riedl, M.O., Young, R.M.: Narrative planning: Balancing plot and character. Journal of Artificial Intelligence Research 39(1), 217–268 (2010)
16. Sullivan, A., Mateas, M., Wardrip-Fruin, N.: Rules of engagement: Moving beyond combat-based quests. In: Proceedings of the Intelligent Narrative Technologies III Workshop. pp. 11:1–11:8. ACM (2010)