Procedural Content Generation in Competitive Multiplayer Platform Games

Georg Volkmar(✉), Nikolas Mählmann, and Rainer Malaka

Digital Media Lab, TZI, University of Bremen, 28359 Bremen, Germany
{gvolkmar,nikmaehl}@uni-bremen.de, malaka@tzi.de
https://www.uni-bremen.de/

Abstract. Procedural content generation (PCG) techniques have become increasingly established over the years in the context of video games. In terms of generating level layouts, PCG has proven to be a cost-efficient alternative to handcrafted design processes. However, previous research is mostly concerned with singleplayer experiences only. Since multiplayer games differ strongly in regards to the requirements that have to be met by map layouts, we addressed the following question: Which PCG approaches are suited best to ensure qualities relevant in the area of level creation for competitive multiplayer platform games? We conclude that a combination of constructive and grammar-based methods serves as a viable solution. We developed a PCG prototype which was integrated into a competitive platform game. Results of a user study indicate that a constructive-grammar PCG algorithm can be used to generate map layouts that are perceived to be fun and compelling.

Keywords: Game design · Procedural content generation · Multiplayer platform games

1 Introduction

In recent years, the use of procedural content generation (PCG)[1] has risen in popularity, especially in the context of video game content generation [7]. PCG has become highly attractive for game development studios in terms of cost-efficiency since it offers a more time- and money-saving alternative [5]. Various elements of video games are suited to be generated procedurally. One established application domain in the field of video game related PCG is the construction of levels for platform games such as *Super Mario Bros.* [15,19]. However, previous research heavily focused on the utilization of such methods for singleplayer experiences. Since multiplayer games differ immensely in terms of structure and

[1] In the domain of games, procedural content generation can be defined as "the algorithmical creation of game content with limited or indirect user input." [20].

Published by Springer Nature Switzerland AG 2019
E. van der Spek et al. (Eds.): ICEC-JCSG 2019, LNCS 11863, pp. 228–234, 2019.
https://doi.org/10.1007/978-3-030-34644-7_18

gameplay, it is questionable whether these established methods can be applied without alteration. Thus, the first problem addressed in this paper is concerned with the identification of quality requirements for procedurally generated multiplayer platform maps. Additionally, we review a number of various commonly used PCG techniques in terms of suitability to satisfy these demands. Aside from covering the technical aspects of PCG-based level generation, we have to bear in mind, that games are ultimately developed to be enjoyable forms of entertainment. Though it has been shown that PCG can lead to higher replayability and new forms of gameplay [18], it remains unclear if an entertaining experience can be provided in the context of competitive multiplayer platform games. Therefore, we evaluate whether procedurally generated maps provide a satisfying player experience. This paper is a contribution to the research domain of procedural content generation and the empiric validation of PCG methods.

2 Identification of Quality Requirements

In this section, we compile a list of quality requirements that have to be met for the generator to be a viable solution. How to evaluate the quality of video games in a standardized way is a topic of debate as the consumption of games is highly subjective leading to different preferences based on personality and experience [10]. However, literature in the domain of video game design has brought up a variety of quality criteria that can be considered when evaluating games [1,2,6,8,13,14]. For the purpose of evaluating the quality of the procedurally generated maps used in a multiplayer platform game, we have derived a number of criteria from the mentioned literature:

Internal Completeness: In an internally complete game, the player never reaches a point that leads to compromising gameplay or functionality [6]. For that purpose, unreachable areas and dead ends[2] should be avoided in any case. This property is especially relevant in a multiplayer scenario since dead ends might lead to the feeling of unfair level design. Reaching a dead end as one player might give the opponent a competitive edge and the opportunity to strike the other player down.

Flow: Traditionally, the concept of *Flow* describes a mental state of being fully involved in an activity while the proportion of the task's challenge to the operator's skills is perfectly balanced [3]. In video games, *Flow* strongly correlates with immersion and player enjoyment [6,14]. In the context of multiplayer games, an applicable level generator shouldn't create obstacles that act as hindrances for the players and reduce *Flow*. Therefore, lethal gaps, spikes, traps and the likes should be used scarcely and gaps should be designed smaller than the maximum jumping range of players.

Risks and Rewards: Having meaningful and impactful choices spread throughout a game can increase player enjoyment immensely [6]. To have a positive effect,

[2] "A dead end occurs when a player gets stranded in the game and cannot continue toward the game objective [...]" [6].

choices need to be designed in proportion to *Risks and Rewards*. In competitive platform games, risks can be designed as gaps or spikes whereas rewards typically take the form of power-ups. Both should be spread over the map in a way that gives meaning to player choices (e.g. placing a power-up in an area that's hard to reach). For multiplayer games, *Risks and Rewards* face the challenge of providing a fair distribution of items such as power-ups.

Diversity: As mentioned before, increasing the level of replayability is one core motivational factor to utilize PCG methods. For this purpose, generated map layouts should provide a certain *Diversity* in level structure. For multiplayer games, this factor gains even more relevance as playing the same or similar map over and over again would lead to boredom and players using the same tactics that have proven to work on the map layout.

Performance: According to a recent survey conducted by *Electronic Entertainment Design and Research (EEDAR)*, 72% of the interviewed PC players stated that reasonable loading times are a key factor when it comes to enjoyment in playing video games [4]. In multiplayer games, players already spend many seconds or even minutes before the actual game starts by waiting for other players in a lobby or contents to be downloaded [11]. For the *Performance* requirement to be fulfilled, we define a threshold of 15 s, as loading times below that time span are generally considered "good" [9].

Determinism: Modern games such as *Minecraft* [12] allow players to share randomly generated maps with others by providing a seed. For competitive multiplayer games, *Determinism* plays a vital role as it allows to repeat a match on a specific map multiple times, ensuring fairness among all players for tournament-like game modes.

3 Review of PCG Methods for Level Generation

PCG techniques can be divided into the following categories: optimization, constraint satisfaction, grammars, content selection and constructive [18]. Algorithms based on optimization generate content iteratively based on quality evaluation. The generative process endures until a sufficient result is achieved [16]. The evaluation can be conducted by humans or the algorithm itself. If conducted by the algorithm, optimization is computationally expensive, especially if the generated content ought to be playable [21]. Since we have identified *Performance* as a vital requirement, optimization-based approaches might not be the perfect solution. Constraint satisfaction requires a level designer to formulate properties that a generated map should provide which in turn are fed into a constraint solver algorithm. This method of content generation has been utilized before to create solvable tile-based mazes [17] and might proof suitable for multiplayer platformers as well. Content selection describes a method of generating content by assembling smaller pieces into larger segments. Despite fulfilling the *Performance* requirement, it is rather questionable if generated maps will entail the needed level of *Diversity* as players might notice repeating patterns

[18]. Grammars can be used in the context of PCG with the help of production rules. If these rules are well defined, content does not require additional evaluation leading to an increased *Performance* [18]. Constructive generators combine building blocks that represent small pieces of content to create levels. Often, they are tailored to fit a specific game and do not need any additional evaluation, similar to grammars [18].

4 Prototype Implementation

From reviewing existing literature, we conclude that constraint satisfaction, grammars and constructive approaches may provide the qualities we identified as vital in multiplayer platformer levels. For a first prototype, we developed a constructive-grammar algorithm combining the advantages of both techniques in Haskell. As a basic principle, the approach described here fills up an empty map by combining pre-authored building blocks until the area is occupied entirely. As a starting point, we have defined the following set of building blocks or tiles, commonly used in platform games: *air* - players can move freely, *border* - define the frame of each level, *spike* - kill players upon contact, *surface* - provide platforms for players to stand on and finally *any* - tiles in an unfinished map that need to be filled by other types of tiles. With the help of these building blocks, we have created a standard empty map, consisting of a large area of *any* tiles, confined by *border* tiles. At the bottom, *spike* tiles are placed to kill players that fall off the screen. This basic layout served as an initial input for our PCG algorithm to build a level from. Next up, as the map shall be filled with sets of tiles, we had to define a collection of building blocks for the generator to choose from and combine. For this purpose, we have grouped standard tiles into larger segments that served as building blocks. Combining these blocks randomly can lead to malformed level structure which is clearly a violation of the *Internal Completeness* requirement we identified before. This is where grammars and production rules come into play. A single rule could be described as follows: "Building blocks are only to be inserted if they do not overwrite existing tiles in the process." While making sure that all rules are respected, the algorithm integrates building blocks into the map until no insertions can be performed anymore. As a final step in level generation, power-ups and player-spawns were spread over the map. Ultimately, the level generator was integrated into a competitive multiplayer game. The structure of the game was kept rather simple, two players spawn at dedicated positions and aim to hit each other with flying disks that have to be thrown in the other player's direction (see Fig. 1).

5 Evaluation

After the development of the constructive-grammar prototype had been concluded, we conducted an evaluation based on quality requirements that were defined beforehand.

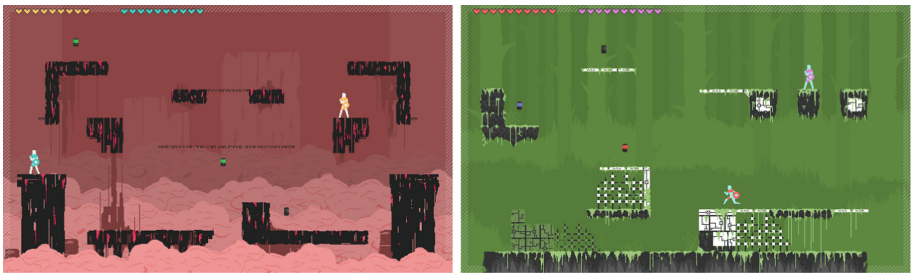

Fig. 1. Screenshots taken from *Diskophoros*, which was played in the study.

5.1 Performance Assessment

To get quantitative measures rating performance, we ran the algorithm 1000 times and noted the time individually for each round. Execution times varied between 0.43 and 6.47 s ($M = 0.43$) while about 99% of recorded times lie below 3 s. Since we had defined the threshold of operation time as a maximum of 15 s, we argue that our approach fulfills the *Performance* requirement sufficiently.

5.2 Explorative Pre-study

To get a first insight regarding the remaining requirements *Flow*, *Risk and Rewards* and *Diversity*, we conducted a small-scale user study. For this purpose, we integrated the constructive-grammar PCG algorithm into the existing multiplayer platform game *Diskophoros*.

Participants. The experiment consisted of two trials in which two subjects competed against each other. In the first group, two male software developers, aged 39 and 30 years participated. The second group contained two male students, both aged 22 years studying computer science and teaching.

Procedure. Before being exposed to procedurally generated maps, subjects were playing a test run on handcrafted levels. When they felt comfortable with the game's controls and overall gameplay, participants were asked to play on random maps, created by the constructive-grammar algorithm, for 20 min. During the procedure, the examiner took notes of observations and feedback given by the players. Ultimately, all subjects filled in a short questionnaire.

Material. The study was conducted using a PC with Windows 10 and standard game controllers. For data collection, a questionnaire was compiled that contained questions aimed to examine certain qualities of the game such as *Diversity*, *Flow*, overall impression and wishes for improvements.

Results. Three participants explicitly stated that they liked the game in terms of *Diversity*. No subjects mentioned monotony or repetition to be a problem. All players agreed that the game felt enjoyable and playable. As for suggestions

for improvement, some pointed out that gaps between platforms were sometimes too large and that maps felt a little empty. Moreover, players criticized cavities that emerged in some maps leading to unescapable death traps and a distortion of *Flow*. Being asked regarding the collection of power-ups, all subjects agreed that they didn't feel like they had to pay extra attention to them and just picked them up. Hence, *Risks and Rewards* felt like they were balanced properly.

Discussion. We need to address some limitations of the study and its reliability in general. It is important to note that this user study served as a first evaluation of the constructive-grammar approach in terms of playability. The objective of this study was to gather qualitative data in order to collect user insights that shall help to improve the map generator. It should be taken into consideration that our study involved a very small sample which is sufficient to obtain qualitative data but shouldn't be misinterpreted as a representative group. Additionally, since only male subjects took part in the study, results are gender-biased, potentially leading to skewed effects.

6 Conclusion and Future Work

The aim of this paper was to examine the applicability of PCG techniques in the context of multiplayer platform games under consideration of quality criteria that are required in this genre. For the purpose of addressing this topic, we identified a set of quality requirements (*Internal Completeness*, *Flow*, *Risks and Rewards*, *Diversity*, *Performance* and *Determinism*) and concluded that a combination of constructive and grammar-based methods have the potential to satisfy those needs. Furthermore, we investigated if PCG-based levels offer a satisfying player experience in a competitive multiplayer platform game. To analyze this problem, we conducted an explorative user study and came to realize that procedurally generated map layouts can indeed provide a fun and engaging experience for the players involved.

Concerning future developments, we plan to address the following remaining issues. Regarding the empiric validation of our approach, we rely on a few heavily gender-biased samples, limiting the reliability of the user study immensely. On top of that, only qualitative data has been gathered so far. Therefore, we will conduct another study involving more participants and standardized questionnaires to compare PCG-based levels with hand-crafted ones.

Acknowledgements. We would like to thank all participants of the study for their time and effort. This work was partially funded by the EU-project *first.stage*.

References

1. Costkyan, G.: I have no words & i must design: toward a critical vocabulary for games. In: Proceedings of the Computer Games and Digital Cultures Conference, Finland (2002)
2. Crawford, C.: The art of computer game design (1984)

3. Csikszentmihalyi, M., Csikszentmihalyi, I.: Beyond Boredom and Anxiety, vol. 721. Jossey-Bass, San Francisco (1975)
4. Electronic Entertainment Design and Research: Gaming infrastructure survey (2017). https://www.akamai.com/uk/en/multimedia/documents/report/akamai-eedar-gaming-infrastructure-survey.pdf
5. Franklin-Wallis, O.: Games of the future will be developed by algorithms, not humans, January 2016. https://www.wired.co.uk/article/games-developed-by-algorithms
6. Fullerton, T.: Game Design Workshop: A Playcentric Approach to Creating Innovative Games. AK Peters/CRC Press, Natick (2014)
7. Hendrikx, M., Meijer, S., Van Der Velden, J., Iosup, A.: Procedural content generation for games: a survey. ACM Trans. Multimed. Comput. Commun. Appl. (TOMM) **9**(1), 1 (2013)
8. Hunicke, R., LeBlanc, M., Zubek, R.: MDA: a formal approach to game design and game research. In: Proceedings of the AAAI Workshop on Challenges in Game AI, vol. 4, p. 1722 (2004)
9. Ip, B., Jacobs, G.: Quantifying game design. Des. Stud. **25**(6), 607–624 (2004)
10. Johnson, D., Gardner, J.: Personality, motivation and video games. In: Proceedings of the 22nd Conference of the Computer-Human Interaction Special Interest Group of Australia on Computer-Human Interaction, pp. 276–279. ACM (2010)
11. King, D., Delfabbro, P., Griffiths, M.: Video game structural characteristics: a new psychological taxonomy. Int. J. Ment. Health Addict. **8**(1), 90–106 (2010)
12. Mojang: Minecraft. Game [PC], May 2009
13. Salen, K., Tekinbaş, K.S., Zimmerman, E.: Rules of Play: Game Design Fundamentals. MIT Press, Cambridge (2004)
14. Schell, J.: The Art of Game Design: A Book of Lenses. AK Peters/CRC Press, Natick (2014)
15. Shaker, N., Nicolau, M., Yannakakis, G.N., Togelius, J., O'neill, M.: Evolving levels for Super Mario Bros using grammatical evolution. In: 2012 IEEE Conference on Computational Intelligence and Games (CIG), pp. 304–311. IEEE (2012)
16. Shaker, N., Togelius, J., Nelson, M.J.: Procedural Content Generation in Games. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-42716-4
17. Smith, A.M., Mateas, M.: Answer set programming for procedural content generation: a design space approach. IEEE Trans. Comput. Intell. AI Games **3**(3), 187–200 (2011)
18. Smith, G.: Understanding procedural content generation: a design-centric analysis of the role of PCG in games. In: Proceedings of the 32nd Annual ACM Conference on Human Factors in Computing Systems, pp. 917–926. ACM (2014)
19. Snodgrass, S., Ontañón, S.: Player movement models for video game level generation. In: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, pp. 757–763 (2017). https://doi.org/10.24963/ijcai.2017/105
20. Togelius, J., Kastbjerg, E., Schedl, D., Yannakakis, G.N.: What is procedural content generation?: Mario on the borderline. In: Proceedings of the 2nd International Workshop on Procedural Content Generation in Games, p. 3. ACM (2011)
21. Togelius, J., Yannakakis, G.N., Stanley, K.O., Browne, C.: Search-based procedural content generation: a taxonomy and survey. IEEE Trans. Comput. Intell. AI Games **3**(3), 172–186 (2011)