# How much Tetris can Wave Function Collapse put up with?

Rolf Piepenbrink, Rafael Bidarra
rpiepenbrink@proton.me|R.Bidarra@tudelft.nl
Computer Graphics and Visualization Group
Delft University of Technology
The Netherlands

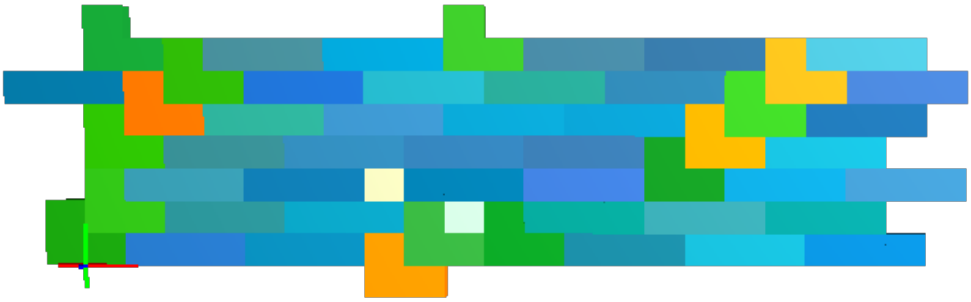

**Figure 1: A Tetris-like texture generated with WFCNUT**

## KEYWORDS

wave function collapse, procedural content generation, constraint programming

**ACM Reference Format:**
Rolf Piepenbrink, Rafael Bidarra. 2024. How much Tetris can Wave Function Collapse put up with?. In *Proceedings of Foundations of Digital Games 2024 (FDG 2024).* ACM, New York, NY, USA, 2 pages. https://doi.org/XXXXXXX.XXXXXXX

## 1 INTRODUCTION

Wave Function Collapse (WFC) is a local constraint solver for procedural content generation [1]. WFC has proven to be a flexible and versatile PCG method capable of generating various output from the same input texture. Its simplicity and ability to generate high quality output are valued among researchers and game developers alike. It has been applied in several domains, ranging from map generation to texture synthesis.

Currently, most approaches limit WFC to using tiles of the same size, each spanning one cell in a grid. While this works well for the aforementioned domains, more is to be gained by extending a tile's boundaries. One could for instance allow tiles that span multiple cells, which we call *non-uniform tiles*. By supporting non-uniform tiles, doors are opened to many other domains, such as constrained piece-wise construction of LEGO models, creation of Tetris-like textures, or modular generation of floor plans and buildings with (prefabricated) components of different sizes.

In this hands-on demo, we introduce Wave Function Collapse on Non-Uniform Tiles (WFCNUT), an enhanced variation of WFC with support for non-uniform tiles. WFCNUT can infer non-uniform tile adjacency constraints through masking, and follows an approach that makes it suitable for 3D applications.

## 2 RELATED WORK

Since Gumin's publication of WFC in 2016 [1], it has been a topic of great interest, especially in the PCG research field and in games. In the PCG research field, several researchers have addressed tiles spanning multiple cells.

Newgas' take on WFC, called Tessara, supports tiles spanning multiple cells, which he calls *big tiles*, through painting-based tile annotation [4]. Big tiles are split into multiple single tiles where adjacency constraints between those tiles maintain the tile's structure. In the same paper, Newgas explains how Tessera is not limited to a square grid, but extends to other grids with a fixed number of neighbours per cell. While Tessera appears to work well for convex tiles, it is unclear whether and how Tessera extends to concave tiles. WFCNUT takes inspiration from Newgas' insights of subdividing tiles into tiles spanning one cell and using constraints to maintain the structure, while taking a strictly different approach in order to be able to support concave tiles.

Stålberg, the creator of the WFC method used in Bad North, follows a similar approach, where the sides of a tile dictate which other tiles may be adjacent [3]. Levels in Bad North can be made from a tileset of over 300 tiles, some of which span multiple cells. The main purpose of those tiles is to create smooth transition between regions of a level. For this reason, these tile are still relatively small, spanning no more than two cells.


*FDG 2024, May 21–24, 2024, Worcester, Mass.*

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

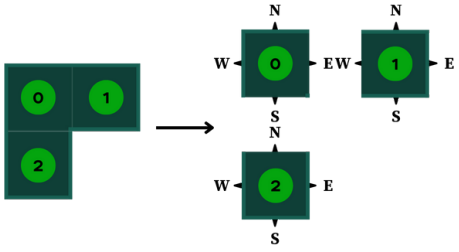


**Figure 2: Atomization of a (concave) corner tile.**

## 3 METHODOLOGY

For WFCNUT to support non-uniform tiles, two main challenges had to be overcome. The first challenge concerns storing information of non-uniform tile placement. Since tiles can span multiple cells, one cell does not necessarily contain one tile. Yet, each cell should contain information of which and how a tile covers said cell. The second challenge relates to defining adjacency constraints of non-uniform tiles. Defining these adds a layer of complexity, since non-uniform tiles can have multiple neighbours per (cardinal) direction. They can also have an arbitrary structure, so that finding the valid configurations in which two non-uniform tiles are adjacent is non-trivial. The former challenge can be solved with tile atomization, while the latter challenge can be overcome through a sliding window approach with masking.

### 3.1 Tile atomization

The solution to first challenges lies with tile atomization, similar to Newgas' approach [4]. Tile atomization reduces the non-uniform tiles problem into one that can be integrated with WFC and works as follows. Each tile is considered 'split' into atoms that each cover one cell. Then, each atom is assigned a unique identifier and an index relative to the tile. Next, to represent a tile's structure, adjacency constraints between atoms within the same tile are enforced, which we call inner *atom adjacency constraints*. With these steps, a non-uniform tile can be represented by a set of atoms and a set of adjacency constraints, as shown in Figure 2.

### 3.2 Atom adjacency constraints

In order to formalize tile adjacency constraints as atom adjacency constraints, a clear definition of tile adjacency in the context of atoms is required. Two non-uniform tiles are said to be *adjacent* when an atom of one tile touches an atom of the other tile. It is thus possible that multiple configurations of those tiles are valid for a given direction, if multiple atoms have no inner atom adjacency constraints for that direction.

Each tile adjacency constraint can be expressed as a set of atom adjacency constraints by considering all possible valid configurations of the two tiles in question, which can be found following a linear sliding window approach.

For each configuration, the adjacent pairs of atoms are recorded as an adjacency constraint. For concave tiles, the slider may need to be offset in order to be adjacent. These offsets can be found through *void masking*. Void masking is a technique that counts the number of empty cells in the bounding box of the tile, also referred to as

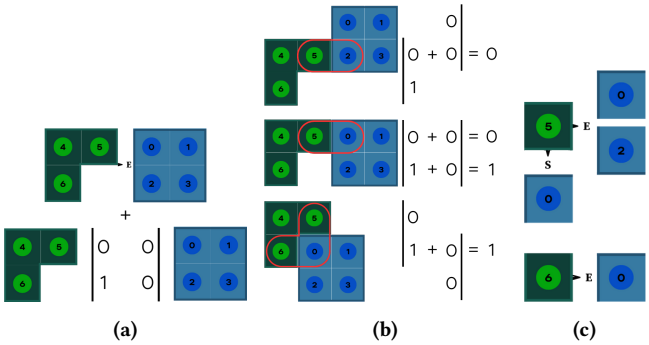


**Figure 3: Tile adjacency constraint transformation: (a) tile adjacency constraint and corresponding void masks; (b) offset tiles through summed overlaid void masks; (c) obtained atom adjacency constraints.**

*voids*, for each direction relative to the tile. This precomputed mask can be passed along with the sliding process, as shown in Figure 3a, where the required offset can be found by finding the minimum sum of overlaid void masks, as shown in Figure 3b. This then results in a set of atom adjacency constraints, as shown in Figure 3c.

Supporting the representation of non-uniform tiles has the advantage of being more restrictive of the search space. For instance, collapsing a cell into an atom of a given tile enforces other cells to contain the remaining atoms of that tile. This multiple cell collapse can thus be determined at once, which positively affects scalability.

## 4 VISUALIZATION AND INTERACTION

In this demo, we present a hands-on experience with the WFCNUT prototype with a variety of tilesets, both 2D and 3D. Special care is given to making the visualization interactive, with options to interrupt and inspect the process during runtime. Furthermore, a form of backtracking is provided, where the user can choose to manually or automatically create savepoints, which can be used to revert to an earlier state of the process providing more control.

Besides that, for the initialization, one can easily specify, select or change a subset of tiles from the tileset. For instance, in the demo we will show how to create new non-uniform tiles or load existing ones. One can also set the adjacency constraints between non-uniform tiles through simple toggles in a familiar table-based approach from within the views of tile adjacency constraints [2]. To provide a greater sense of understanding, the graphical user interface also includes views for atom adjacency constraints. These features should lower the barrier of entry and make WFCNUT more accessible.

## REFERENCES

[1] Maxim Gumin. 2016. *Wave Function Collapse Algorithm.* personal repository. https://github.com/mxgmn/WaveFunctionCollapse
[2] Thijmen S. L. Langendam and Rafael Bidarra. 2022. miWFC - Designer empowerment through mixed-initiative Wave Function Collapse. In *Proceedings of PCG 2022 - Workshop on Procedural Content Generation for Games.* ACM Press.
[3] Oskar Stålberg. 11 Jun 2018. *EPC2018 - Oskar Stålberg - Wave Function Collapse in Bad North.* Youtube. https://www.youtube.com/watch?v=0bcZb-SsnrA
[4] Adam Newgas. 2021. Tessera: A practical system for extended WaveFunctionCollapse. In *Proceedings of the 16th International Conference on the Foundations of Digital Games.* 1–7.