CG-WFC: A Hybrid Cyclic-Graph & WFC Method for Designer-Guided and Replayable Procedural Content Generation

Laurent Voisard
laurent.voisard@mail.concordia.ca
Concordia University
Montréal, QC, Canada

Christiano Politowski
cristiano.politowski@ontariotechu.ca
Ontario Tech University
Oshawa, ON, Canada

Fabio Petrillo
fabio.petrillo@etsmtl.ca
École de technologie supérieure - ÉTS Montréal
Montréal, QC, Canada

Yann-Gael Géhéneuc
yann-gael.gueheneuc@concordia.ca
Concordia University
Montréal, QC, Canada

ABSTRACT

Procedural generation techniques offer powerful ways to produce varied game environments, but they often struggle to balance designer control with emergent replayability. In this paper, we present a hybrid generation method that combines the pattern-based constraints of Wave Function Collapse (WFC) with the structural expressiveness of cyclic graph generation. This approach allows designers to define high-level narrative or spatial structures through graph grammar, while WFC ensures locally coherent and aesthetically consistent layouts. By decoupling global mission flow from local content assembly, the method enables both fine-grained authoring and significant variability across playthroughs.

KEYWORDS

Procedural Content Generation, Wave Function Collapse, Mission Graphs, Designer Control, Replayability, Game Level Generation

ACM Reference Format:
Laurent Voisard, Christiano Politowski, Fabio Petrillo, and Yann-Gael Géhéneuc. 2025. CG-WFC: A Hybrid Cyclic-Graph & WFC Method for Designer-Guided and Replayable Procedural Content Generation. In . ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Procedural content generation (PCG) has become a central technique in the development of games that emphasise replayability, scalability, and systemic design [7]. From the dungeon layouts of early titles like Rogue[1] to the expansive, open worlds of Minecraft[2], procedural methods have allowed developers to produce large amounts of content efficiently. Among these, dungeon-like structures have been particularly influential in both research and industry, providing a clear gameplay framework and a flexible canvas for procedural methods[8].

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

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

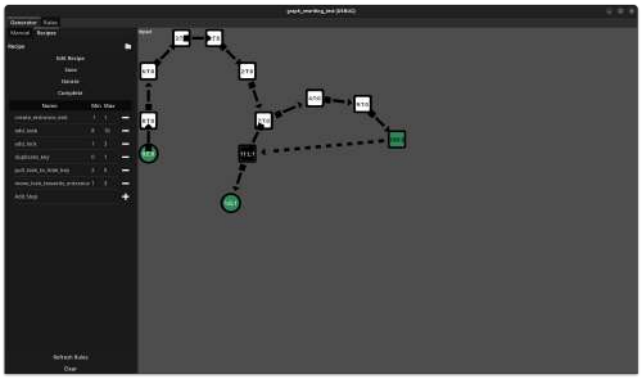

Figure 1: The CG-WFC tool. On the left, a recipe is loaded and iterated until complete, producing the mission graph on the right.

Despite their advantages, procedural methods frequently struggle to balance designer control with variability. Automated systems can generate vast amounts of content but often fail to reproduce the pacing, structure, and intentionality found in handcrafted levels. As noted by van der Linden et al. [8], many dungeon generation techniques offer either fine-grained structure at the cost of diversity or high variability with limited control over gameplay flow.

To address this tension, we propose a hybrid procedural generation method that combines Wave Function Collapse (WFC) with cyclic graph-based mission structures, CG-WFC. This two-layer approach separates global mission structure from local spatial instantiation, allowing designers to define progression and rhythm while still benefiting from procedural variety at the tilemap level. This method aims to offer both structured authorial intent and replayable content, particularly suited to genres where mission structure and exploration are central design pillars, such as role-playing games or roguelikes.

In the sections that follow, we go over background 2, the method 4, present some early results 5, discuss the results and the current state of the tool 6, and finally we conclude the paper by going over the main contributions 7.

[1] Rogue - Wikipedia [2] Minecraft - Wikipedia

2 BACKGROUND

2.1 Procedural Content Generation

Procedural content generation is widely used in games to produce environments, missions, and other game assets through algorithmic means rather than manual authoring. This enables developers to scale content production efficiently, reducing reliance on large teams of level designers. PCG techniques have been applied across a wide range of genres, from early roguelikes to contemporary open-world experiences, where procedural methods create large, varied spaces that enhance replayability and player-driven exploration.

However, while PCG excels at producing content quickly and at scale, it often lacks the fine-grained control and narrative intentionality of handcrafted environments. Hand-authored levels allow designers to carefully shape pacing, player experience, and thematic beats, whereas purely procedural methods tend to focus on structural feasibility or pattern matching without embedding explicit design goals. This creates a fundamental tension between automation and authorial control, which has been widely recognised as a key challenge in procedural generation research [8].

Various approaches have emerged to address this issue, including constraint-based generation, semantic tagging, and mixed-initiative design tools that allow designers to guide algorithmic generation[8]. Our work builds on this trajectory by combining procedurally generated mission graph structures and WFC, aiming to bridge high-level design intent with low-level procedural detail.

3 RELATED WORK

3.1 Cyclic Graph Generation

Dormans et al. [2, 3] elaborated a procedural mission generation technique based on graph grammars, enabling the automatic creation of mission graphs, a structured sequence of tasks that players must complete to finish a level. This approach draws inspiration from classic dungeon design in The Legend of Zelda series, where players encounter obstacles or puzzles that can only be overcome by exploring other areas of the world, acquiring key items or abilities, and then returning to unlock previously inaccessible paths. By encoding such progression patterns into graph transformation rules, this method allows designers to express high-level structural motifs such as branching questlines, loops, and parallel objectives.

The generation of complex mission structures is formalised using graph grammars, a specialised form of generative grammar and rewrite systems that operate on nodes and edges [5]. In this context, the mission graph is modelled as a directed graph representing the logical flow of tasks, challenges, and dependencies required to complete a level. A simple, linear sequence of objectives can be progressively transformed into a non-linear structure through a series of rewrite steps, introducing branching paths, detours, and interdependencies. One particularly effective technique involves adding locks and keys, which require players to revisit or explore alternate branches, turning a straightforward path into a more engaging, exploratory mission structure (see Figure 3).

Dormans chains multiple rules together, calling this process *Recipes*. Each rule in a recipe can be applied in one of three ways: a fixed number of times, a random number of times within a range, or to each identified pattern within the target graph of a grammar rule, similar to L-Systems [6]. In the end, each graph generated by a recipe will be different but share common elements.

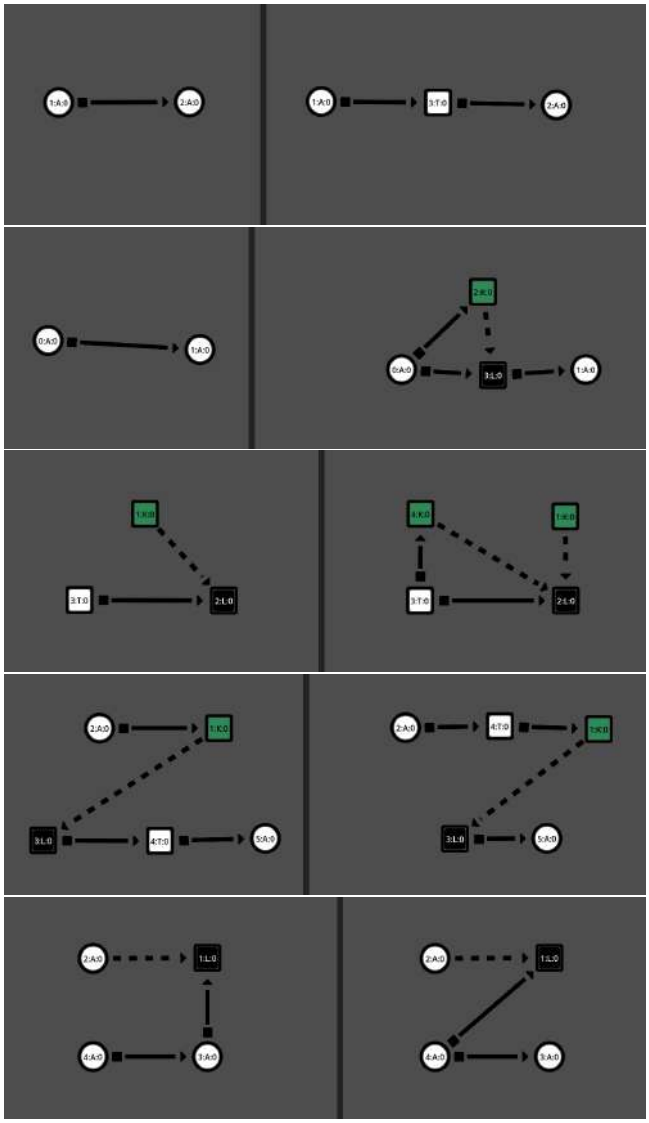

Figure 2: Designer authored mission graph rules. The left side of the rules is the lookup pattern, the right side of the rules is the output. In order from top to bottom: Add a task, add a key and lock, duplicate key, hide key, move lock towards the entrance.

3.2 Wave Function Collapse

Wave Function Collapse (WFC) [4] has emerged as a powerful procedural generation technique for producing locally coherent layouts from example data. By learning and reproducing spatial adjacency patterns, WFC enables designers to generate rich and visually consistent environments with minimal manual authoring. It has been widely adopted for tilemap generation, environment

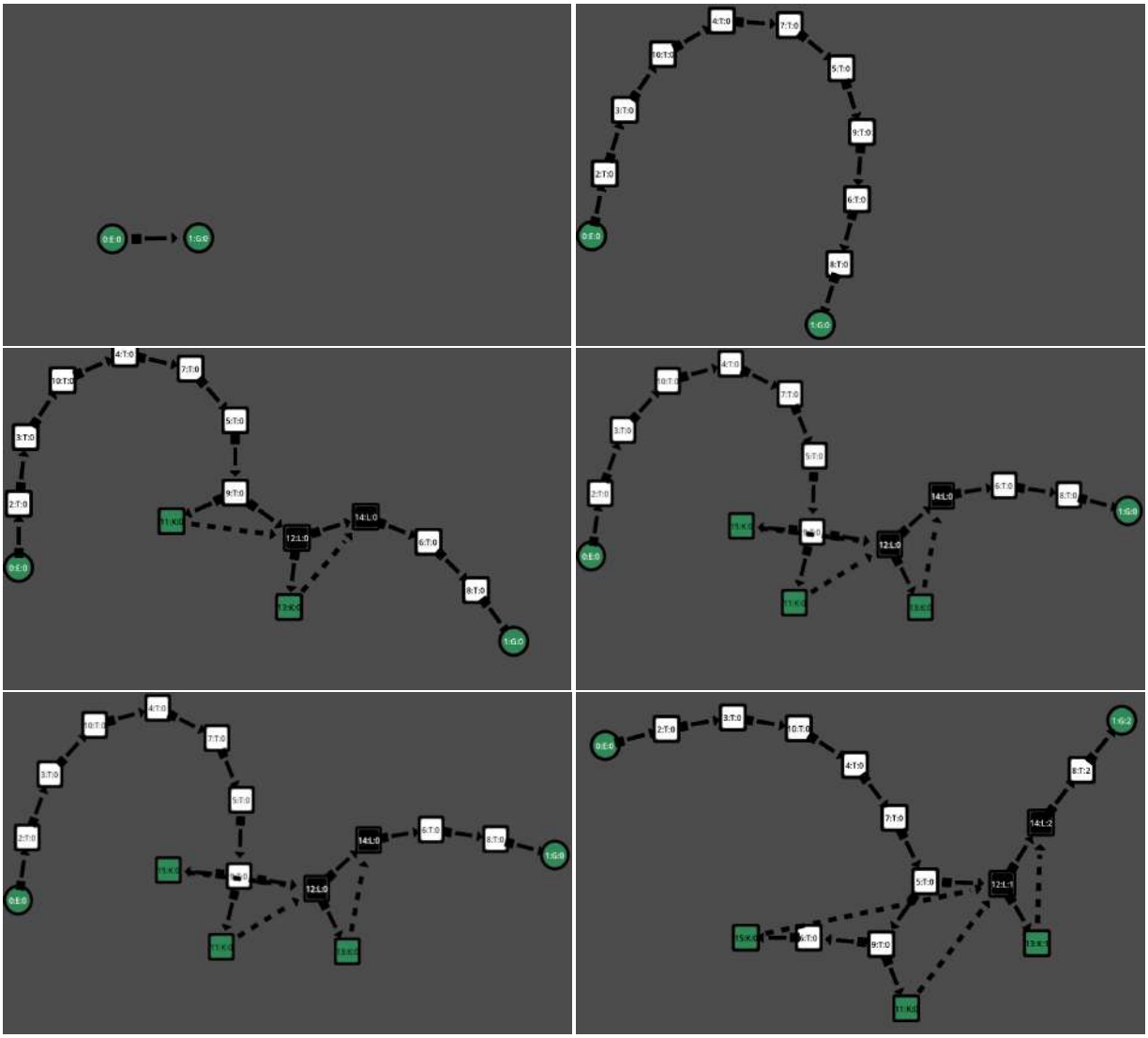

Figure 3: Iterative process of applying rules from a mission graph recipe. First, an entrance and goal are generated, next, a random value from 8-10 tasks are added. The next step adds 1-3 lock and keys. The following step can duplicate a key 0-1 time. The next step pulls a task to hide the key further from the main path 3-5 times. The last step moves the lock one room closer to the entrance 1-3 times.

design, and similar layout problems due to its simplicity and expressive potential.

Despite these strengths, WFC provides limited control over global structure. While the algorithm ensures local consistency, it does not inherently encode higher-level concepts such as pacing, progression, or thematic flow. As a result, levels generated purely through WFC can lack the intentional rhythm and narrative framing that hand-authored structures provide.

Several extensions have sought to address these limitations. One notable example is hierarchical semantic WFC [1], which introduces an abstraction layer by grouping tiles into semantic categories, for example: forests, deserts or cities. This allows designers to work with abstract tiles in an initial pass, which are then refined into more detailed patterns during a second collapse stage.

4 METHOD

Our approach builds upon the complementary strengths of cyclic graph generation and WFC by integrating them into a two-layer procedural generation pipeline. This hybrid structure enables designers to maintain high-level control over mission topology while delegating the local spatial detail to an example-based generation process.

We implement a prototype of the hybrid system in a 2d tile-based environment[3] using the Godot[4] game engine. The source code is publicly available on Github[5]. For this tool, we implement the mission graph layer ourselves, referring to the papers from Dormans et al. [2, 3], and for the WFC, we use the most popular tool for Godot, which has 424 stars on Github[6]

4.1 Mission Graph Layer

The first step of our pipeline is to define a mission graph using cyclic graphs generated via designer-authored graph grammar rules. Each node can represent rooms, tasks and more, while each edge keeps track of spatial and logical relationships. Designers can create patterns such as branching paths, lock and key generation, loops and more. To systematically create similar levels, designers can author *recipes*, which are a set of rules applied one after the other (Figure 1). Each node can contain metadata such as encounter types or spatial constraints, which will carry over the WFC pass. Figure 2 lists a few fundamental graph grammar rules in the creation of our levels.

To create a level from a designer-authored recipe, each rule in the recipe is applied a random amount from a range onto the graph. The example from figure 3 shows the process of the recipe iteration. First, a few tasks are added to create a linear mission of a fixed length. It then incorporates 1-3 keys and locks, can duplicate them, and move rooms around to hide keys and move the locks closer to the entrance. Finally, we apply a force-based layout to the graph in order to separate nodes from each other.

4.2 WFC Layer

The second layer uses WFC to generate locally coherent spatial layouts corresponding to each mission graph node generated in the previous step. A pattern library is learned from designer authored samples (Figure 4, encoding adjacency rules implicitly through observed patterns. As a first pass, the generation process will paint the location of each room in the mission graph as well as its connections. Once the layout is set, WFC can replace the tile hierarchies with their appropriate tiles based on the designer examples.

This decoupling between global structure and local generation allows designers to maintain control over progression and mission rhythm, while still obtaining high replayability through random WFC instantiations.

5 RESULTS

Using the process described in section 4.1, we generate a mission graph from a designer-authored recipe (Figure 5 (a)). Next, we can draw hierarchical WFC levels[1] of the dungeons (Figure 5 (b)), rooms and exteriors in a first pass, and then allow WFC to fill in the details in the tilemap (Figure 5 (c)). The whole process can take around ten seconds, and the majority of the time is consumed by the WFC algorithm, around 9-10 seconds, whereas the mission graph generation can take up to 100ms. These values are gross estimations based on personal observations and can vary from system to system.

[3] https://kenney.nl/assets/tiny-dungeon
[4] https://godotengine.org/
[5] https://github.com/LVoisard/graph-reconstruction-godot.git
[6] https://github.com/AlexeyBond/godot-constraint-solving.git

Moreover, the outputs produced through this pipeline illustrate how small variations in the mission graph recipe can lead to significant differences in the resulting layouts. For example, altering the number of tasks or the placement of lock-and-key structures produces markedly different progression paths while retaining an overall coherent structure. Similarly, because the mission graph constrains the semantic layout, multiple WFC instantiations from the same recipe yield distinct yet structurally consistent dungeons. This demonstrates one of the method's key strengths: designers can shape high-level structure once and obtain a diverse set of replayable levels without reauthoring content.

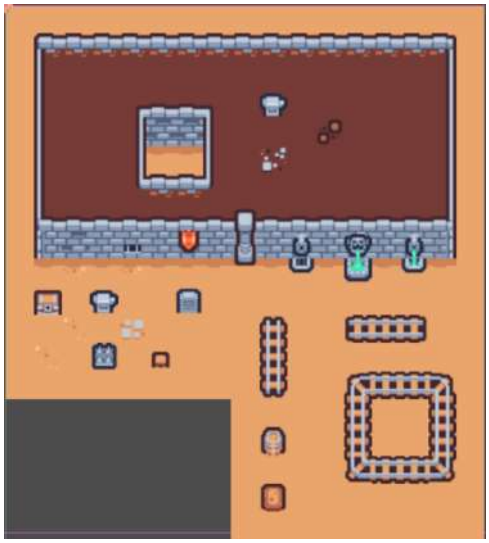

Figure 4: Sample input made by a designer used by WFC

6 DISCUSSION

In creating this hybrid approach, we aimed to address two key design goals in procedural generation:

- **Designer Control:** The generated mission graphs enable explicit control over structure, pacing, and rhythm, ensuring that generated content aligns with designer intent rather than being purely random.
- **Replayability:** The combination of mission graphs and WFC introduces variation at both the structural and local layout level, increasing the number of unique playthroughs that can emerge from a single authored mission recipe.

While the preliminary results of the approach meet these objectives, the method also introduces new challenges. Connectivity constraints between WFC layouts can become brittle for dense graphs, requiring careful placement of entry/exit anchors. Moreover, WFC itself remains sensitive to the quality and diversity of its input patterns; limited or homogeneous samples can lead to repetitive outputs and reduce replayability.

More critically, while the current system successfully generates mission graphs and spatial layouts, the output levels are not yet fully playable. The semantic metadata defined in the mission graph, such as room functions, encounter types, or key-lock relationships, is not

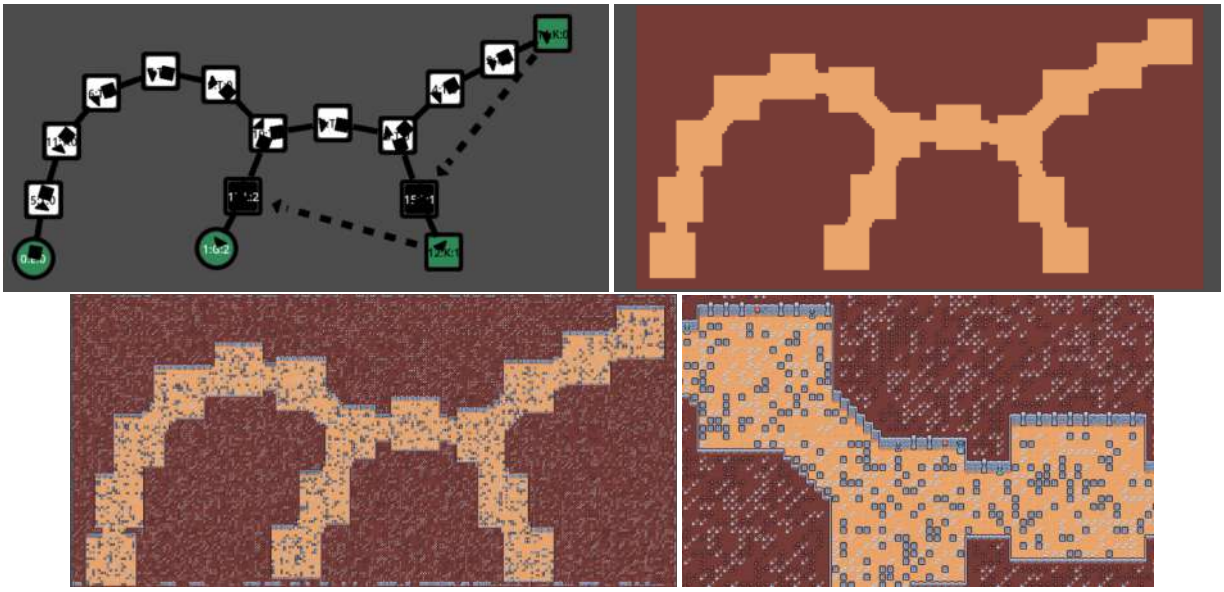

Figure 5: Once the mission graph has been generated (a), it is grossly drawn to the tilemap (b). WFC uses the template as a constraint and replaces the classes with the appropriate tiles(c), light brown: room, dark brown walls.

currently propagated into the final WFC-generated environment. As a result, the generated levels contain the intended structural rhythm but lack gameplay elements that would make them functionally meaningful. Integrating this metadata into the spatial layer is a crucial next step toward making the generated levels interactive and game-ready.

Another limitation lies in the choice of the WFC generation tool. The current implementation uses a Godot WFC plugin[7], which enforces full-area tiling rather than the collection of rooms. This creates unnecessary filler space outside of intended rooms and corridors, which can greatly hinder generation time depending on the total extent of the level space. Moreover, the hierarchical feature in the tool does not account for tile probability, which explains the abnormal distribution of tiles in the final output compared to the sample input (See figures 5 and 4). A custom or modified WFC implementation would be needed to support more fine-grained, region-specific generation.

Finally, while this approach offers a promising balance between designer control and replayability, empirical evaluation is still missing. Future work should investigate how players perceive levels generated through this hybrid method, including their sense of narrative structure, navigability, and replay value. Likewise, designer studies could assess how effectively this approach supports authoring goals compared to purely handcrafted or purely procedural workflows.

7 CONCLUSION

Procedural generation systems can produce vast amounts of content efficiently, but often struggle to replicate the pacing, structure, and intent of handcrafted levels. This work set out to bridge that gap, giving designers meaningful control over level structure while retaining the expressive power of procedural methods.

We presented CG-WFC, a hybrid generation approach that combines Wave Function Collapse with cyclic graph-based mission structures to balance designer control and replayability. By separating global mission structure from local pattern instantiation, the method preserves authorial intent across structural, spatial, and thematic layers while supporting varied, replayable layouts. This makes it particularly suited to genres where mission structure plays a central role, such as role-playing games.

Future work will focus on integrating mission graph metadata into WFC outputs to produce fully playable levels with embedded gameplay semantics, optimising WFC for region-constrained generation, and conducting empirical studies to evaluate the perceived structure, navigability, and replay value of the generated content.

REFERENCES

[1] Alaka, S., and Bidarra, R. Hierarchical semantic wave function collapse. In *Proceedings of the 18th International Conference on the Foundations of Digital Games* (New York, NY, USA, 2023), FDG '23, Association for Computing Machinery.

[2] Dormans, J. Cyclic generation. In *Procedural Generation in Game Design*. AK Peters/CRC Press, 2017, pp. 83–96.

[3] Dormans, J., and Bakkes, S. Generating missions and spaces for adaptable play experiences. *IEEE Transactions on Computational Intelligence and AI in Games 3*, 3 (2011), 216–228.

[4] Gumin, M. Wave Function Collapse Algorithm, Sept. 2016.

[5] Rekers, J., and Schurr, A. A graph grammar approach to graphical parsing. In *Proceedings of Symposium on Visual Languages* (1995), IEEE, pp. 195–202.

[6] Rozenberg, G., and Salomaa, A. *The mathematical theory of L systems*, vol. 90. Academic press, 1980.

[7] Shaker, N., Togelius, J., and Nelson, M. J. Procedural content generation in games.

[8] van der Linden, R., Lopes, R., and Bidarra, R. Procedural generation of dungeons. *IEEE Transactions on Computational Intelligence and AI in Games 6*, 1 (2014), 78–89.

[7] https://godotengine.org/