An Experiment on Content Generation of Game Software Engineering

Anonymous Author(s)

ABSTRACT

Background Video games are complex projects that involve a seamless integration of art and software during the development process to compound the final product. In the creation of a video game, software is fundamental as it governs the behavior and attributes that shape the player's experience within the game. When assessing the quality of a video game, one needs to consider specific quality aspects, namely 'design', 'difficulty', 'fun', and 'immersibility', which are not considered for traditional software. On the other hand, there are not well- established best practice for the empirical assessment of video game as instead there are for the empirical evaluation of more traditional software. **Aims** Our goal is to carry out a rigorous empirical evaluation of the latest proposals to automatically generate content for videogames following best practise established for traditional software. Specifically, we compare Procedural Content Generation (PCG) and Reuse-based Content Generation (RCG). Our study also considers the perception of players and professional developers on the content generation. **Method** We conducted a controlled experiment where human-subjects had to play with and evaluate content automatically generated for a commercial video-game by the two techniques (PCG and RCG) based on specific quality aspects of video games. 44 subjects including professional developers and players participated in our experiment. **Results** The results suggest that RCG generates content of higher quality than PCG which is more aligned with the pre-existent content. **Conclusions** The results can turn the tides for content generation. RCG has been underexplored so far because the reuse factor of RCG is perceived as repetition by the developers, who ultimately want to avoid repetition in their video games as much as possible. However, our study revealed that using RCG unlocks latent content that is actually favoured by players and developers.

KEYWORDS

Empirical Study, Automated Software Transplantation, Procedural Content Generation

ACM Reference Format:

Anonymous Author(s). 2024. An Experiment on Content Generation of Game Software Engineering. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM. . . $15.00

https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Video games industry is in continuous growth every year [38]. Despite being one of the fastest growing industries, video game software engineering has been identified as an area of knowledge that needs more fundamental research [2, 12]. One of the areas where video game software engineering needs more research are empirical research methods [2].

While theoretical frameworks provide foundational understanding, empirical studies offer the necessary validation and refinement crucial for effective implementation. As in other disciplines dealing with human behaviour (e.g., social sciences or psychology), empirical research allows building a reliable knowledge base in software engineering [43, 50]. By empirically investigating the user experience of video game techniques, researchers can illuminate both the strengths and limitations of existing approaches, paving the way for advancements that align more closely with the diverse needs and preferences of players.

One of the video game development challenges is the need of content [45]. Content generation is often a slow, laborious, costly, and error-prone process. This results in issues such as significant delays in content development [28, 49] and the growing need for game content from post-launch updates. Through rigorous experimentation, empirical studies can serve as the cornerstone for fostering innovation and pushing the boundaries of what is achievable within content generation.

In this work, we aim to evaluate empirically two different video game content generation techniques along with two different users profiles (players and developers). We study the feasibility of Reuse-based Content Generation (RCG), and Procedural Content Generation (PCG), and whether they have an impact on the quality of the generated content. We do so by analyzing Kromaia, a commercial video game released on PlayStation 4 and Steam.

We present an experiment in which we compare content generated by RCG and PCG, in terms of video game specific measures 'design', 'difficulty', 'fun', and 'immersibility'. A total of 44 subjects performed the tasks of the experiment, assessing the generated content in two scenarios of Kromaia. We conduct three distinct sessions, one for players and the other two for developers, in order to investigate whether the profile of the participants assessing video games influences their perception.

Our results suggest that RCG generate content of higher quality than PCG which is more aligned with the pre-existent content. **TODO** *add more conclusions from the results.*

The structure of this paper is as follows. Section 2 reviews the related works in the area. Section 3 presents the approaches under study and the context of the experiment, Kromaia. Section **??** outlines the experimental design. Section **??** presents the experiment results, followed by a discussion in Section 4. Section **??** summarizes the threats to the validity. Finally, Section **??** concludes the paper.

Table 1: Measurements: Design (De), Difficulty (Diff), Fun (F), Human Made (HM), Immersibility (I). Evaluation of the content generated by the proposed algorithm (A), variants of the proposed algorithm (VA), or the proposed algorithm compared to a baseline (C).

Work Year	Evaluation	Measurements	Hypothesis & Validity	Statistical Analysis	Replication Package	Sample size
Cardamone *et al.* [10] 2011	VA	De	✗	✗	✗	5 players
Plans *et al.* [34] 2012	A	F	✗	✓	✗	31 players
Adrian *et al.* [1] 2013	VA	De, Diff, F	✗	✗	✗	22 players
Dahlskog *et al.* [14] 2013	VA	De, Diff, F	✗	✗	✗	24 players
Togelius *et al.* [44] 2013	A	De, Diff, F	✓	✓	✗	147 players
Gravina *et al.* [21] 2015	A	F	✗	✗	✗	35 players
Kaidan *et al.* [24] 2015	VA	De	✗	✗	✗	12 players
Olsted *et al.* [31] 2015	VA	De	✗	✗	✗	13 players
Prasetya *et al.* [37] 2016	C	F	✗	✗	✗	33 players
Ferreira *et al.* [17] 2017	VA	De, Diff, F, I	✗	✓	✗	139 players
Charity *et al.* [11] 2020	A	De, Diff	✗	✗	✗	2 players
Lopez-Rodriguez *et al.* [27] 2020	VA	Diff	✗	✗	✗	30 players
Kraner *et al.* [25] 2021	A	De	✗	✗	✗	5 players
Pereira *et al.* [32] 2021	VA	Diff, F, HM	✗	✓	✗	70 players
Pereira *et al.* [33] 2021	C	Diff, F	✗	✓	✗	16 players
Brown *et al.* [8] 2022	A	De	✗	✗	✗	35 players
De Lima *et al.* [16] 2022	A	HM	✗	✓	✗	38 players
Our work	PCG vs RCG	De, Diff, F, I	✓	✓	✓	32 players + 12 developers

2 RELATED WORK

Experimentation in software engineering is a practice that has been studied for decades [6]. Throughout time, researches have adopted established guidelines to be rigorous [50], such as the use of hypothesis, validity, statistical analysis or replication packages.

Content generation is a large field [51]. The types of content generated are diverse, such as vegetation [29], sound [34], terrain [19], Non-Playable Characters [48], dungeons [47], puzzles [15], and even the rules of a game [9]. However, it is difficult to find experiments with human-subjects that compare approaches [3].

Table 1 shows content generation work with human-subjects. In content generation, it is common that experiments with human subjects explore the quality of the generated content [8, 44] or different variants of the proposed approach [1, 33]. On other hand, work such us Pereira *et al.* [32] or Prasetya *et al.* [37] compared the generated content by their approach to a baseline. In this work, we compare two techniques for generating content that the community uses without any previous experiments to compare them.

In terms of measurements, studies have been conducted to examine the distinctive characteristics of video games [39]. Studies have investigated subjects, more precisely players, preferences and perceptions regarding various aspects of video games, including design [24, 31], difficulty [27, 32], or fun [34, 37]. Another aspect of video games is the user engagement and immersion, which plays crucial roles in shaping the overall gaming experience [23]. Our work considers all these measurements simultaneously.

Table 1 shows that none of the previous work is compliant with the practices adopted in experiments by traditional software. In fact, 65% have neither hypothesis and validity, statistical analysis nor replication package. Our work aims to compare with empirical rigour the content generated. To do so, we adopted traditional software guidelines for experimentation.

Thus far, previous work has only used players to evaluate content. In other words, they have not considered the perception of the developers themselves. We study not only the players assessment, but also the point of view of professional video game developers, and their differences when assessing the quality of the generated content.

3 BACKGROUND

In this section, we present the importance of software in video game development, the generation of content for video games, and the real-world context that we make use of on our experiment to perform the corresponding tasks.

3.1 Software in video games

The development process of video games requires a harmonious combination of artistic elements and software integration, resulting in intricate and multifaceted creations. Software plays a crucial role in every aspect of a video game's creation as it dictates the behavior and features that can be seen or experienced within the game. For instance, software is responsible for controlling the logic behind the behaviors of non-playable characters (NPCs) in a game. As video games evolve and become more sophisticated, the software powering them also becomes increasingly intricate.

Nowadays, most video games are developed by means of game engines. One can argue that game engines are software frameworks [36]. Game engines integrate a graphics engine and a physics engine as well as tools for both to accelerate development. The most popular ones are Unity and Unreal Engine, but it is also possible for a studio to make its own specific engine (e.g., CryEngine [13]).

One key artefact of game engines are software models. These are software models such as those proposed by the Model Driven Development paradigm [41] which should not be confused with either 3D Meshes or AI Models. Unreal proposes Unreal Blueprints [7], Unity proposes Unity Visual Scripting [40], and a recent survey in Model-Driven Game Development [52] reveals that UML and Domain Specific Language (DSL) models are also being adopted by development teams. Developers can use the software models to create video game content instead of using the traditional coding approach (C++ on Unreal or C# on Unity). While code allows for more control over the content, software models raise the abstraction level, thus promoting the use of domain concepts and minimizing implementation and technological details.

3.2 Content Generation for Video Games

The process of content generation for video games is typically slow, tedious, expensive, and susceptible to errors. Thus, leading to problems that the industry have such us: (1) excessive delays in content creation (with notorious examples in Cyberpunk 2077 [49] or GTA VI [28]) or (2) the ever-increasing demand for game content derived from post-launch updates, Downloadable Content (DLCs), games as a service, or platform-exclusive content.

To address these challenges, researchers have been exploring procedural content generation techniques as a potential solution to (semi)automate the generation of new content within video games [22]. Procedural content generation can be grouped in three main categories according to the survey by Barriga *et al.* [5]: Traditional techniques that generate content under a procedure without evaluation; Machine Learning techniques [26, 42] that train models to generate new content; and Search-Based techniques [46] that generate content through a search on a predefined space guided by a meta-heuristic using one or more objective functions.

Content can also be created through reuse. In fact, since the term software engineering was coined at the NATO Conference held in

Garmisch in 1968 [30], its evolution has been tied to the concept of reuse. Either applying an opportunistic approach such as clone-and-own [18], or applying systematic approaches as software product lines (assembling predefined features) [35] or as software transplantation (a feature is transplanted from a donor to a host) [4]. A recent SLR on game software engineering [12] identifies the relevance of both Reuse-based Content Generation (RCG) and Procedural Content Generation (PCG).

3.3 Kromaia Video Game for the Experiment

Kromaia is a commercial video game released on Playstation and Steam, translated into eight languages. On Kromaia, each level consists of a three-dimensional space where a player-controlled spaceship has to fly from a starting point to a target destination, reaching the goal before being destroyed. The gameplay experience involves exploring floating structures, avoiding asteroids, and finding items along the route, while basic enemies try to damage the spaceship by firing projectiles. If the player manages to reach the destination, the ultimate antagonist corresponding to that level (which is referred to as *boss*) appears and must be defeated in order to complete the level.

In the context of Kromaia, developers generate content through PCG by means of the work of Gallota *et al.* (which combines an L-system with an evolutionary Algorithm) [20] because it is specific for spaceships that can play the role of bosses, and it achieves the best state-of-the-art results for this type of content. Developers also generate content through RCG by means of reusing features between Kromaia´s content. Specifically, the developers select a feature (a fragment of content) from a donor, and a host (another content) that will receive the feature. Despite the research efforts in both PCG and RCG and the importance of content generation for video game development, there is no study that directly compares them.

4 DISCUSSION

In the context of video games, reuse is not perceived as a completely positive practice. Developers fear that reusing might be perceived as repetitive by players. On the other hand, the randomness of PCG is perceived positively as an extension in the range of the creativity space for new content. Our experiment shows that this negative view of reuse is not aligned with the results. On the contrary, it reinforces the RCG pathway which boosts the latent content and leads to better results than PCG. During the focus group, subjects agree on that RCG was a natural evolution of the original content. In contrast, PCG was negatively classified as content that did not appear to have been developed by professional developers.

Previous studies considered only players as the subjects of the experiments. In our experiment, we go one step ahead and analyse the differences between players and developers. For researchers it can be difficult to find developers to run experiments. However, that could not be the case for development studios. For instance, a large studio can enroll developers from different projects from the studio. This is relevant for studios because they put a lot of effort into enrolling players (not developers) for their games. It may seem paradoxical that it is hard to find players, but the experience of testing parts of a game in development is not the same as testing a full game as the developers in the focus group pointed out. Our experiment reveals that there are no relevant differences in terms of statistical values between players and developers, suggesting that studios can leverage their developers. Furthermore, when it comes to feedback developers provided more beneficial feedback as the focus group acknowledge.

This experiment combines the specific quality aspects of video games ('design', 'difficulty', 'fun', and 'immersibility') and the rigorousness of more traditional software work. This includes the replication package that we have not found in previous work. One may think that the complexity of video games makes it difficult to design packages for replication. Nevertheless, we expect that our work along with the replication package available will provide a basis and inspiration for future researchers of the game software engineering community.

Availability Replication package is at:

Acknowledgements Omitted for blind review.

REFERENCES

[1] Diaz-Furlong Hector Adrian and Solis-Gonzalez Cosio Ana Luisa. 2013. An approach to level design using procedural content generation and difficulty curves. In *2013 IEEE Conference on Computational Inteligence in Games (CIG)*. IEEE, 1–8.

[2] Apostolos Ampatzoglou and Ioannis Stamelos. 2010. Software engineering research for computer games: A systematic review. *Information and Software Technology* 52, 9 (2010), 888–901.

[3] Apostolos Ampatzoglou and Ioannis Stamelos. 2010. Software engineering research for computer games: A systematic review. *Information and Software Technology* 52, 9 (2010), 888–901.

[4] Earl T Barr, Mark Harman, Yue Jia, Alexandru Marginean, and Justyna Petke. 2015. Automated software transplantation. In *Proceedings of the 2015 International Symposium on Software Testing and Analysis*. 257–269.

[5] Nicolas A. Barriga. 2019. A Short Introduction to Procedural Content Generation Algorithms for Videogames. *International Journal on Artificial Intelligence Tools* 28, 2 (2019), 1–11. https://doi.org/10.1142/S0218213019300011

[6] Victor R. Basili and H. Dieter Rombach. 1988. The TAME Project: Towards Improvement-Oriented Software Environments. *IEEE Transactions on Software Engineering* (1988).

[7] Unreal Blueprint. [n. d.]. Unreal Blueprint. https://docs.unrealengine.com/4.27/en-US/ProgrammingAndScripting/Blueprints/GettingStarted/. Accessed: 01/02/24.

[8] Joseph Alexander Brown and Marco Scirea. 2022. Evolving Woodland Camouflage. *IEEE Transactions on Games* (2022).

[9] Cameron Bolitho Browne. 2008. *Automatic generation and evaluation of recombination games*. Ph. D. Dissertation. Queensland University of Technology.

[10] Luigi Cardamone, Daniele Loiacono, and Pier Luca Lanzi. 2011. Interactive evolution for the procedural generation of tracks in a high-end racing game. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*. 395–402.

[11] Megan Charity, Ahmed Khalifa, and Julian Togelius. 2020. Baba is y'all: Collaborative mixed-initiative level design. In *2020 IEEE Conference on Games (CoG)*. IEEE, 542–549.

[12] Jorge Chueca, Javier Verón, Jaime Font, Francisca Pérez, and Carlos Cetina. 2023. The consolidation of game software engineering: A systematic literature review of software engineering for industry-scale computer games. *Information and Software Technology* (2023), 107330.

[13] CryEngine. [n. d.]. CryEngine. https://www.cryengine.com. Accessed: 01/02/24.

[14] Steve Dahlskog and Julian Togelius. 2013. Patterns as objectives for level generation. In *Design Patterns in Games (DPG), Chania, Crete, Greece (2013)*. ACM Digital Library.

[15] Edirlei Soares de Lima, Bruno Feijó, and Antonio L Furtado. 2019. Procedural Generation of Quests for Games Using Genetic Algorithms and Automated Planning.. In *SBGames*. 144–153.

[16] Edirlei Soares de Lima, Bruno Feijó, and Antonio L Furtado. 2022. Procedural generation of branching quests for games. *Entertainment Computing* 43 (2022), 100491.

[17] Lucas Nascimento Ferreira and Claudio Fabiano Motta Toledo. 2017. Tanager: A generator of feasible and engaging levels for Angry Birds. *IEEE Transactions on Games* 10, 3 (2017), 304–316.

[18] Stefan Fischer, Lukas Linsbauer, Roberto E Lopez-Herrejon, and Alexander Egyed. 2015. The ECCO tool: Extraction and composition for clone-and-own. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, Vol. 2. IEEE, 665–668.

[19] Miguel Frade, Francisco Fernandéz de Vega, Carlos Cotta, et al. 2009. Breeding terrains with genetic terrain programming: the evolution of terrain generators. *International Journal of Computer Games Technology* 2009 (2009).

[20] Roberto Gallotta, Kai Arulkumaran, and LB Soros. 2022. Evolving spaceships with a hybrid L-system constrained optimisation evolutionary algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. 711–714.

[21] Daniele Gravina and Daniele Loiacono. 2015. Procedural weapons generation for Unreal Tournament III. In *2015 IEEE Games entertainment media conference (GEM)*. IEEE, 1–8.

[22] Mark Hendrikx, Sebastiaan Meijer, Joeri Van Der Velden, and Alexandru Iosup. 2013. Procedural content generation for games: A survey. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 9, 1 (2013), 1–22.

[23] Charlene Jennett, Anna L Cox, Paul Cairns, Samira Dhoparee, Andrew Epps, Tim Tijs, and Alison Walton. 2008. Measuring and defining the experience of immersion in games. *International journal of human-computer studies* 66, 9 (2008), 641–661.

[24] Misaki Kaidan, Chun Yin Chu, Tomohiro Harada, and Ruck Thawonmas. 2015. Procedural generation of angry birds levels that adapt to the player's skills using genetic algorithm. In *2015 IEEE 4th Global Conference on Consumer Electronics (GCCE)*. IEEE, 535–536.

[25] Vid Kraner, Iztok Fister Jr, and Lucija Brezočnik. 2021. Procedural content generation of custom tower defense game using genetic algorithms. In *International Conference "New Technologies, Development and Applications"*. Springer, 493–503.

[26] Jialin Liu, Sam Snodgrass, Ahmed Khalifa, Sebastian Risi, Georgios N Yannakakis, and Julian Togelius. 2021. Deep learning for procedural content generation. *Neural Computing and Applications* 33, 1 (2021), 19–37.

[27] Carlos López-Rodríguez, Antonio J Fernández-Leiva, Raúl Lara-Cabrera, Antonio M Mora, and Pablo García-Sánchez. 2020. Checking the Difficulty of Evolutionary-Generated Maps in a N-Body Inspired Mobile Game. In *International Conference on Optimization and Learning*. Springer, 206–215.

[28] Tuhin Das Mahapatra. 2023. Why is Rockstar delaying GTA 6? Here are some possible breakdowns. https://www.hindustantimes.com/technology/why-is-rockstar-delaying-gta-6-here-are-some-possible-breakdowns-101681440818791.html. Accessed: 01/02/24.

[29] Carlos Mora, Sandra Jardim, and Jorge Valente. 2021. Flora Generation and Evolution Algorithm for Virtual Environments. In *2021 16th Iberian Conference on Information Systems and Technologies (CISTI)*. IEEE, 1–6.

[30] Peter Naur and Brian Randell. 1969. *Software Engineering: Report of a conference sponsored by the NATO Science Committee, Garmisch, Germany, 7-11 Oct. 1968, Brussels, Scientific Affairs Division, NATO*.

[31] Peter Thorup Ølsted, Benjamin Ma, and Sebastian Risi. 2015. Interactive evolution of levels for a competitive multiplayer FPS. In *2015 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 1527–1534.

[32] Leonardo Tortoro Pereira, Paulo Victor de Souza Prado, Rafael Miranda Lopes, and Claudio Fabiano Motta Toledo. 2021. Procedural generation of dungeons' maps and locked-door missions through an evolutionary algorithm validated with players. *Expert Systems with Applications* 180 (2021), 115009.

[33] Leonardo T Pereira, Breno MF Viana, and Claudio FM Toledo. 2021. Procedural enemy generation through parallel evolutionary algorithm. In *2021 20th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*. IEEE, 126–135.

[34] David Plans and Davide Morelli. 2012. Experience-driven procedural music generation for games. *IEEE Transactions on Computational Intelligence and AI in Games* 4, 3 (2012), 192–198.

[35] Klaus Pohl and Andreas Metzger. 2018. Software product lines. *The Essence of Software Engineering* (2018), 185–201.

[36] Cristiano Politowski, Fabio Petrillo, João Eduardo Montandon, Marco Tulio Valente, and Yann-Gaël Guéhéneuc. 2021. Are game engines software frameworks? A three-perspective study. *Journal of Systems and Software* 171 (2021), 110846.

[37] Hafizh Adi Prasetya and Nur Ulfa Maulidevi. 2016. Search-based Procedural Content Generation for Race Tracks in Video Games. *International Journal on Electrical Engineering & Informatics* 8, 4 (2016).

[38] Piotr Rykała. 2020. The growth of the gaming industry in the context of creative industries. *Biblioteka Regionalisty* 20 (2020), 124–136.

[39] Ronnie ES Santos, Cleyton VC Magalhães, Luiz Fernando Capretz, Jorge S Correia-Neto, Fabio QB da Silva, and Abdelrahman Saher. 2018. Computer games are serious business and so is their quality: particularities of software testing in game development from the perspective of practitioners. In *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. 1–10.

[40] Unity Scripting. [n. d.]. Unity Scripting. https://unity.com/features/unity-visual-scripting. Accessed: 01/02/24.

[41] Bran Selic. 2003. The pragmatics of model-driven development. *IEEE software* 20, 5 (2003), 19–25.

[42] Adam Summerville, Sam Snodgrass, Matthew Guzdial, Christoffer Holmgard, Amy K. Hoover, Aaron Isaksen, Andy Nealen, and Julian Togelius. 2018. Procedural Content Generation via Machine Learning (PCGML). *IEEE Transactions on Games* 10, 3 (2018), 257–270. https://doi.org/10.1109/tg.2018.2846639 arXiv:1702.00539

[43] Walter F Tichy. 1998. Should computer scientists experiment more? *Computer* 31, 5 (1998), 32–40.

[44] Julian Togelius, Mike Preuss, Nicola Beume, Simon Wessing, Johan Hagelbäck, Georgios N Yannakakis, and Corrado Grappiolo. 2013. Controllable procedural map generation via multiobjective evolution. *Genetic Programming and Evolvable Machines* 14, 2 (2013), 245–277.

[45] Julian Togelius, Georgios N Yannakakis, Kenneth O Stanley, and Cameron Browne. 2011. Search-based procedural content generation: A taxonomy and survey. *IEEE Transactions on Computational Intelligence and AI in Games* 3, 3 (2011), 172–186.

[46] Julian Togelius, Georgios N. Yannakakis, Kenneth O. Stanley, and Cameron Browne. 2011. Search-based procedural content generation: A taxonomy and survey. *IEEE Trans. on Computational Intelligence and AI in Games* 3, 3 (2011), 172–186.

[47] Breno MF Viana and Selan R dos Santos. 2019. A survey of procedural dungeon generation. In *2019 18th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*. IEEE, 29–38.

[48] Breno MF Viana, Leonardo T Pereira, and Claudio FM Toledo. 2022. Illuminating the space of enemies through map-elites. In *2022 IEEE Conference on Games (CoG)*. IEEE, 17–24.

[49] Steve Watts. 2020. All The Cyberpunk 2077 Delays. https://www.gamespot.com/gallery/all-the-cyberpunk-2077-delays/2900-3618/. Accessed: 01/02/24.

[50] Claes Wohlin, Per Runeson, Martin Höst, Magnus C Ohlsson, Björn Regnell, and Anders Wesslén. 2012. *Experimentation in software engineering*. Springer Science & Business Media.

[51] Georgios N Yannakakis and Julian Togelius. 2018. *Artificial intelligence and games*. Vol. 2. Springer.

[52] Meng Zhu and Alf Inge Wang. 2019. Model-driven game development: A literature review. *ACM Computing Surveys (CSUR)* 52, 6 (2019), 1–32.