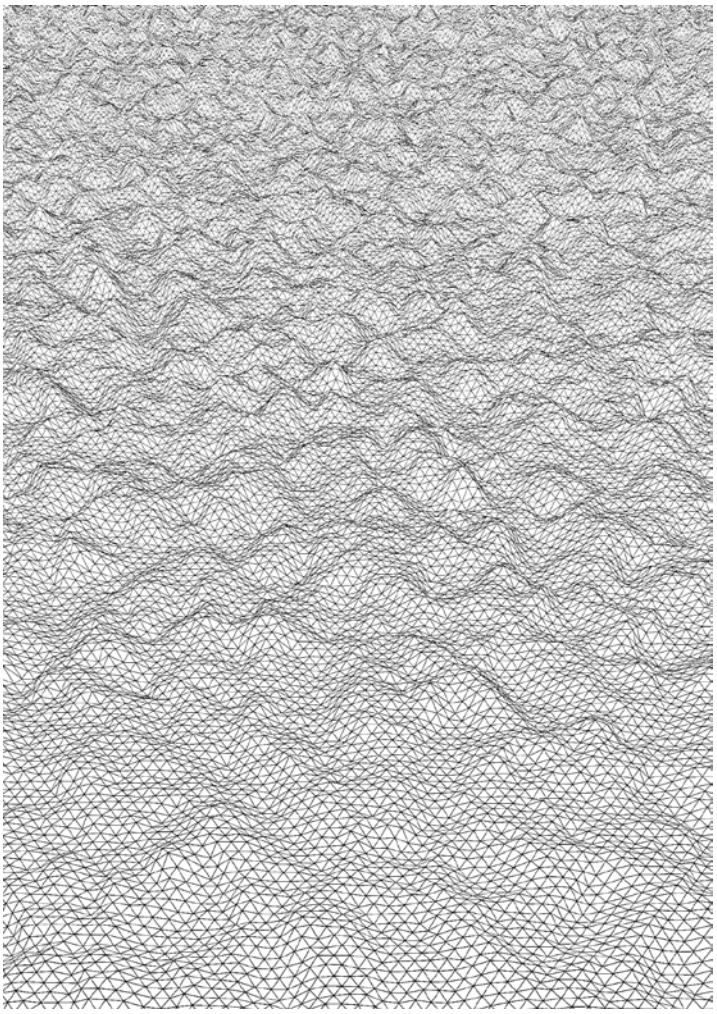
DAC 01

DIGITAL JOURNAL FOR ARTS & CULTURAL STUDIES DIGITALES JOURNAL FÜR KUNST- & KULTURWISSENSCHAFTEN

10 Patterns 0101

Michael Perl & Max Resch



DAC 01

DIGITAL JOURNAL FOR ARTS & CULTURAL STUDIES DIGITALES JOURNAL FÜR KUNST- & KULTURWISSENSCHAFTEN

10 Patterns 0101

AUTHORS

Michael Perl and Max Resch

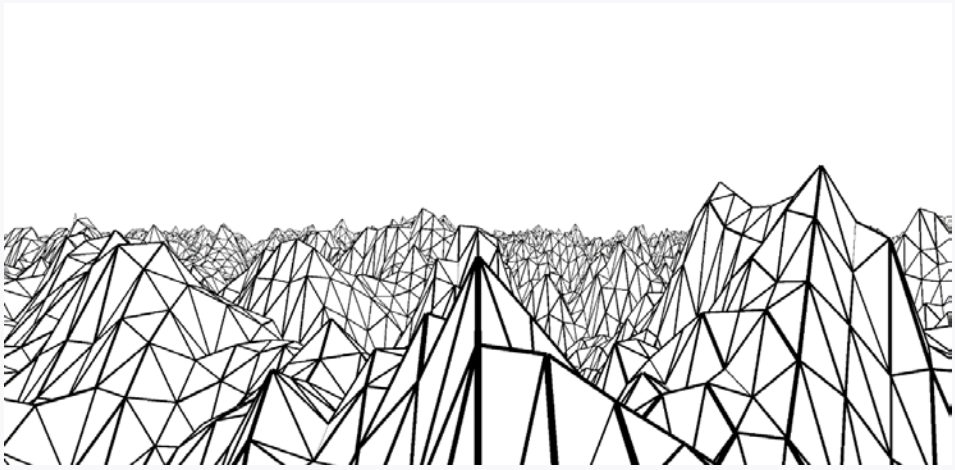


Image 1: Vast Landscapes I, Michael Perl

ABSTRACT

In this paper we explore various facets of procedural content generation (PCG) in computer games, particularly focusing on the interplay between randomness and curation, the use of gradient noise, the concept of seeds, and advanced techniques like Wave Function Collapse (WFC). Further we discuss how PCG has evolved over time, influenced by both technical necessity and cultural innovation, we delve into the concept of order within noise, showcasing how rules and constraints can shape generated content to meet specific needs in game development. By examining these techniques, we offer insights into the potential and challenges of AI-aided game development and the future of PCG in the gaming industry.

KEYWORDS

Procedural Content Generation (PCG), Randomness and Curation, Gradient Noise, Wave Function Collapse (WFC), Computer Games, Techno-cultural Innovation, Game Development Techniques

SUGGESTED CITATION

Perl, M. & Resch, M. (2024). Patterns 0101. DAC – *Digital Journal for Arts & Cultural Studies*, 1. https://doi.org/10.48341/6sfz-ky57

Introduction

1 Patterns and habits in computer games have long captured the attention of researchers, helping us to understand many things like user experiences within the UX-paradigm. This exploration has focused on how people interact with and play software and computer games. However, it is equally intriguing to shift our perspective a little bit away from the user towards patterns that emerge and constitute the games themselves. As we witness the rise of AI-aided development, evident in tools like ChatGPT for writing and DALL-E or Midjourney for visual content generation – or their already well-integrated life-cycle as a part of the Adobe Suite – many questions are raised about the current state and the future to come for games altogether and game development driven by these high-level algorithms.

2 In this context, procedural content generation (PCG) stands out as a cornerstone in computer games (Blatz and Korn, 2017). Reflecting on the evolutionary trajectory of software and game development, we encounter numerous pivotal moments that have shaped the landscape. In the beginning, due to technical necessity, some of the solutions soon developed into a cultural technique on their own, reproduced and refined over many years and beyond a single games life cycle – some of them defining whole genres of games, as with *Rogue*-likes that we'll discuss further on. Tracing back this way starting today with the rise of artificial intelligence, going back to seemingly much simpler times of PCG, might help us to understand the impact these tools have right now, have had on game development and what is yet to come.

Differences in Randomness

3 A significant distinction in PCG can be seen between a random generation and non-random generation or curated generation.

> Procedural content generation is the automatic creation of digital assets for games, simulations or movies based on predefined algorithms and patterns that require a minimal user input. (Freiknecht 2021, p. 107)

4 While random generation works similar to the way one would roll a dice in a game or flip a coin, the curated generation works through setting up a system of rulesets controlling the way content is generated on the get go. Chess or Go are games with fixed rulesets, ruling out random number generation as a factor for deciding the winner of the game. While they are also both games with perfect information (which means that

no information is hidden), another good example to look at is Sudoku, as it offers a finite solution that has to be discovered by the user through deciding the entropy of the missing fields, while not relying on working with a finite set of solutions.

5 While random generation and curated generation might occur in their purest forms, they can also be combined on many different levels and dimensions, adding finesse and opening up possibilities for the game development beyond the two basic forms.

6 One of the most complex examples with a very simple ruleset is Conway's *Game of Life*[1] (Gardner, 1970) premise to the mentioned *Sudoku* would be *Minesweeper*, with the differentiation that the map has no finite set of solutions, but is generated randomly. As with many uses of procedural generation described further down, the power lies within the controllability, as the user can select the difficulty by adjusting the size of the field and the number of bombs in the field.

Generating with Noise

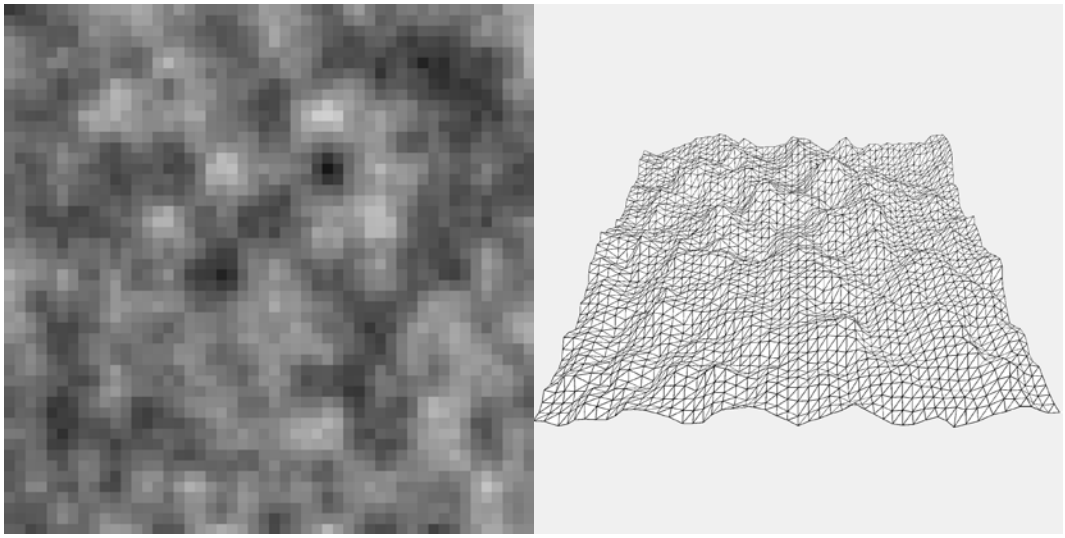


Image 2: The left image shows a black and white noise map. The right image is the application of the noise to a three dimensional terrain. Lighter parts in the map equal a higher altitude, darker equals a lower height.

7 A well developed mixture of selection and randomness are different noise gradient variants. While the first of these can be traced back to Ken Perlin's development of Perlin Noise (Perlin, 1985), there have been many improvements (Perlin, 2002)directions (Gustavson, 2005). While the focus might be on generating structures (Galerne et al.,

1 https://playgameoflife.com (visted 2024-04-23)

2012) or lifelike, organic textures (Worley, 1996; Efros and Leung, 1999), the algorithms might still be applied in a way not intended.

8 Continuing on the examples above, gradient noise is often applied for terrain generation in games. While a pseudorandom number generation might offer a good way to place things "seemingly random" as the mines in Minesweeper, gradient noises excel in terrain generation due to their customizability and smoother and more natural characteristics. The application of multiple gradient noises on top of each other is a common technique in modern game development.

9 Using different seeds and granularity (Scher, 2017) of noise for different entities, almost anything can be created procedurally. In *Factorio*, a sandbox-style construction and production game, the maps are generated procedurally (Earendel and Genhis, 2023). Different variants of gradient noise are used to generate the height maps for land/sea/volcano ratio on different planets as well as the ever so important placement for metal spawns (TOGoS and Twinsen, 2018). The considerations of what has to go where, can be addressed by using different restrictions for different noises, providing scalability but also guaranteeing certain rules of distance they need for the game to be enjoyable. *Terraria*[2] and *Starbound*[3] use these techniques in similar fashions to generate two-dimensional worlds above and beyond the sea level, but also houses, paths and items and even different planets.

10 Procedural generation offers the ability to go infinite, in the sense of offering limitless levels, making PCG a popular choice for providing increasingly difficult levels or maps in many different games like *Rogue*[4] – even spawning an own genre of rogue-like games – or *Stardew Valley*[5], where a mine is seemingly endlessly deep. Similarly, in the Diablo series[6], the levels and the loot are generated procedural, offering a high replayability character, as the linear path of succeeding in the game is broken up by PCG.

Going further with more dimensions

11 While all the examples until now have been in the realm of two-dimensional applications, some gradient noises can even go beyond that. The terrain in *Minecraft* is proce-

2 https://terraria.org/ (Visited: 2024-03-24)
3 https://playstarbound.com/ (Visited: 2024-03-24)
4 https://en.wikipedia.org/wiki/Rogue_(video_game) (Visited: 2024-03-24)
5 https://www.stardewvalley.net/ (Visited: 2024-03-24)
6 https://en.wikipedia.org/wiki/Diablo_(series) (Visited: 2024-03-24)

durally generated[7], entering the realm of three dimensional terrain generation. Additional rules provide context for materials – a layer of sand being added on top of the more solid ground, where flowers etc. are placed. While being more in the spirit of the two-dimensional games above regarding aesthetics, *Dwarf Fortress*[8] goes beyond the scope of generating geography – characters, history and the backstory of the characters going back multiple hundred years of family trees. Reserving two dimensions for time offers the possibility to loop seamlessly while applying disorder and movement through other noise dimensions.

Static Randomness – Seeds

12 Unlike in security applications, where unguessable integers are a fundamental requirement, these generation algorithms work with pseudorandom number generation, i.e. they are predictable in relation to a randomly chosen seed. In many cases this will be truly random and reset on every run, thus allowing making a game with a truly unique experience in every run.

13 Some games opt for a fixed seed like *Elder Scroll II: Daggerfall* – each run of the program will result in the same layout of streets and locations, thus on one hand enabling to create a game that could provide a map containing over 15,000 cities[9] in less than 200 MiB (GOG.com download size in 2024).

14 In the case of *Minecraft* several sites and communities[10] have sprung up collecting and documenting seeds of desirable spawn points. These desirabilities range from just being "beautiful" or being unique to just useful by having above average access to certain resources. The developers of *Minecraft* curated a list of some seeds for selection on the map generator, each providing the player a specific starting experience.[11]

Applying Order to Noise

15 Different types of noises have been shown to be productive in the context of computer game development. In combination with constraints or rules to be applied, the

7 https://minecraft.fandom.com/wiki/Noise_generator (Visited: 2024-03-24)
8 http://www.bay12games.com/dwarves/ (Visited: 2024-03-24)
9 https://en.wikipedia.org/wiki/The_Elder_Scrolls_II:_Daggerfall (Visited: 2024-03-24)
10 https://www.reddit.com/r/minecraftseeds/ (Visited: 2024-03-24)
11 https://minecraft.fandom.com/wiki/Seed_Templates (Visited: 2024-03-24)

aforementioned techniques can become more complex but also distinct and capable to solve specific needs in production with PCG.[12]

16 For this section we consider – for illustrative purposes – a Sudoku game. Here game rules state how each of the nine tiles, each consisting of the numbers 1-9 can be positioned in relation to each other. In the traditional game the tiles are squares and in each row and column of the game field the numbers may only appear once.

17 There are many possibilities in how to create such a selection of numbers, one possibility is to begin by putting random numbers in random positions in a way that the game's parameters are not violated. After some iterations a solver can find a valid solution. In this iterative approach, the burden falls on a solver that unequivocally states the validity of the solution. Sudoku in the general case is NP-complete (Yato and Seta, 2003), but in practice solvers exist (e.g., SAT s[13]) that are fast enough to run iteratively and provide information on the tractability of a specific instance, i.e. answering the questions: can the current selection be solved and is the solution singular.

18 Analogous, this illustrative approach can help understand the problem in tiling, by *fixing* some parts of the generation based on the specific ruleset. In case of map generation, this is done in a first pass, thus creating some information for certain tiles. This is the case in *Factorio*, generating enough terrain of a specific type around the spawn point to later create the necessary resources for game play. (TOGoS and Twinsen, 2018)

19 In the case of our earlier example, *Minecraft*, here special resources spawn on the map after generation of the terrain, thus having no influence on the noise algorithm e.g., *Nether Portals* are generated at fixed positions on the map, the terrain around is altered slightly to match.

Model synthesis and Wave Function Collapse

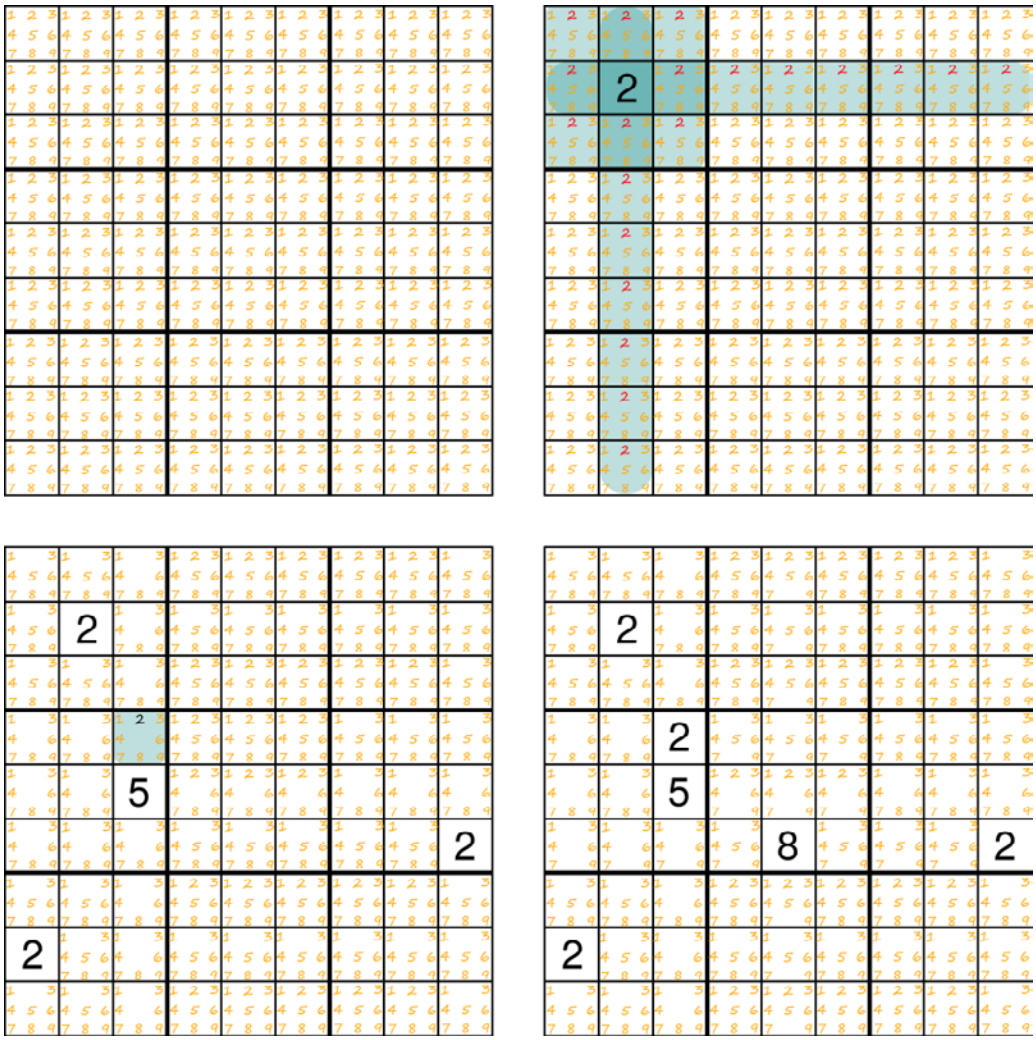
20 If we continue the thought from the previous Section, we arrive at a method called *Wave Function Collapse*[14]. This idea *comes* from the field of Quantum Mechanics, where

12 Interestingly enough, many topics discussed in game development or software development altogether share a lot of similarities to techniques and topics in AI research: https://de.wikipedia.org/wiki/Constraint-Satisfaction-Problem

13 SAT: (Boolean) Satisfiability Problem: Find a valid assignment of boolean variables such that an expression holds valid

14 WFC is closely related to Model Synthesis, for further reading: **https://paulmerrell.org/model-synthesis/** (Visited: 2024-03-24)

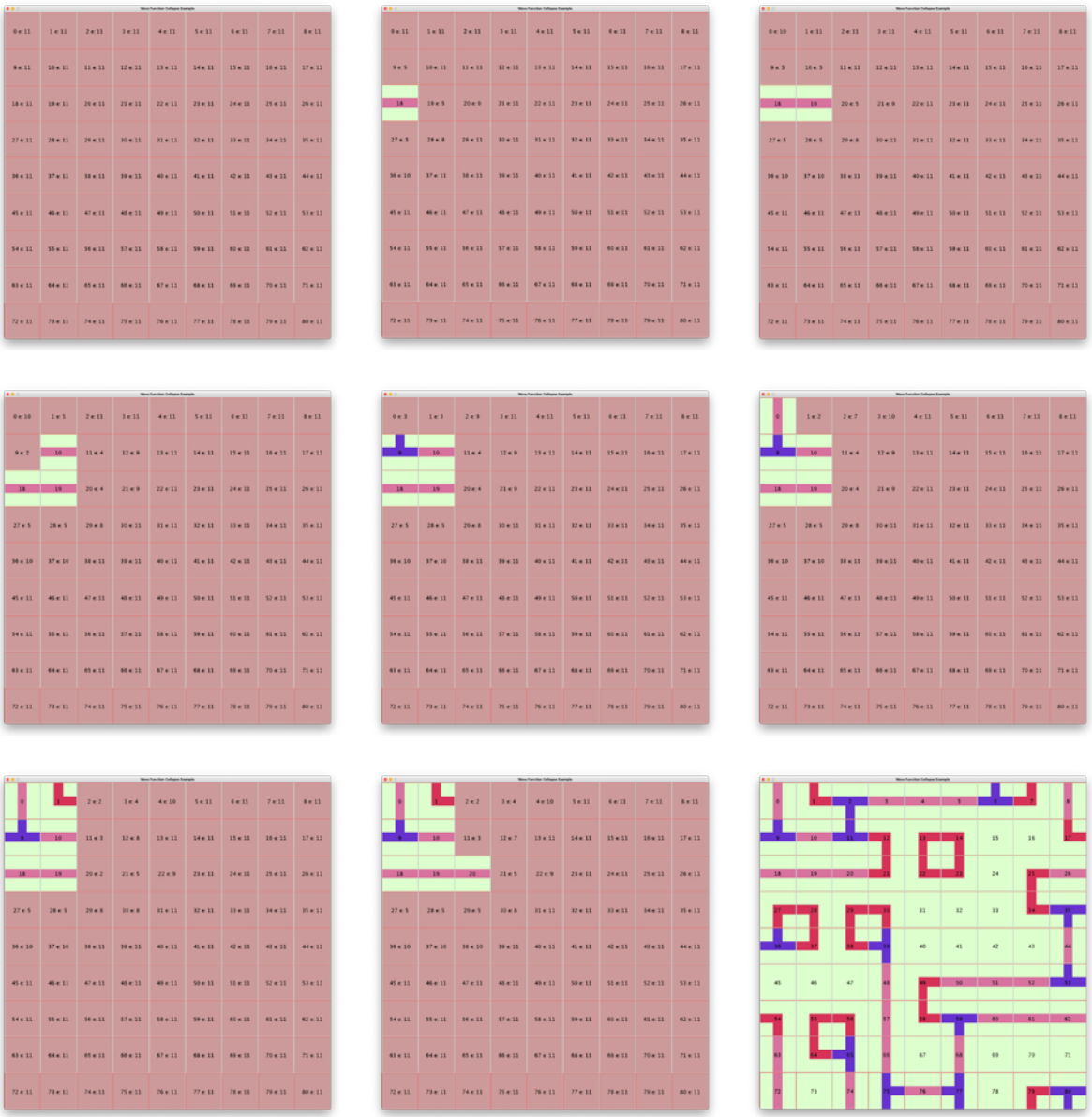
states can exist in a superposition. This superposition collapses to a certain state, when an element of the function gets assigned a value. Now starting at our Sudoku example from before, each field of the game grid is in a superposition and as soon as a field is assigned a value, in related fields (same row, column or subgrid) the number of possible assignments gets less.



21 In the graphic above some steps are illustrated. First all possibilities are open, i.e., each field is in a superposition. In the second image one position gets fixed to a specific value e.g., 2, so along the related axis the function collapses, i.e., the probabilities for values get shifted. The third image shows the case that the collapse functions only allow one value for a specific field, thus forcing the constrained selection, i.e. 2, since neither column nor square allows for another value.

22 This idea can be used by a tiling algorithm to make, e.g. a connected labyrinth. The idea is not bound to be used in only two dimensions, three (or multi-) dimensional struc-

tures can also be created with WFC (BorisTheBrave, 2021), continuing the ideas connected to noises mentioned above about going infinite with PCG (marian42, 2019) by attaching rulesets.



23 Similarly to the Sudoku example above, these pictures show the first few steps of a wave function collapse algorithm picking the tile and candidate (based on entropy), setting them in a grid one after another following the given rules. The last image skips ahead and shows the solved labyrinth.

Conclusion

24 Our exploration of PCG in computer games provided valuable insights into the intersection of creativity, technology, and gameplay. As discussed, techniques such as randomness, curated generation, gradient noise, and advanced algorithms like Wave Function Collapse play pivotal roles in shaping virtual worlds and enhancing player experiences. Moreover, with the integration of AI technologies into game development engines like Unreal Engine and Unity, these techniques are becoming increasingly accessible and powerful.

25 However, the implications of PCG extend beyond entertainment. Computer games serve not only as platforms for immersive experiences but also as training grounds for AI models. (Cerny and Dechterenko, 2015) By leveraging PCG techniques, developers can create vast and diverse datasets for training AI algorithms, enabling them to learn and adapt in dynamic virtual environments.

26 Looking ahead, the continued evolution of PCG and its integration with AI-driven technologies hold immense potential for the gaming industry and beyond. As games become more complex and lifelike, powered by sophisticated AI algorithms, they offer new opportunities for interactive storytelling, immersive simulations, and innovative gameplay experiences. Future developments for AI generated content offer many possibilities, not only in the form of integration as a tool, but as a kind of reverse-engine itself. (Bruce et al., 2024) Moreover, the insights gained from studying PCG in games can inform advancements in AI research and application domains beyond entertainment, such as architecture, urban planning, and education.

27 In essence, the convergence of PCG, AI, and computer games represents a fascinating frontier in technology and creativity. By harnessing the power of these synergistic forces, developers can unlock new realms of possibility, shaping the future of gaming and AI-driven innovation.

References

Blatz, Michael, & Korn, Oliver. (2017). A very short history of dynamic and procedural content generation. In O. Korn & N. Lee (Eds.), Game dynamics: Best practices in procedural and dynamic game content generation (pp. 1–13). Springer International Publishing. https://doi.org/10.1007/978-3-319-53088-8_1

BorisTheBrave. (2021, August 30). Arc consistency explained. Retrieved April 23, 2024, from https://www.boristhebrave.com/2021/08/30/arc-consistency-explained/

Bruce, Jake, Dennis, Michael, Edwards, Ashley, Parker-Holder, Jack, Shi, Yuge, Hughes, Edward, Lai, Matthew, Mavalankar, Aditi, Steigerwald, Richie, Apps, Chris, Aytar, Yusuf, Bechtle, Sarah, Behbahani, Feryal, Chan, Stephanie, Heess, Nicolas, Gonzalez, Lucy, Osindero, Simon, Ozair, Sherjil, Reed, Scott, … Rocktäschel, Tim. (2024). Genie: Generative interactive environments. https://doi.org/10.48550/arXiv.2402.15391

Cerny, Vojtech, & Dechterenko, Filip. (2015). Rogue-like games as a playground for artificial intelligence – evolutionary approach. In K. Chorianopoulos, M. Divitini, J. Baalsrud Hauge, L. Jaccheri, & R. Malaka (Eds.), Entertainment computing - ICEC 2015 (pp. 261–271). Springer International Publishing. https://doi.org/10.1007/978-3-319-24589-8_20

Earendel & Genhis. (2023, December 22). Factorio friday facts #390 - noise expressions 2.0. Factorio. Retrieved April 23, 2024, from https://www.factorio.com/blog/post/fff-390

Efros, A.A., & Leung, T.K. (1999). Texture synthesis by non-parametric sampling. Proceedings of the Seventh IEEE International Conference on Computer Vision, 2,1033–1038 vol.2. https://doi.org/10.1109/ICCV.1999.790383

Freiknecht, Jonas. (2021). Procedural content generation for games [Doctoral dissertation]. https://madoc.bib.uni-mannheim.de/59000

Galerne, Bruno, Lagae, Ares, Lefebvre, Sylvain, & Drettakis, George. (2012). Gabor noise by example. ACM Trans. Graph., 31(4). https://doi.org/10.1145/2185520.2185569

Gardner, Martin. (1970). Mathematical Games - the fantastic combinations of John Conway's new solitaire game "Life". Scientific American, 120–123.

Gustavson, Stefan. (2005). Simplex noise demystified (tech. rep.). Linköping University, Linköping, Sweden. https://cgvr.cs.uni-bremen.de/teaching/cg_literatur/simplexnoise.pdf

marian42. (2019, January 6). Infinite procedurally generated city with the wave function collapse algorithm. Retrieved April 23, 2024, from https://marian42.de/article/wfc/

Perlin, Ken. (1985). An image synthesizer. Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques, 287–296. https://doi.org/10.1145/325334.325247

Perlin, Ken. (2002). Improving noise. Proceedings of the 29th annual conference on Computer graphics and interactive techniques. https://doi.org/10.1145/566570.566636

Scher, Yvan. (2017). Playing with Perlin noise: Generating realistic archipelagos. Medium. Retrieved April 23, 2024, from https://medium.com/@yvanscher/playing-with-perlin-noise-generating-realistic-archipelagos-b59f004d8401

TOGoS & Twinsen. (2018, August 31). Factorio friday facts #258 - new autoplace. Factorio. Retrieved April 23, 2024, from https://www.factorio.com/blog/post/fff-358

Worley, Steven. (1996). A cellular texture basis function. Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, 291–294. https://doi.org/10.1145/237170.237267

Yato, Takayuki, & Seta, Takahiro. (2003). Complexity and completeness of finding another solution and its application to puzzles. IEICE transactions on fundamentals of electronics, communications and computer sciences, 86(5), 1052–1060.

About the authors

Michael Perl is an artist and researcher specializing in experimental visual art and music. Interested in the intersection of technology and creativity, Michael's work explores the effects of technological advancements on game development.
https://michael-perl.com/

Max Resch is a software developer and worked in Operations Research. He works at the University Krems focusing on technical innovations based on existing legacy data infrastructure and the development of new systems for digital collection management.
https://orcid.org/0000-0002-4848-5161