Applying the Graph-Based Wave Function Collapse Algorithm to Procedurally Generate 3D-Environments

Isak Reinholdsson
200002176493
isakre@kth.se
KTH Royal Institute of Technology
Stockholm, Sweden

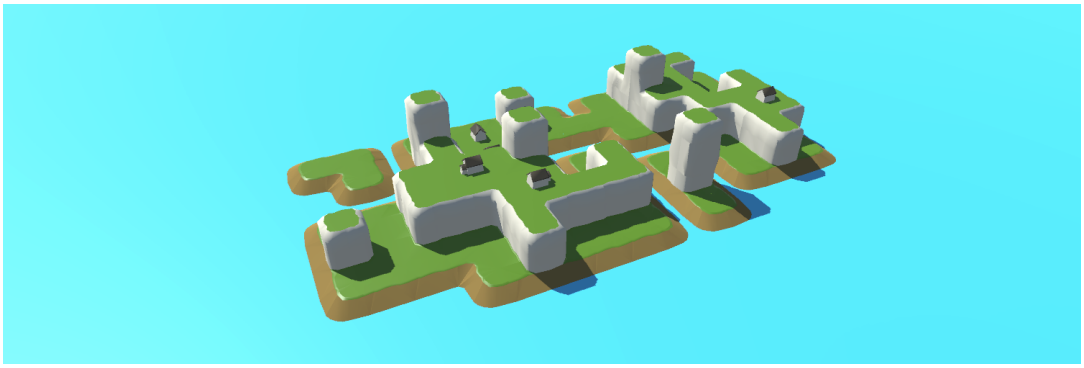

Figure 1: The a preview of the final result of the report. The island on the image was procedurally generated using the graph-based wave function collapse algorithm on a 3-dimensional grid of width 17, depth 10 and height 4.

ABSTRACT

The Wave Function Collapse (WFC) algorithm is a greedy search method used for procedural content generation (PCG). Drawing inspiration from quantum mechanics, the algorithm utilizes a set of constraints to reduce a possibility space into randomized yet coherent patterns, images, or other types of content. This report discusses two different versions of the algorithm: the traditional grid-based version and a more recent graph-based approach. Both of these versions of the WFC are also applied on two different two-dimensional problems to highlight their capabilities. Additionally, it is shown how the graph-based approach can be generalized and applied on generating three-dimensional environments. The results indicate that the graph-based WFC is a useful and effective extension on the original idea, enhancing the original algorithm and facilitating the creation of more complex content without a large penalty in terms of implementation difficulty or execution time. Furthermore, it is found that three-dimensional problems are well suited for the graph-based WFC.

1 INTRODUCTION

Procedural content generation (PCG) is a pivotal aspect of computer graphics. As the complexity and size of simulations, games and other graphical applications increases, there is also a growing need for automatically creating content through computational means. There are several techniques and algorithms employed to achieve this, including noise functions, rule-based systems, and machine learning [5]. In this report, a recent addition to this list is examined: the Wave Function Collapse (WFC) algorithm, a greedy search algorithm that has gained considerable attention for its potential to generate high-quality, versatile content from minimal input [8].

The WFC algorithm was initially introduced by Gumin [4] in 2016, who was inspired by concepts from quantum mechanics to use the concept of superposition and then systematically reduce a set of possibilities to a coherent output. On a surface level, the algorithm works by taking a set of building blocks (in most instances small images) and a set of rules regarding how these building blocks are allowed to be put next to each other. Using this information it then produces a random constellation of the blocks that breaks none of the placement constraints.

In this report, we explore two different versions of the WFC algorithm. We begin with the original grid-based WFC algorithm introduced by Gumin [4] and then discuss its subsequent enhancement through the graph-based WFC developed by Kim et al. [6]. These algorithms are implemented and tested on two-dimensional problems. Finally, we extend the graph-based WFC to handle three-dimensional environments, demonstrating both the capabilities of the graph-based algorithm and how to make it more useful.

2 RELATED WORK

2.1 Current State of WFC

In their article, Karth and Smith [5] highlight how the WFC algorithm is increasingly used in various applications across various domains within procedural content generation. The authors cite

several examples that have creatively utilized this technology for tasks like automatic texture and level generation.

Furthermore, Karth and Smith [5] note that the algorithm is still in its early stages, with extensive research being conducted to optimize it. For instance, recent studies have focused on integrating more sophisticated constraint handling methods, such as a concept they introduce called Answer Set Programming (ASP) where the goal is to optimize pattern selection and placement.

2.2 Graph-Based WFC

Recently, Kim et al. [6] introduced a Graph-Based WFC, extending the original grid-based constraints to graph structures. This adaptation significantly enhances the capabilities of the model as there is no limitations for how tiles can influence each other [6]. As an example, Kim et al. [6] illustrate how it can be used for both solving Sudoku-puzzels and for coloring Voronoi non-grids.

Implementing Graph-Based WFC requires modifying standard WFC processes to manage the dynamic and complex connectivity of graphs. This includes adjustments in propagation, compatibility, and backtracking mechanisms to efficiently handle the variable neighbor configurations of nodes [6].

Kim et al. [6] mention that looking forward, expanding Graph-Based WFC to include machine learning could enhance its efficiency and output quality. Additionally, supporting multi-layered graphs could allow for more sophisticated results [6]. This ongoing development underscores the potential of Graph-Based WFC as a pivotal tool in future PCG applications [6].

2.3 Advancements in Nested WFC for Infinite and Deterministic Content Generation

In 2023, the Nie et al. [7] introduction Nested Wave Function Collapse (N-WFC), a version of the algorithm that marks a significant advancement in the abilites of the model, particularly in handling large-scale and infinite content generation. The method addresses the core limitations of the traditional WFC algorithm, like exponential time complexity and frequent backtracking, by splitting the generation process into manageable sub-grids that are processed individually and then integrated, ensuring consistency and continuity across the generated content [7].

Experimental results underline the superiority of N-WFC over traditional methods, particularly in terms of reducing the time complexity but also in other areas. The practical implementation of this method can be seen in its application to the game Carcassonne, where it was able to generate large-scale, complex maps in real time [7].

2.4 What This Paper Contributes

Despite extensive research on WFC applications and their extension to graph-based models, there remains a gap in applying graph-based WFC to three-dimensional problems. This report aims to explore this uncharted territory, demonstrating the potential of graph-based WFC as an idea and how to expand its usefulness further.

The idea of applying WFC to 3D problems is not new. For instance, both Gumin [4] and Nie et al. [7] mention it as potential use cases. However, none of these sources (not even Kim et al. [6], the founders of graph-based WFC) have discussed the approach of using specifically the graph-based WFC for this purpose.

Both Kim et al. [6] and Nie et al. [7] introduce various optimizations to the WFC. However, for this report, improvements mostly from the former will be used. Given the size of the problems addressed in this report, this will suffice to provide good results. Nevertheless, a potential future study could improve on the report of this model by utilizing further optimizations from sources like Nie et al. [7].

3 IMPLEMENTATION AND RESULTS

3.1 Grid-Based WFC

Before writing the graph-based WFC, the original grid-based WFC algorithm was implemented to use as a starting point for later improvements. For the purposes of this part of the work, the simple image in figure 2 was designed and used as input for the algorithm.

Figure 2: Source image for gridbased WFC.

3.1.1 Tiles. The input to this version of the simple algorithm is a simple image. This image is then used to generate tiles and constraints. This image has a width of 30 pixels and a height of 12 pixels. It was then split into 30 different 3×3 tiles which were used as the tiles for the algorithm. The algorithm then used the tiles such that only the ones next to each other were allowed to be next to each other.

Figure 3: Unique tiles derived from figure 2.

3.1.2 Constraints. After generating all 5 unique tiles seen in figure 3. The image was studied to see which tiles were allowed next to each other. Simply put, the constraints were defined such that the tiles next to each other in the source image are the only tiles allowed to be put next to each other in the newly produced image.

3.1.3 Algorithm. Using the five generated tiles and the constraints from the image, the next part was placing these small subimages in a larger grid. To do this the main loop of the grid based WFC was implemented. The steps of this is defined in figure 4.

The steps in figure 4 can be summarized as follows.

- **Initialize superpositions**. All tiles are initialized to have superpositions of all possible states.

- **Find lowest entropy**. The current state of the tiles is observed. Then the tile with the lowest entropy (see section 3.1.4) is chosen. In case there are multiple tiles, one is chosen at random.
- **Collapse**. The chosen tile is then collapsed into one of the possible states it can be in. Once again this is chosen at random. Here different states can be differently weighted, altering the probability of them to be chosen. However, for this report, this was not done. See section 5 future work.
- **Propagate**. Since the possible states of other tiles might be affected by this collapse, their superpositions are updated with respect to the newly collapsed tile and the constraints. If a tile only is left with one possible choice, it is collapsed to, resulting in another propagation to its neighbors. This chain reaction continues until all superpositions are updated to their correct values.
- **Conflict?** Depending on the tile set, a situation might appear where there are no possible tiles to use. In this case there is a conflict and a solution can not be produced when using already collapsed tiles.
- **Backtrack** More advanced backtracking is described in section 3.2.5. For the simplest version of the WFC, where the solution space is large, simply restarting the algorithm is a sufficiently fast option here.
- **All Collapsed?**. If there is no conflict, a check is made to see if all tiles have collapsed. If so, a final image has been produced using the tiles it was given while adhering to the constraints and the algorithm is done. Otherwise the observe step is repeated to collapse a new state.

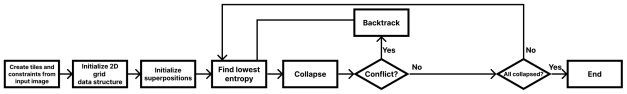

Figure 4: The main loop of the grid based WFC algorithm. Format inspired by Kim et al. [6].

3.1.4 Entropy. In the propagate step there are multiple ways of choosing the next tile. This choice greatly affects how fast the algorithm is able to converge. For instance, if a tile is chosen completely at random, the probability of encountering conflicts later on increases 9. A good way to minimize this risk is defining the entropy of a given state to be equal to the number of possible states in its superposition. This way, the algorithm makes sure to use as much information as possible when making choices for the collapses to make. In section 5, other approaches for choosing the next tile to collapse are discussed.

3.1.5 Results. . After implementing the grid based WFC defined by [4] using figure 2 as input, the image in 5 was produced.

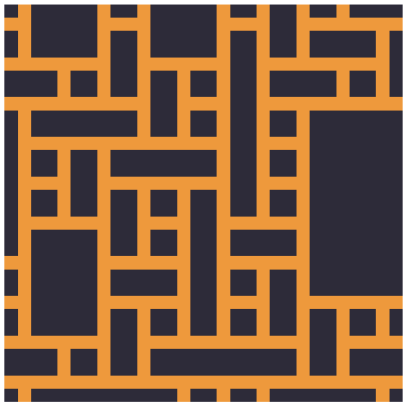

Figure 5: Generated image using the grid based WFC and the source image in figure 2.

3.2 Graph-Based WFC

Based on the results in section 3.1, a more sophisticated version of the WFC was written.

3.2.1 Tiles. A major drawback in the grid-based WFC described by Gumin [4], is its reliance on a source image to contain all the information about the desired results. This results in a limited solution space that is often much smaller than it needs to be where only the edges specified by the image are allowed. For the improved algorithm, the idea of using a source image was replaced by using tiles as input. Since each tile can be next to with each other tile in 4 ways distinct, the number of possible interactions grows substantially when the number of tiles increases. For our example below, there are 14 tiles, resulting in a total of 364 interactions with the other tiles. In the initial version of the algorithm, all these 364 ways of combining tiles would have to be represented in the image which might not be feisible, especially when the number of tiles grows further.

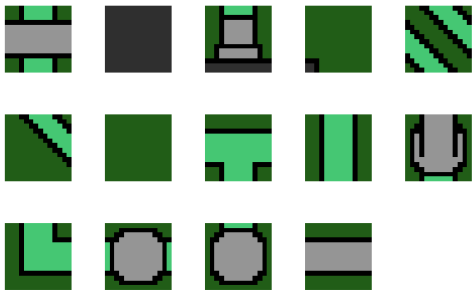

Figure 6: Tiles used for the graph-based WFC in 2D.

3.2.2 Constraints. When not relying on constraints from the source image, the algorithm needs some other way to know the relations between tiles. Defining these constraints can be done in many ways and can greatly affect the final results. For the purposes of this 2d test, compatibility between two tiles in a given direction was tested by comparing specific pixels on the relevant edges. If all tested

pixels had identical colors, the two tiles were allowed to be put next to each other.

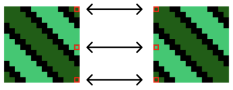

Figure 7: Example for how automatic constraints between two images were generated in a given direction.

3.2.3 Augmented tiles. A big factor in generating as interesting content and for fast result, is allowing for as large of a solution space as possible. In other words, when designing the WFC, it is a good practice to to try to have an as large number of possible interactions between tiles as possible. An important way to do this is supporting augmented versions of tiles. This was implemented such that both mirrors and rotational augmentations were added to the total set of tiles. For the 14 tiles shown in figure 6, this resulted in 26 new tiles after having removed duplicates.

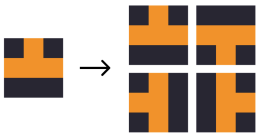

Figure 8: Example for how one tile can be augmented into more tiles. Note that these are not the tiles that were modified for this part of the report.

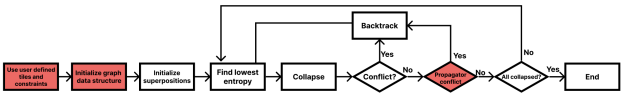

Figure 9: The main loop of the grid based WFC algorithm. Format inspired by Kim et al. [6]. Red boxes indicate modified steps from the original version in figure 4.

3.2.4 Algorithm. In their report, [6] introduces a modified version of the WFC based on graphs. In the original grid based version of the algorithm, the number of neighbors for each tile is constant at four, which limits the expressiveness of the model. However, in the graph-based version the WFC each node is able to handle an arbitrary number of neighbors. As an example Kim et al. [6] illustrate the examples of figure 10 where arbitrary cells can be neighbors.

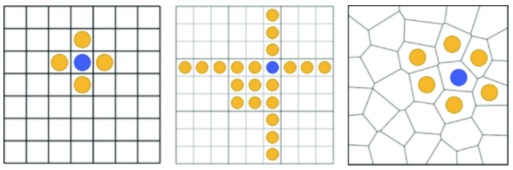

Figure 10: Possible neighbors for a cell from a regular rectangular grid, Sudoku game grid, and Voronoi non-grid. Blue cell is an arbitrary cell. Yellow cells are neighbors. Image created by Kim et al. [6].

To correctly implement the graph-based Wave Function Collapse (WFC) algorithm, additional preprocessing steps are required compared to the standard approach. Instead of merely collapsing a grid and propagating constraints through adjacent cells, it is necessary to initialize a new graph data structure using the input data. This graph forms the foundational structure upon which the algorithm operates. In this implementation, each vertex of the graph represents a tile in the output, and each edge denotes permissible adjacency between two tiles. The edges are labeled according to the direction of adjacency, which is relevant since the system continues to operate within a 2D grid. Consequently, each vertex in the graph is connected by four edges, corresponding to the directions: up, down, left, and right. Note that the expressiveness of the model here is the same as for the grid-based WFC. However, as discussed and shown in section 3.3, it is very simple to update the graph structure and still use the same algorithm.

When it comes to the actual WFC solver, the main idea of the algorithm is the same as discussed in section 3.1.3. However, Kim et al. [6] introduce the need for the propagator conflict check. That is that conflicts are avoided in every propagation. This is needed as the increased number of edges and complexity will result in many more conflicts in the propagation step. The main loop of the graph-based WFC algorithm is illustrated in figure 9.

3.2.5 Backtracking. As the complexity grows and the solution space shrinks, there is an increased risk for contradictions in the produced results. That is, there is an increased risk for situations where a tile has no possible states to collapse into. If the graph that the WFC is applied to, one can simply reset the algorithm when a solution is not found. However, for larger problems, this might be too slow.

To solve this, backtracking can be implemented. This was done by saving each system state on a stack data structure. Then, whenever a contradiction occurred in the algorithm, the last valid solution can be popped from this stack and one can then simply retry with a different tile to collapse.

Finally, to enhance the performance further, a counter was kept such that if multiple contradictions was found in row from the same state, the number of backtracking steps increased by one each time.

3.2.6 Replicating 2D Baseline Results with Graph-Based Approach. After rewriting the grid-based WFC introduced by Gumin [4], in the steps described above, the baseline results that Kim et al. [6] present were recreated (see figure 11). Note that while this image is still in 2D and created using a grid structure, it would not have

been able to been created without the modifications discussed in previous sections of 3.2.

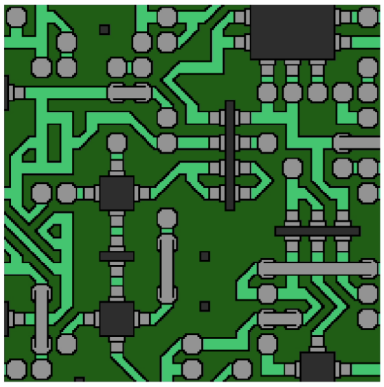

Figure 11: Generated image using the graph-based WFC and the tiles in figure 6.

3.3 Applying the Graph-Based WFC on 3D

For this final part, the goal was to show how the graph-based WFC can be applied on 3D problems.

3.3.1 Creating the input. As in the solution discussed in section 3.2, the input for this problem is also a set of building blocks instead of an image. However, this time meshes were used as tiles instead of 2d sprites. These meshes were downloaded from Donald [1] and created such that each one is of equal size and has a size $1 \times 1 \times 1$ units. These meshes are shown in figure 12.

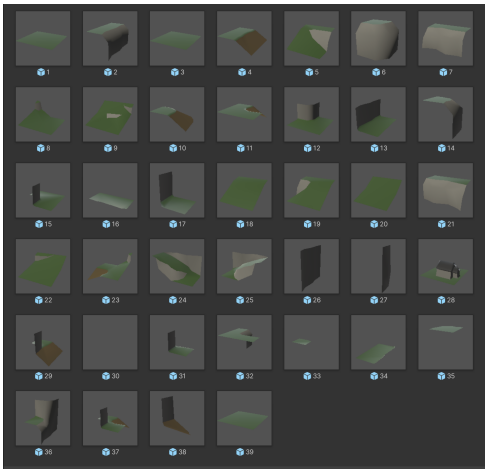

Figure 12: The 40 meshes used for the 3D implementation.

3.3.2 Defining Constraints. The second major adaptation for 3D support involved managing constraints. Instead of comparing individual pixels, a new system was implemented to compare the edges of each mesh. This system identifies all triangles within a mesh that align with any of the six edges of a bounding box. For each side, a set of six "sockets" is defined for the mesh. The system then compares the 40 tiles against each other for all possible interactions. Compatibility is determined if the sockets match, meaning they have identical sets of vertices arranged in the correct order on opposing sides.

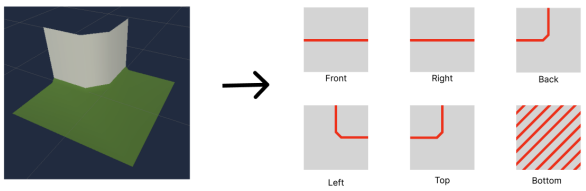

Figure 13: The generated sockets from a mesh.

3.3.3 Algorithm. The final step of adapting the graph-based WFC to 3d, is representing the tiles and new constraints to an appropriate graph structure for the WFC to process. This was done by defining the problem such that each cell in the 3D-grid is represented as a node in the graph. Each cell has 6 edges to the neighboring cells in each direction. These edges are labeled based on their direction. Just by doing these simple changes, the problem is converted to 3 dimensions and the same WFC-graph solver can be used as before.

3.3.4 Final Results. Using the tiles, new constraints, new graph structure and the old graph-based WFC-algorithm the results in figure 14 were produced.

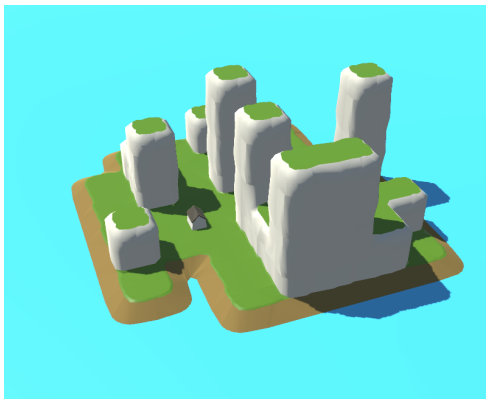

Figure 14: Generated result from the Graph-Based WFC Algorithm applied on a 3d-setting

4 CONCLUSION

Looking at the three produced results, it is clear that the WFC algorithm is an effective way to produce interesting content. Furthermore, one can conclude that the graph-based WFC is a useful and effective extension on the original idea, enhancing the original algorithm and facilitating the creation of more complex content without a large penalty in terms of implementation difficulty or

execution time. Finally, this report showed that the graph-based WFC can be applied on 3 dimensional problems.

5 FUTURE WORK

In this report, both a grid based and a graph-based wave function collapse algorithm were implemented and applied to 2 dimensional problems. Finally the graph-based version was applied on 3-dimensional environments. There are many ways to build upon these results.

5.1 Expanding the Algorithm

5.1.1 Entropy Adjustments: In this report, the entropy of a given tile was determined as being equal to the number of states that the tile in question could collapse into. In other words, when collapsing a state the tile with the lowest number of possible states was always chosen. However, this is not the only way to approach this. For instance, instead of always choosing the lowest entropy tile, the best candidates of tiles could be chosen with some probability scaled based on their entropy. This could then be regulated using some temperature hyperparameter to regulate the amount of randomness allowed in the model. Doing this would increase the expressiveness of the model, and might introduce some more interesting results.

In their report, Fu et al. [2] showed how a similar approach can be very effective when improving the quality of generated text from deep learning models. In each iteration of text syntesis, the algorithm generally has access to the probabilites of the next word, and by introducing some randomness (regulated by a temperatureparameter) when choosing this next word, much better results were found [2]. A similar approach could be examined for picking the next tile with the WFC.

5.1.2 Weighted Tiles: In the implementation of this report, the different tiles were always chosen uniformly random. However, there can be many use cases where it might be a need for picking different tiles. For instance, in the island example of the report, one might want to regulate the amount of flat-land relative to mountains. A future study could examine how to effectively implement a weighted system, where the probability of choosing different tiles is additionally affected by some weight parameter. This would enhance the usefulness of the model furher and might provide more interesting results.

5.1.3 Sequential WFC. . In this report there was only a singular WFC collapse performed per generation. That is, the loop was ran only once to produce the entire result. However, there has been recent examples for for how to combine the algorithm in multiple sequences to generate interesting results. As an illustrative example, the WFC could first be used to generate the floorplanning of the house. Then a second and different WFC could decorate the interior. Gumin [3] has already done some sequential WFC in his repository MarkovJunior[3] that illustrate the potential of the idea, however there is a great potential for more thorough research.

5.1.4 WFC and other PCG algorithms. Besides combining multiple WFC runs, it would also be a good future study to examine how effective the algorithm can be when used in combination with other Procedural Content Generation (PCG) algorithms. For instance, integrating WFC with Genetic Algorithms could provide a hybrid approach where the structure provided by WFC is refined by the evolutionary strategies of GAs. This could be particularly useful in complex environments where adaptive solutions are needed. Similarly, combining the WFC with automatic mesh generating algorithms like marching cubes could be investigated.

5.2 Potential Perceptual Studies

To effectively evaluate the performance of WFC-generated environments, a perceptual study could be conducted. This study would probably require the generated worlds being interactable, something that could easily be implemented with default unity controllers.

The study should involve participants ranging from casual enthusiasts to professionals, to gather feedback on the generated scenarios. Participantswould be exposed to various scenarios created by the WFC algorithm, and their interactions with these environments would be recorded. The evaluation process would then be divided into three steps:

- Behavioral Observation: Observation sessions would be conducted to monitor how participants navigate and interact within the environments. Important metrics, such as the amount of time spent in different sections, frequency of interactions with specific features, and observable signs of confusion or frustration, would inform the assessment of the algorithm.
- Surveys and Questionnaires: After engaging with the environments, participants would be asked to complete surveys to asses their experience with things like visual appeal, realism and navigability. Additionally, these surveys would include open-ended questions designed to get more personal feedback.
- Interviews: In-depth interviews would be carried out as follow-up sessions to gain deeper insights into the participants' experiences. These interviews would explore subjective perceptions and emotional responses that might not be fully captured through surveys alone.

5.2.1 Expected Outcomes and Impact. The insights garnered from this perceptual study would be pivotal in refining the WFC algorithm. For instance, if participants consistently report that certain elements within the environments feel unrealistic or visually unappealing, these elements could be earmarked for modification or removal in future iterations of the algorithm. Conversely, features that significantly enhance user engagement and satisfaction could be emphasized and further refined.

Moreover, the study could reveal important correlations between user satisfaction and specific algorithmic parameters, enabling more precise adjustments that tailor the algorithm to different use cases. This could involve fine-tuning the algorithm to better balance aesthetic appeal with practical functionality, or to increase the challenge and engagement in game levels while maintaining easy navigability.

By directly engaging end-users in the development process through these perceptual studies, the project not only enhances the technical aspects of the WFC but also ensures that it aligns with the practical needs and preferences of its intended audience.

REFERENCES

[1] Martin Donald. 2020. *Island Project Files.* https://www.patreon.com/bolddunkley

[2] Zihao Fu, Wai Lam, Anthony Man-Cho So, and Bei Shi. 2021. A theoretical analysis of the repetition problem in text generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 12848–12856.

[3] Maxim Gumin. 2022. *MarkovJunior.* https://github.com/mxgmn/MarkovJunior

[4] Maxim Gumin. 2022. *Wave Function Collapse Algorithm.* https://github.com/mxgmn/WaveFunctionCollapse

[5] Isaac Karth and Adam M Smith. 2017. WaveFunctionCollapse is constraint solving in the wild. In *Proceedings of the 12th International Conference on the Foundations of Digital Games*. 1–10.

[6] Hwanhee Kim, Seongtaek Lee, Hyundong Lee, Teasung Hahn, and Shinjin Kang. 2019. Automatic generation of game content using a graph-based wave function collapse algorithm. In *2019 IEEE Conference on Games (CoG)*. IEEE, 1–4.

[7] Yuhe Nie, Shaoming Zheng, Zhan Zhuang, and Xuan Song. 2023. Extend Wave Function Collapse to Large-Scale Content Generation. arXiv:2308.07307 [cs.AI]

[8] Arunpreet Sandhu, Zeyuan Chen, and Joshua McCoy. 2019. Enhancing wave function collapse with design-level constraints. In *Proceedings of the 14th International Conference on the Foundations of Digital Games*. 1–9.