Degree Project in Technology

First cycle, 15 credits

Representing video game style with procedurally generated content

How wave function collapse can be used to represent style in video games

FILIP HEDMAN
MARTIN HÅKANSSON

Stockholm, Sweden, 2023

Abstract

As the video gaming industry continues to grow, developers face increasing pressure to produce innovative content swiftly and cost-effectively. Procedural Content Generation (PCG), the use of algorithms to automate content creation, offers a solution to this problem. This paper explores the PCG algorithm wave function collapse's (WFC) potential for replicating the stylistic design in video games. We provide an exploration of how the WFC algorithm works and discuss the methodology used to evaluate the generator's ability to generate content that mimics a video game style. The study evaluates the algorithm's efficacy by generating levels in the style of the iconic game Super Mario Bros, highlighting its ability to produce original content while maintaining the game's stylistic features. Additionally, we do an examination of the research surrounding PCG and Machine Learning in Super Mario Bros, drawing comparisons with our methodology. The paper concludes with an assessment of WFC's capabilities to replicate style with its generated content with the help of earlier established evaluation metrics.

Keywords

Procedural content generation, Procedural generation, Wave function collapse, Video games, Platform game, Expressive range

Sammanfattning

Med den växande videospelsindustrin så möter utvecklare ett ökande tryck att producera innovativt innehåll snabbt och kostnadseffektivt. Procedural Content Generation (PCG), användningen av algoritmer för att automatisera skapandet av sådant innehåll, erbjuder en lösning på detta problem. Denna artikeln utforskar PCG-algoritmen wave function collapses (WFC) potentiella användning för att replikera design i datorspel. Vi ger en förklaring hur WFC-algoritmen fungerar och diskuterar metodiken som används för att utvärdera generatorns förmåga att generera innehåll som efterliknar ett visst datorspel stil. Studien utvärderar algoritmens effektivitet genom att generera nivåer i samma stil som i det ikoniska spelet Super Mario Bros, vilket betonar algoritmens förmåga att producera originellt innehåll samtidigt som den bevarar spelets stilistiska egenskaper. Dessutom undersöker forskningen kring PCG och maskininlärning i Super Mario Bros, och gör jämförelser med vår egna metodik. Uppsatsen avslutas med en bedömning av WFC:s förmåga att replikera stil med dess genererade innehåll med hjälp av tidigare etablerade utvärderingsmått.

Nyckelord

Generering av procedurinnehåll, Processuell generering, Vågfunktion kollaps, Videospel, Plattformsspel, Uttrycks räckvidd

Acknowledgments

We would like to thank our thesis supervisor, Ylva Fernaeus, our examinator, Haibo Li, and our supervision group.

Stockholm, June 2023
Filip Hedman Martin Håkansson

Contents

1 Introduction

1.1 Background

The increasing advancement of media technology has driven a constant need for innovation in the way to create, distribute, and consume content. Video games, an aspect of media technology, continually demand new methods to generate engaging and dynamic experiences for players. This study delves into the use of procedural content generators as a means to automate and streamline the creation of game content that has the ability to mimic a chosen style. We examine WFC's effectiveness in replicating specific styles in video game-level design, a task traditionally requiring substantial human input and creativity. The significance of this research extends beyond the gaming industry, offering insights into the broader field of media technology where automated content creation and machine learning algorithms are increasingly employed to enhance user experiences and optimize resource allocation. By shedding light on the capabilities and potential limitations of WFC, we aim to contribute to the evolving discussion on how machine learning plays a central role in advancing media technology, a very contemporary and intriguing subject today.

As video games get increasingly bigger and the demand for them grows [1] companies and developers are put under increasing pressure to create new games or extend existing games with new content [2, 3]. One of the methods used to ease the workload and the creation of new content is procedural content generation [4–6]. The use of procedural generation to create video game content has gotten progressively more common as the technology surrounding it evolves [7] and as the demand for larger games and more content has increased. Procedural content generation (PCG) is computer software that can create game content on its own, or together with one or many human players or designers.

The use of such algorithms to automatically and procedurally generate content can both speed up and ease the creation of, for example, level design or asset creation in video games [4–6]. By speeding up these processes, developers can focus resources on other areas and reduce production time and cost, thereby creating a competitive advantage in the industry [7]. The use of PCG as an intelligent design tool can also allow smaller teams or companies and even hobbyists to create content-rich games without having the resources of a big company while still overseeing the direction of the game [7].

In the book “Procedural Content Generation in Games” it is mentioned how the implementations of PCG methods can be seen as a solution to content generation problems [7]. An example of such a problem could be to generate a low-level detail grass texture that does not look strange. While the list of desirable properties for PCG solutions differs for each problem there is a list of commonly desirable properties:

- Speed
- Reliability
- Controllability
- Expressivity and diversity
- Creativity and believability

1.2 Problem and Purpose

The algorithm we have focused on in this study is characterized by having high controllability and reliability values [8].

- Controllability being defined as the algorithm being able to follow a ruleset in a specific dimension when generating content [7]. This ruleset could for example be in the dimension of textures and one of the rules could be that the ground of the generated map can only be made out of a dirt texture.
- and Reliability being defined as the algorithm guaranteeing that the generated content satisfies some given criteria [7]. This could, for example, be that a dungeon with no entrance or exit is never generated because if a dungeon is generated it is a criterion that it always is with an entrance and exit.

The algorithm which we will be analyzing is the wave function collapse algorithm. The wave function collapse algorithm, henceforth referred to as WFC, was created by Maxim Gumin around the fall of 2016 [9]. The fundamental principle of the WFC algorithm is: from a small input create constraints for the algorithm and then generate an output using constraint solving and simple machine learning. The version of WFC we use in this article is its overlapping model. In section 2.2 we will give a more thorough explanation of how WFC works and what modifications we have done to the algorithm.

In a report by members of the academic procedural content generation (PCG) community [10], long-term goals and associated challenges were outlined, accompanied by a set of actionable steps to tackle these challenges. In this study, we have opted to concentrate on the actionable step 4.4 ”Competent Mario Levels”, pertaining mainly to challenge 3.2, “Representing style”. Representing style revolves around developing a content generator capable of emulating the style of another artist, representing style could, for example, be a skillful painter who could study a number of Picasso paintings and produce a new painting that was recognizably in the same style as Picasso’s paintings. Another example could be that a language model, like ChatGPT, which is trained on George Orwell’s books could produce new writing that is distinctly similar to that of George Orwell’s style of writing. Frequently, researchers employ the original levels from the iconic game Super Mario Bros (SMB) as a testing ground for creating such content generators [11, 12].

While there have been a few research paper in the area which looks at the use of PCG to imitate a type of style, the problem with the existing research is that the results can create weird-looking structures or structures which deviate from the style which the original input uses [12]. The advantage of WFC is that its overlapping pattern model preserves local patterns found in the input image when generating the output image while still creating original content [8]. This gives the opportunity to create original content while still preserving structures without deviating from the original level design.

Therefore, the research question this article will explore is: How can wave function collapse be used to generate video game content replicating a type of style?

2 Background

In the history of research surrounding procedural content generation (PCG), much of the research has focused on constructive and search-based techniques [8]. Only newer research seems to focus on different types of machine learning and constrain solving algorithms. The Wave Function Collapse (WFC) algorithm, conceived in 2016 by developer Maxim Gumin, is a constraint-solving algorithm [9]. It can be considered a rudimentary form of machine learning, although it distinguishes itself from typical ML algorithms due to its ability to produce satisfactory results with minimal input data [9]. The WFC algorithm has found applications in the gaming industry, predominantly in niche genres for which its output is best suited [13, 14]. While the practical implementation of WFC is well-documented, there is a scarcity of research on its performance within the context of machine learning algorithms.

2.1 Generating levels with wave function collapse

2.1.1 Definitions

Some of the names of the following concepts differ a bit between sources, we have chosen the following definitions for the concepts:

PCG - Procedural content generation, the algorithmic creation of game content with limited or indirect user input [15]

WFC - Wave function collapse, the PCG algorithm we make use of in this research paper

LL - Linearity and Leniency, the metric of our evaluation method

Overlapping pattern model - A version of wave function collapse where the solver operates on N x N patterns of tiles rather than on individual tiles [9].

Field - The area on which WFC operates. With roughly the size of the output image, an element in the field is a tile and contains all possible modules that could potentially populate the part of the map.

Tile - A structure containing all possible modules at their current state, identified with an x and y value. When the field is not collapsed, each tile will contain all the possible modules. When the full field has collapsed and the algorithm has finished, each tile will contain only one module.

Module - N x N structure of input blocks. In a naive no-overlap solution (N = 1) a module is equal to an input block

2.1.2 Generation

Maxim Gumin created the wave function collapse algorithm with the goal of creating a game map generator that produces outputs with strong local similarity [8]. Where every N x N area could be exactly traced back to a part of the input picture. [9].

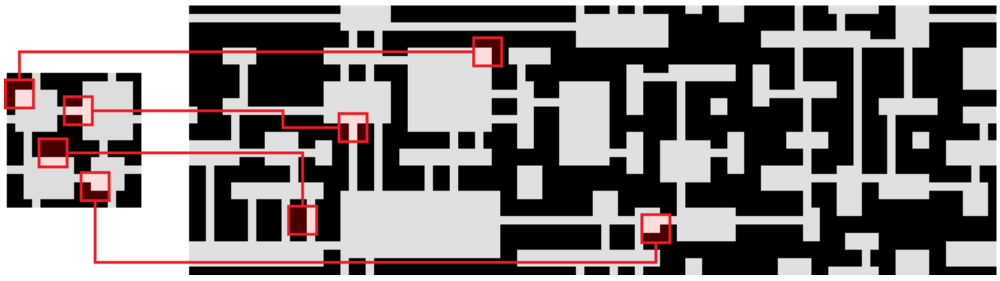

Figure. 2.1: *Input image on the left, output on the right*

A thorough rundown of the algorithm can be found in the article ”WaveFunctionCollapse: Content Generation via Constraint Solving and Machine Learning” by Isaac Karth and Adam M. Smith, or in the original GitHub repository. Here we present a simpler technical explanation of how the WFC algorithm works. It is important to note that this explanation is based on how our own version of the algorithm is written and that this is not the original WFC written by Maxim Gumin, although all the functionality stays the same, just fitted for our purpose. It looks like this: [9].

1. Defining building blocks from the input

   The algorithm starts off by, from the input image, iterating through and saving the unique x*y building blocks of the image.

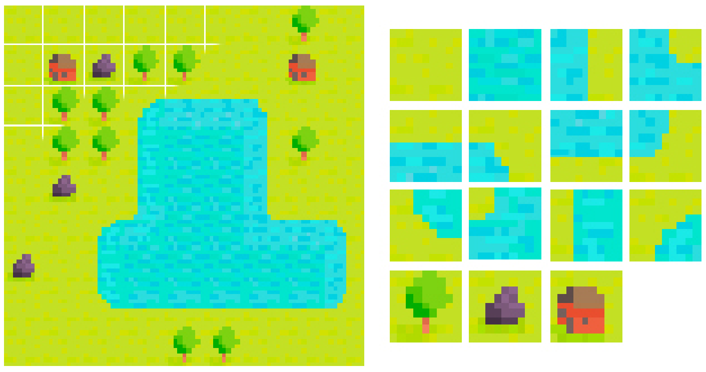

Figure. 2.2: *Input image on the left, unique building blocks on the right*

2. Creating Modules

   With the building blocks created the algorithm then defines patterns including them. We call the size of the patterns N. In this thesis, we work with N = 3. A pattern will be all the unique 3x3 patterns of building blocks. This pattern is what we then call a module. Figure 3 shows some examples of 3x3 modules. For a 9x9 building block input image, the total amount of complete patterns will be 7x7=49.

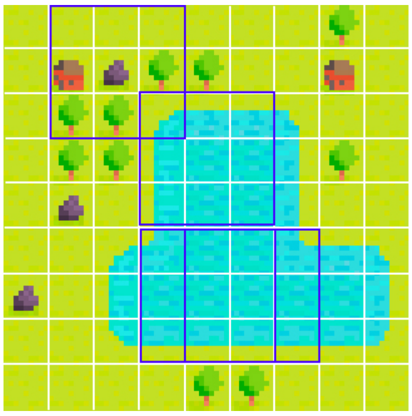

Figure. 2.3: *Example of how the 3x3 patterns might look*

3. Defining Rules

   Rules are defined by the module's adjacency in the input picture. If there is a horizontal or vertical adjacency between two modules in the input picture, this will be marked as an accepted adjacency rule. This ensures the local N x N similarity. It's important to note that this adjacency is between Modules, and not building blocks. It's not as easy to picture what the adjacency between two modules looks like because a module is defined by a pattern. To make it easier we can imagine that the middle piece of the pattern/module being adjacent to the middle piece of another pattern/module is what defines the adjacency. The term for this that often comes up is overlap as opposed to the "naive" solution where the pattern size is 1x1.

4. Building Output

   The output is built according to a simple, lowest entropy rule. A probability for each module is set corresponding to their frequency of appearance in the input image. An entropy calculation is done within each tile using all the possible modules that it contains. The tile with the current lowest entropy is chosen, and within that tile, a single module is randomly chosen in accordance with its probabilities, called a collapse.

This is followed by propagation, that is updating the field according to the adjacency rules. This main loop continues until the entire field has collapsed. There is no guarantee that it will reach a result, if no collapse is possible, the algorithm can either back-propagate or entirely restart.

WFC was created with the desire of having a model that preserves local patterns [8], thus achieving high reliability, a desirable trait in PCG [16]. Meaning a map generated using WFC contains no big surprises or patterns that differ from the patterns seen in the input image [9]. What this means is that it is easy to achieve results that look good at first sight. But how good is WFC on closer inspection, as a Machine learning algorithm? The algorithm is fairly simplistic, so could it compare to a seemingly more sophisticated machine learning algorithm to create levels that represent style? Or is it only easy on the eyes?

2.2 PCG and Super Mario Bros

Steve Dahlskog and Julian Togelius are examples of researchers who have conducted used SMB in their research. They've employed a search-based multi-level approach for content generation by identifying different scales of patterns and generating at these different scales simultaneously [11], the solution is a more classic approach to representing style problems. More like how a human would go about solving a problem, seeing patterns at different levels and trying to imitate these. In the report Linear levels through n-grams, also by Steve Dahlskog as well as Julian Togelius, Mark J. Nelson, The linguistic model known as n-grams is used to generate the linear levels of SMB, a lot like WFC, n-grams have a more local awareness when generating content.

Both of the articles, along with many others managing PCG, use something called "expressive range" as a measurement tool, together with the metrics linearity and leniency [12, 16]. An evaluation method that we employ for our methodology.

3 Methodology

3.1 Expressive range

The expressive range of a generator is the style and variety of levels that can be generated with the generator. The variety and style are measured with appropriate metrics for the content that the generator produces giving different generators different metrics depending on the content it produces [17]. It can also be used to see biases or holes in the expressive range of the generator, facilitating the detecting and fixing of biases and holes. The primary goal of evaluating a content generator is to assess its ability to achieve the intended outcomes. In this study, we aim to determine whether the modified WFC algorithm can effectively represent style in its generated content. To evaluate this, we visualize the content space covered by the generator using a top-down approach, adopting content generation statistics, particularly expressivity measures, as the axis for this visualization [16]. This method allows us to assess the expressive range of our content generator, and employ it to compare the expressive range of the original map with that of the generated maps [16].

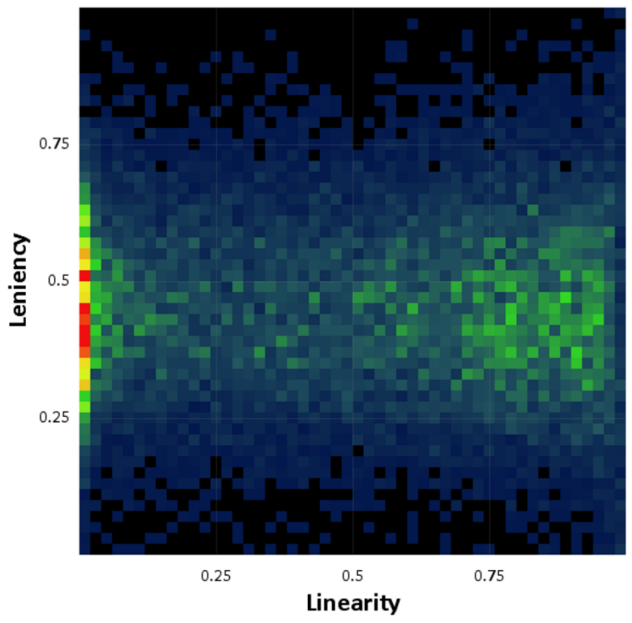

Figure. 3.1: *Example graph showing a generators expressive range* [18]

3.2 Evaluation

For our evaluation, we utilize Smith and Whitehead's Linearity and Leniency metrics [16], which are commonly used in the field when assessing PCG algorithms in the context of 2-dimensional platforming games [11, 12, 16]. A low linearity value indicates that players are required to jump frequently, while a high linearity value suggests a flatter map. Conversely, high leniency values imply that the map contains more enemies and gaps where players may lose a life. We generate a larger number of maps (n=1000) to gain a more comprehensive understanding of the content generator's capabilities [16].

3.3 Linearity and Leniency

Our implementation of linearity and leniency values follows a simple model. For each element that potentially can cause the player to lose a life, for example, gaps and enemies, are given a negative leniency value, and for tiles like ground and jumps with no danger, a positive score. Likewise, a negative linearity value is given to all the tiles whose highest platform deviates from the ground level. The size of the values depends on which tiles they are, for example, an enemy tile has a leniency of -2 while a gap has -1.

3.4 Generating maps with WFC

SMB levels 1-1, 1-2, 2-1, and 3-3 are used as input to carry out the experiment. We generated 1000 maps from each input map using WFC. Generating SMB levels with WFC can be done in different ways. The maps have block structures consisting of some amount of 16x16 size blocks, for our experiment, we needed to choose an appropriate size for the WFC building blocks and also an appropriate N, the size of the patterns. We set N = 3 because it's a middle ground between high-solving difficulty and low output complexity. The size of the building blocks can be chosen in different ways. Because of SMB's linear map structure, it seems intuitive to use vertical blocks that span across the map height.

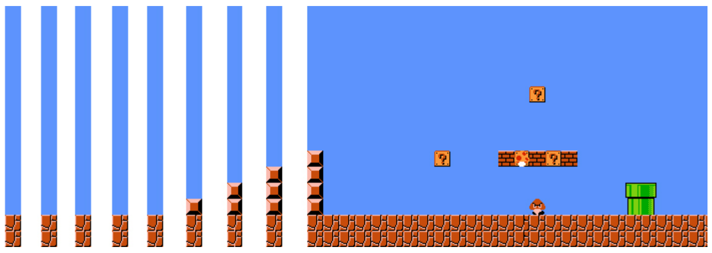
Figure. 3.2 *Visualization of building block size used in our WFC*

note that such a partition only requires WFC to run in one direction. Instead of a 2-dimensional overlay, it will only be a 1-dimensional overlay.

4 Result

After confirming the validation of our algorithm we tested it on different input maps from SMB to investigate how they affected the expressive range of our generator and its' efficiency. We then compared the original map's expressive range with the mean values of the generated maps and looked at their Pearson correlation.

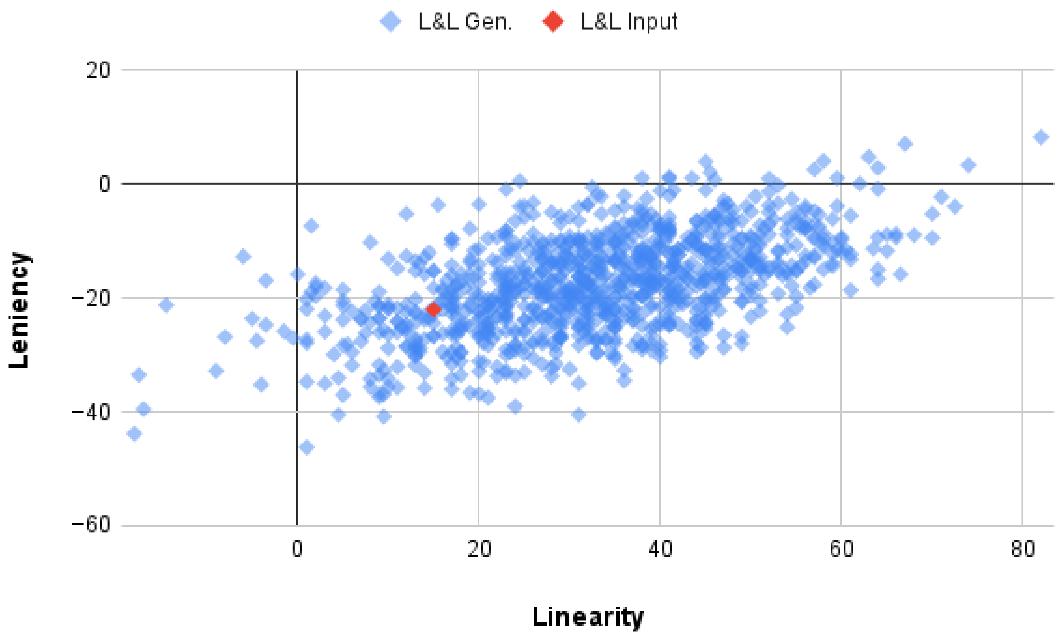

Figure 4.1: *Scatter plot for SMB level 1-1 detailing the expressive range of input and generated maps for n=1000*

The maps plotted above give us a vague idea of the linearity and leniency of the generated maps, looking at these together with our generated maps (4.2, 4.3) we get an idea of the expressive range of the system. The generated maps we see here are in figure 4.2 the maps furthest away from the input and therefore considered the worst according to our style of measurement, and in figure 4.3 are the ones closest to the input and therefore considered the best. As we can see the spread is not huge but still noticeable.

Figure. 4.2:

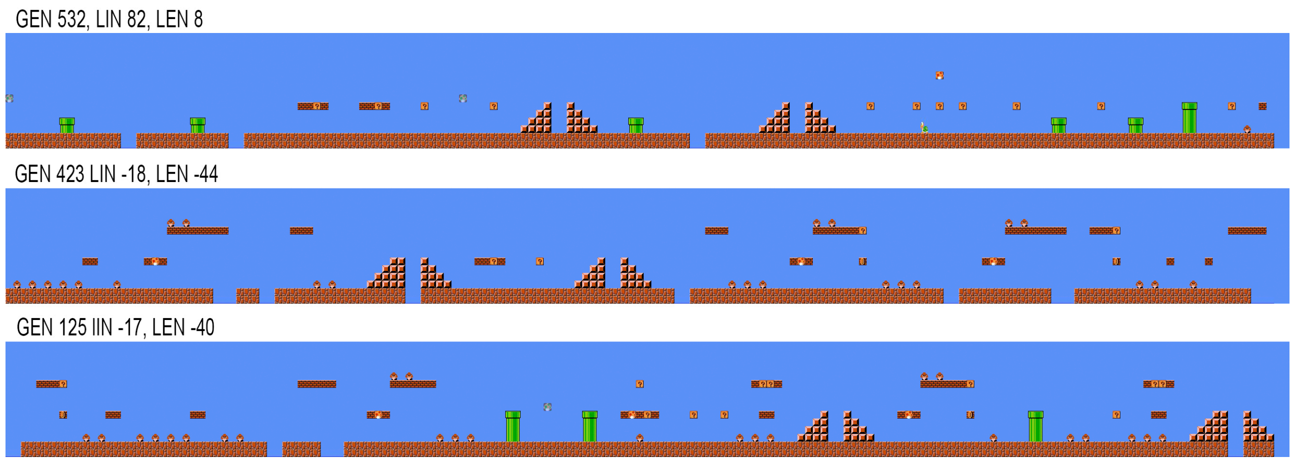

Figure 4.2: *Results that are the furthest away from the inputs L&L values*

Figure. 4.3:

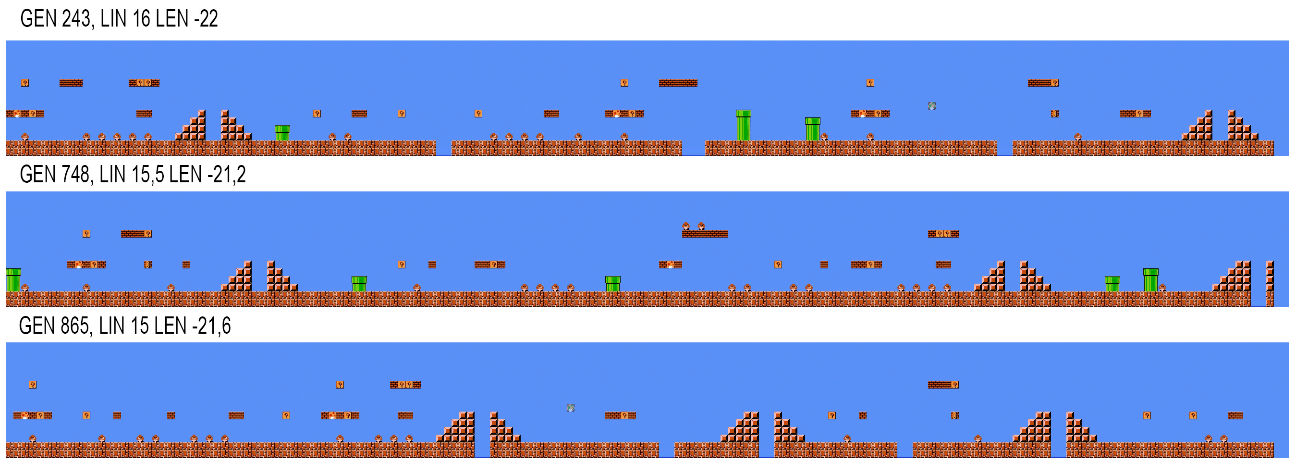

Figure 4.3: *Results that are the closest to the inputs L&L values*

It's important to point out the probabilistic nature of the WFC algorithm and that it can contribute to unreliable results using this type of measure. While it is quite common to make use of probabilistic algorithms in this field [6, 12] it is a topic that we discuss in more detail in 5.1.1.

4.1 Effect of varying input data

While there is a plethora of maps in SMB to choose from many of them are variations of each other that use the same assets. Therefore we opted to choose maps that have distinct differences in both structures and visual styles to see how the algorithm holds up to different input types and if there is any difference in performance depending on the input used.

4.2 Expressive range

As mentioned in section 3.1 we utilize the concept of expressive range to conduct the style analysis of our generated maps [16, 17]. We then plot the two extracted metrics onto a 2-dimensional space and compare the generated results with the input map values of the same metrics. We do this comparison by calculating the Pearson correlation between the generated maps mean linearity with that of the input maps' linearity and then the same for the leniency for every tested map. This gives us an idea of how good our generator is at mimicking style. The mean linearity and leniency values that we got from the different inputs can be seen in table 4.4

Input map	Lin. (SMB)	Lin. (gen. rounded)	Len. (SMB)	Len. (gen. rounded)
1-1	15	32,7	−22	−17,5
1-2	−17	−15,2	−17,4	−17,9
2-1	23	31,0	-45.8	−47,1
3-3	−13	8,4	−47,2	−37,6

Figure 4.4: *Mean expressive range values for n=1000*

The Pearson correlation between the linearity of the input and the generated level was 0,9179 rounded; for leniency, the values were 0,9461 rounded. These are pretty good values but the lack of different input maps could have a negative effect on the results, a problem which we discuss in more detail in 5.1.2.

5 Discussion

5.1 Study Limitations

5.1.1 Biased outputs

Because the wave function collapse algorithm in its nature chooses modules depending on the frequency of appearance in the input it is likely that with a large enough output, the distribution converges to the input distribution. So it's fair to argue that the chosen metrics of linearity and leniency might be biased toward the input parameters. As written by N. Shaker et al. in the book Procedural Content Generation in Games, "The important rule of thumb to remember when choosing metrics for your content generator is the following: strive to choose metrics that are as far as possible from the input parameters to the system. The goal of performing an expressive range evaluation is to understand the emergent properties of the generative system" [16]. The WFC algorithm doesn't strive to produce the same linearity and leniency values as the input per se, it is only likely producing the same distribution in modules. And just because it strives to produce the same distribution it doesn't mean the same distribution will show up in every output, this is something that would need further research.

This is a presumed complication with the wave function collapse which we are aware of, although most scientific articles which explore the use of algorithms in PCG make use of probabilistic algorithms of some sort or some type of constraint solving to generate content.

5.1.2 Pearson correlation and small sample size

Using Pearson correlation to analyze the generated maps gives us an idea of the relationship between the input maps and the corresponding output maps. It's worth noting that only the mean of all the generated maps is used for the correlation. It's not the perfect measurement to use in our case, however, it still gives us a hint of how similar the generated maps are to the input maps. As of right now, our assessment is that manual inspection together with the expressive range analysis and the Pearson correlation is sufficient to get a good general idea of the performance of the generator.

To get a proper representation out of using Pearson correlation a higher sample size would have been needed. In the scope of this project, the amount of four maps was deemed reasonable taking into consideration both the manual and generation time of the maps. However, in a more accurate survey, all of the SMB maps would be used. As of right now, the Pearson correlations measurement has a P-value of 0.22 which is not great.

5.2 WFC limitations

5.2.1 Lack of original assets

Because of how the wave function collapses overlapping model works there are never any original assets created from the algorithm. While it can create original levels it will always ”use” or ”borrow” assets from its input.

5.2.2 Repeating patterns in output

With some inputs, the algorithm tends to repeat patterns more commonly than on other maps. Which might end up looking ugly in the output and boring for the player if one was to play the level.

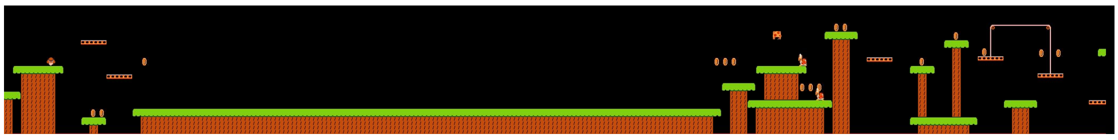

Figure. 5.1: *Example of a severe case of a repeating pattern*

However, heavily flawed maps like the one in figure 5.1 can be found by analyzing the characteristics of the generator in the generative space using the expressive range. In figure 5.2 we can see that there are quite a few maps that are far away from the main cluster of maps (top right).

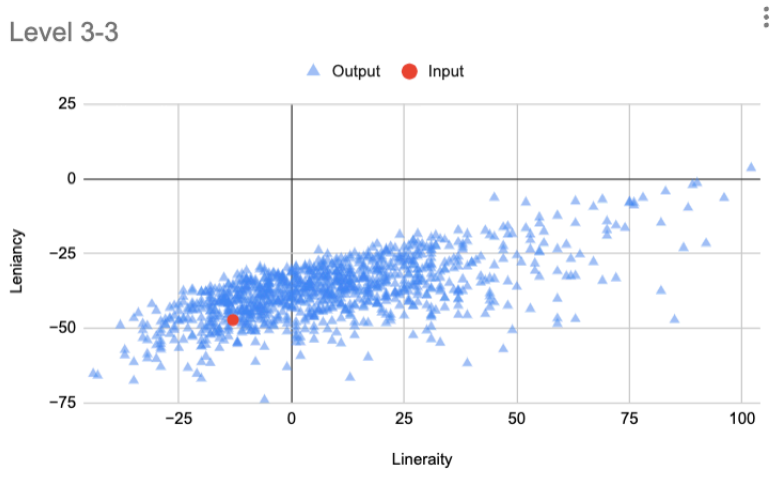

Figure. 5.2: *Expressive range of map 3-3*

These have both high linearity and leniency and by checking these maps manually we that all of them possess the same issue. By for example only choosing maps, by some distribution, that are within the main cluster we can eliminate such generations.

5.2.3 Hard coded placements of assets

Another limitation with the algorithm also in this niche case is that some parts will have to be hard coded into the output, in this case, that would be the castle and the flag at the end. These parts will also have to be removed from the input or somehow be excluded in the generation to prevent a castle from showing up in the middle of the map.

5.3 Future Work

Some interesting future work that could be built upon this research could be trying out different modifications to the wave function collapse algorithm to see how these changes would further alter the output of the algorithm, for example, one could implement the concept of anxiety curves to the generation process [19] to add even more depth to the generation.

Another idea would be to see how the output would change if one changed the probability model when propagating tiles.

Future research could also consist of combining WFC with other more sophisticated models, and thus getting the positive sides, like the high reliability from WFC together with more sophisticated, for example, machine learning algorithms like n-grams.

5.4 Conclusion

The Pearson Correlation we got from the generated maps' expressive range and input map has a correlation over 90% as seen in sec.4.2, which shows that, according to the choice of metric, that WFC can be used to represent the style of SMB level. While the Pearson correlation might be weaker compared to other studies [12] and the statistical uncertainty high, the high reliability of WFC makes it interesting for future research.

Bibliography

[1] P. Zackariasson and T. Wilson, “Paradigm shifts in the video game industry,” *Competitiveness Review: An International Business Journal incorporating Journal of Global Competitiveness*, vol. 20, pp. 139–151, Mar. 2010. DOI: 10.1108/10595421011029857.

[2] C. Teipen, “Work and employment in creative industries: The video games industry in germany, sweden and poland,” *Economic and Industrial Democracy - ECON IND DEMOCRACY*, vol. 29, pp. 309–335, Aug. 2008. DOI: 10.1177/0143831X08092459.

[3] M. Borg, V. Garousi, A. Mahmoud, T. Olsson, and O. Stålberg, “Video game development in a rush: A survey of the global game jam participants,” *IEEE Transactions on Games*, vol. 12, no. 3, pp. 246–259, 2020. DOI: 10.1109/TG.2019.2910248.

[4] Y.-G. Cheong, M. O. Riedl, B.-C. Bae, and M. J. Nelson, “Planning with applications to quests and story,” in *Procedural Content Generation in Games*. Cham: Springer International Publishing, 2016, pp. 123–141, ISBN: 978-3-319-42716-4. DOI: 10.1007/978-3-319-42716-4_7. [Online]. Available: https://doi.org/10.1007/978-3-319-42716-4_7.

[5] M. J. Nelson and A. M. Smith, “Asp with applications to mazes and levels,” in *Procedural Content Generation in Games*. Cham: Springer International Publishing, 2016, pp. 143–157, ISBN: 978-3-319-42716-4. DOI: 10.1007/978-3-319-42716-4_8. [Online]. Available: https://doi.org/10.1007/978-3-319-42716-4_8.

[6] N. Shaker, A. Liapis, J. Togelius, R. Lopes, and R. Bidarra, “Constructive generation methods for dungeons and levels,” in *Procedural Content Generation in Games*. Cham: Springer International Publishing, 2016, pp. 31–55, ISBN: 978-3-319-42716-4. DOI: 10.1007/978-3-319-42716-4_3. [Online]. Available: https://doi.org/10.1007/978-3-319-42716-4_3.

[7] J. Togelius, N. Shaker, and M. J. Nelson, "Introduction," in *Procedural Content Generation in Games*. Cham: Springer International Publishing, 2016, pp. 1–15, ISBN: 978-3-319-42716-4. DOI: 10.1007/978-3-319-42716-4_1. [Online]. Available: https://doi.org/10.1007/978-3-319-42716-4_1.

[8] I. Karth and A. M. Smith, "Wavefunctioncollapse: Content generation via constraint solving and machine learning," *IEEE Transactions on Games*, vol. 14, no. 3, pp. 364–376, 2022. DOI: 10.1109/TG.2021.3076368.

[9] M. Gumin, *Wave Function Collapse Algorithm*, version 1.0, Sep. 2016. [Online]. Available: https://github.com/mxgmn/WaveFunctionCollapse.

[10] J. Togelius *et al.*, "Procedural content generation: Goals, challenges and actionable steps," *Dagstuhl Follow-Ups*, vol. 6, pp. 61–75, Jan. 2013.

[11] S. Dahlskog and J. Togelius, "A multi-level level generator," in *2014 IEEE Conference on Computational Intelligence and Games*, 2014, pp. 1–8. DOI: 10.1109/CIG.2014.6932909.

[12] S. Dahlskog, J. Togelius, and M. Nelson, "Linear levels through n-grams," Nov. 2014. DOI: 10.1145/2676467.2676506.

[13] O. Stålberg, *Oskar stalbergs townscaper*. [Online]. Available: https://oskarstalberg.com/Townscaper/.

[14] [Online]. Available: https://www.badnorth.com/.

[15] D. Schön, "Designing as reflective conversation with the materials of a design situation," *Knowledge-Based Systems*, vol. 5, no. 1, pp. 3–14, 1992, Artificial Intelligence in Design Conference 1991 Special Issue, ISSN: 0950-7051. DOI: https://doi.org/10.1016/0950-7051(92)90020-G. [Online]. Available: https://www.sciencedirect.com/science/article/pii/095070519290020G.

[16] N. Shaker, G. Smith, and G. N. Yannakakis, "Evaluating content generators," in *Procedural Content Generation in Games*. Cham: Springer International Publishing, 2016, pp. 215–224, ISBN: 978-3-319-42716-4. DOI: 10.1007/978-3-319-42716-4_12. [Online]. Available: https://doi.org/10.1007/978-3-319-42716-4_12.

[17] G. Smith and J. Whitehead, "Analyzing the expressive range of a level generator," in *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*, ser. PCGames '10, Monterey, California: Association for Computing Machinery, 2010, ISBN: 9781450300230. DOI: 10.1145/1814256.1814260. [Online]. Available: https://doi.org/10.1145/1814256.1814260.

[18] S. O. Kimbrough, G. J. Koehler, M. Lu, and D. H. Wood, "On a feasible–infeasible two-population (fi-2pop) genetic algorithm for constrained optimization: Distance tracing and no free lunch," *European Journal of Operational Research*, vol. 190, no. 2, pp. 310–327, 2008, ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2007.06.028. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221707005668.

[19] N. Sorenson and P. Pasquier, "The evolution of fun: Automatic level design through challenge modeling," Jan. 2010.

TRITA-EECS-EX-2023:215