miWFC - Designer empowerment through mixed-initiative Wave Function Collapse

Thijmen S. L. Langendam
thijmenlangendam@gmail.com
Delft University of Technology
Delft, The Netherlands

Rafael Bidarra
R.Bidarra@tudelft.nl
Delft University of Technology
Delft, The Netherlands

ABSTRACT

Wave Function Collapse (WFC) is a powerful generative algorithm, able to create locally-similar output based on a single example input. One of the inherent limitations of the original WFC is that it often requires users to understand its inner workings, and possibly make their own ad-hoc mods, to achieve satisfactory results. Besides distracting from your creative task, this strongly reduces the algorithm's effective usefulness to a small group of technical users. We propose *miWFC*, a novel mixed-initiative approach to WFC aimed at overcoming these drawbacks. Its main focus is on providing intuitive control to its users, in a way that matches their usual creative workflow. Among its main features, this approach provides (i) interactive navigation through design history, including controlled backtracking, (ii) precise manual editing of the output for direct expression of design intent, and (iii) interactive manipulation of tile weights, to tweak the global appearance of the output. We evaluated a prototype implementation of our approach among game artists and other creative professionals, and concluded that its features were largely considered useful and supportive of their creative work.

KEYWORDS

procedural content generation, mixed-initiative, human-computer interaction, interaction design, wave function collapse, level generation, texture synthesis, constraint solving

1 INTRODUCTION

Most game level designers and artists do hard and unstructured creative work. Procedural Content Generation (PCG) methods have often been proposed as a powerful tool to assist them [15]. However, to be effectively helpful, such means have to empower their artistic users, not only respecting but amplifying their creative freedom [16].

Mixed-initiative approaches propose a type of human-computer interaction in which the computer and the human user alternatively take steps towards the desired goal. Mixed-initiative PCG systems have long been proposed as promising tools for a variety of purposes in game development, from game level [18, 9, 2, 23] to complete game world generation [17, 5, 12]. However, the challenges of combining PCG with manual editing of its output have also been pointed out [14].

Wave Function Collapse (WFC) is a PCG method that has recently gained widespread popularity [4, 7]. The original WFC partly resembles the model synthesis work of Paul Merrell [10], which was initially geared towards procedurally generating complex 3D models based on one input model. WFC simplified and facilitated its use and application, particularly for image synthesis purposes. However, this comes at the cost of providing little control on the generative direction followed by the algorithm.

Our work addresses the research question: *what does it take to convert WFC into a mixed-initiative PCG method?* More specifically, we explore how to adapt the WFC algorithm to support and integrate a number of interactive features that more appropriately suit the usual creative workflow of game level designers and artists. In this paper, we propose several of these features, discuss how they work, and present some results of the evaluation of their implementation in our *miWFC* prototype application.

2 RELATED WORK

We briefly discuss the original WFC algorithm, its strengths and weaknesses, some of its extensions proposed so far, as well as other related research work.

2.1 Basic Wave Function Collapse

The original WFC [4] generates bitmaps that are locally similar to an input bitmap. This means that the output only contains patterns of pixels that are present in the input.

To generalize the WFC algorithm, one has to abstract above the particular notion of bitmap. For this, it is convenient to first introduce a few notions as follows:

- a **tile** is a pattern with a distinct combination of several elementary 'space subdivision units' (e.g., pixels, voxels, letters, etc.); it is identified and extracted from the algorithm input;
- a **cell** is the basic building block of the algorithm output space; initially, any tile can potentially be assigned to each cell, hence the (quite remote) quantum analogy of WFC: every cell 'simultaneously contains all possible states' (i.e., potential tiles) until either you 'collapse' it (i.e., assign it one concrete tile) or its neighbour cells constrain its allowed 'states'.

The generic WFC algorithm is presented in Algorithm 1. It initially analyses its input, detecting and extracting from it both a number of patterns (the available tiles) and identifying the constraints (e.g., existing adjacencies) between them (1). Subsequently, the algorithm will iteratively select where to continue collapsing (3), how to collapse (4), and propagate that collapse to the neighbour cells (5). Eventually, the algorithm stops (6) when either all cells have been collapsed or some conflict took place. This simple and generalized version of WFC will be used throughout the paper:

Stated in this way, WFC's simplicity makes it quite attractive for being applied to generative problems in many domains. Among its strongest advantages, one can point out:

```
1 initialize algorithm (building tile and constraint tables)
2 **repeat**
3     choose next cell to collapse
4     choose which tile to collapse it to
5     propagate
6 **until** *all cells have collapsed or a conflict occurs*;
```

Algorithm 1: Generic WFC algorithm

(1) The algorithm requires a single input, which makes it faster than machine learning approaches that require substantial training data as well as training time.
(2) The output can often be generated in milliseconds, giving users a quick feedback.
(3) The iterative nature of the algorithm allows for interception or manual pause at any stage.

Unfortunately, WFC has also a number of drawbacks, which strongly hinder artists in their creative work. Among them, one can point out:

(1) Notions like ‘collapsing a cell’, ‘entropy-based selection’ and ‘constraint propagation’ are *non-trivial*, and require some understanding to grasp why some output is as it is.
(2) WFC originally *lacks an undo* facility, which in turn excludes a trial and error approach, frequently taken by artists. Likewise, WFC is *unable to backtrack*, a much-needed option for restarting, for example, after a conflict occurs, or some of the output is not conforming to the user’s design goal.
(3) Propagation is *not always logical*: whilst the concept is easy to understand, it all happens behind the scenes, which may be often confusing or startling.
(4) Tile selection is *not easily controllable*, because after picking a cell to collapse, the tile choice is based on fixed weights, extracted from the algorithm input.

2.2 WFC variants and extensions

Karth and Smith [7] make a good analysis of the WFC algorithm, placing it in continuity with previous methods: “Both traditional texture synthesis and Markov chain approaches are primarily data-driven and thus accessible to non-programmers.” One can therefore wonder: if these two techniques were made accessible to non-programmers, how could that be done with regard to WFC?

The same authors also discussed how to make the algorithm faster, more efficient and complete [6]. Regarding completeness, one of the answers is the inclusion of backtracking, as it helps cover all possible arrangements, eventually leading to find a solution, if it exists.

Kim and Kang further explore the possibilities of the algorithm by extending it into a graph-based domain, moving away from the simple grid layout [8]. The graph-based domain allows for a broader definition of combination rules, and ultimately to the generation of a much larger variety of content (from maps to Sudoku levels). However, this comes at the cost of a much less intuitive interaction.

To address more flexibility in input usage, Sandhu et al. [13] propose dynamically adjusting tile weights during algorithm execution, which has the potential to lead to more satisfactory output, at least if properly controllable. Their work has partly inspired some of our tile manipulation techniques, described in Section 6.

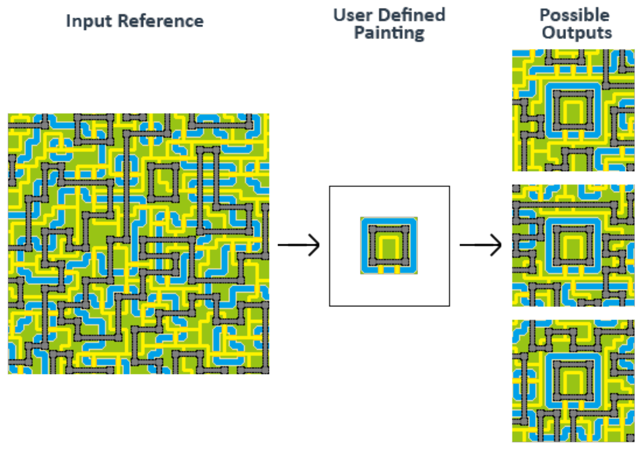


Figure 1: Example of manual drawing before running the WFC algorithm. ***(Input image: Castle [4])***

Cheng et al. [3] implemented three mechanisms into WFC: global constraints, multi-layer generation and distance constraints, to provide a base for non-local constraints, inspired on work by Sandhu et al. [13]. These extensions were added for improved designer control, better playability, and increased similarity to human-designed levels.

Newgas created a library and tool, *Tessera* [11], for constraint-based procedural generation. It contains a ‘painting tool’ for drawing tile adjacencies (rather than the algorithm extracting them from the input), and a number of extensions such as working with pre-placed tiles, tiles spanning multiple cells, more grid types and path graphing constraints. Through these additions, they aim to improve algorithm configurability. In addition, he has made available his WFC C# Library [22], extending the original algorithm with several additional features, including backtracking and non-grid layouts. Our *miWFC* prototype builds upon this library.

Two games developed by Oskar Stålberg [19] are good examples of online PCG through an ad-hoc version of WFC: *Bad North* [20] and *Townscaper* [21]. Bad North customizes the basic WFC algorithm with an extra constraint, to make sure that agents in the game are able to have a navigable path across the levels.

3 MIXED-INITIATIVE CONTRIBUTION

As mentioned before, most creative professionals are not served by the lack of control offered by current WFC algorithm variants. Our work aims at converting WFC into a powerful mixed-initiative creativity tool that makes it much more accessible and usable to a wide, non-technical public. In particular, we address the algorithm drawbacks pointed out above, and propose solutions to them, so that WFC can effectively support artists in freely expressing their intent, and exploring the design space. Naturally, this should be done without compromising the algorithm strengths.

A typical use case might consist of an artist initiating the output by manually creating a specific area of interest at a desired location (something the algorithm would typically not produce on its own), and then utilize the algorithm to fill in the remaining area (see Figure 1). Such a facility is a good example of assisting the user in steering the output in the desired direction. Allowing such manual control at any stage promotes the kind of designer empowerment that drives our research contribution.

4 HISTORY NAVIGATION

In Algorithm 1, selecting a cell (step 3), collapsing it to a given tile (step 4), and the subsequent propagation (step 5), are deterministic processes: these steps always result in the same state, and are therefore revertible. Additionally, the iterative nature of WFC provides a clear and discrete measure for controlled stepping through the process, both in its basic forward (generative) direction and in its backward (undo) direction, when backtracking. By introducing backtracking and manual stepping in the WFC algorithm, and by conveniently bringing these features to an intuitive interface, we allow for more and finer control over the generative process. Because these features relate to the timeline dimension of the algorithm, we call them *History navigation.*

History navigation provides (i) controls to perform an arbitrary number of steps (i.e., algorithm iterations) back and forth, (ii) the ability to 'save' the current progress as a marker to revert to, and (iii) a timeline to indicate progress and visualize these saved markers. These elements work together to address and solve the absence of an undo facility in WFC (Section 2.1). As a result, the algorithm's progress becomes more visible and its control, more accurate and intuitive.

Prototype implementation

Figure 2 shows the history navigation interface of our *miWFC* prototype, featuring a variety of controls, including a play/pause button to control the algorithm, forward/rewind buttons to perform a set amount of steps and, underneath, two progress control buttons to 'save' (place a marker) and 'load' (revert to the previous marker). In addition, two sliders allow you to set (i) the amount of steps to perform before refreshing the display, and (ii) the wait time between steps, effective controlling the speed of the animation. Below the output, a timeline is provided, roughly indicating the fraction of output already generated, by means of a green marker. Above the timeline, the blue markers indicate the 'save' points set so far.

5 DIRECT MANIPULATION

By design, the original WFC algorithm is fully autonomous, choosing the next cell to continue progression through a 'lowest-entropy' heuristic. This boils down to picking the cell with the least amount of potential states, with the goal of minimizing the chances of causing conflicts. The algorithm then chooses, from among the tiles still allowed for that cell, which tile to collapse it to. For this, it uses the tile weights, once computed from the input.

Usually, the chosen cell is found in the vicinity of other collapsed cells, hence the 'flood propagation' appearance of a WFC animation. However, there is nothing preventing us from manually indicating *where* (i.e., at which cell) we would like to proceed nor, for that matter, *how* we wish to 'collapse' there (i.e., to which of its allowed tiles).

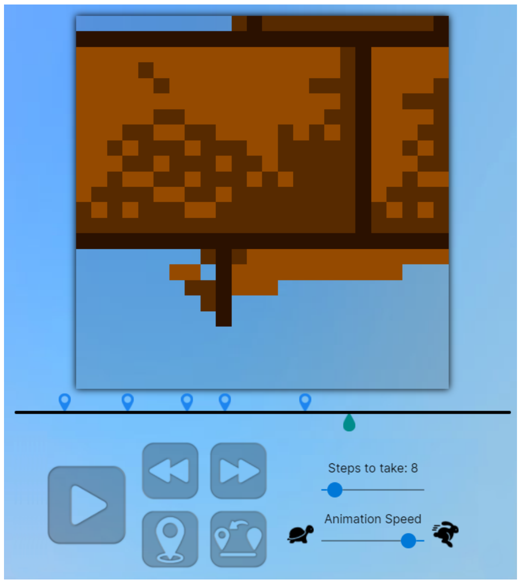


Figure 2: The history navigation interface of our *miWFC* prototype. *(Input image: 3Bricks [4])*

We designed two interactive methods to directly manipulate this spatial progression: (1) a *pencil tool* to collapse a given cell to a desired tile, and (2) a *brush tool* to select a desired area of the output, which is meant to be either fully reset (i.e., cleared up) or preserved from a total output reset.

Both methods are meant to operate directly on the output cells. Therefore, it is possible, and very convenient, to provide a visual insight that previews the outcome of their application. This is possible because the WFC algorithm always keeps track of all allowed states for each cell. Although this information is normally hidden, one can display it real-time, on demand, for example when hovering over a cell. Visualizing all potential tiles for a cell is a very useful insight, and it can even help understand the algorithm's mechanism of tile selection.

5.1 Pencil tool: direct manual collapsing

Manual cell selection for collapsing allows one to steer the output in the desired direction, as illustrated in Figure 1. The pencil tool materializes this idea: by 'painting' with a given tile on a chosen cell, it overrules, in Algorithm 1, the automated picking of a cell (step 3) and of a tile for it (step 4). Furthermore, whilst hovering over the output, it allows for an overlaid preview of the propagated consequences of that collapsing. This can prevent subsequent surprise or misunderstanding on why other cells collapse as a consequence of the user's 'paint' action.

Figure 3 gives an example of the preview representation when hovering: it shows what will additionally collapse, and where, if the cell being hovered were to collapse to yellow.

5.2 Brush tool: direct manual un-collapsing

The second method provides the ability to brush over the output, selecting a desired area to be either reset (i.e., fully 'un-collapse' its cells) or preserved from global resetting. In this way, the user can more accurately choose what to keep from a given output situation. For example, in Figure 4, one can clear up a few small areas, e.g., for further manual tweaking (top), or simply clear up the whole output except the desired area selected (bottom). In either case, brushed masks just indicate the user intent: the actual reset should only take place upon explicit application of the masks.

Indicating an area to be reset does not necessarily guarantee that all its cells will end up fully reset. That happens, in particular, to reset cells neighbouring cells that are to be preserved, because, after applying the mask, re-propagation may collapse (or at least constrain) some of them. Figure 4 illustrates this for both brushing uses.

5.3 User-defined templates

In addition to the Pencil and Brush tools, we designed a third method for direct manipulation of the output. It consists of the definition of tile templates from a region of the output, and their subsequent (re)use (possibly multiple, by 'stamping' that template at desired places in the output. The use of templates allows for improved workflow on larger outputs, as well as for fast creation of output with a specific region of interest. Where in Figure 1 the castle was drawn onto the output, the user could now select this castle (including moat), save it as a template and stamp it onto the output wherever desired, even providing this ability across distinct output generations. Another example of template use is given in Figure 5.

With these direct manipulation features, the designer can especially concentrate on areas of interest of the output, with more control to manually tweak local details while leaving the rest to the algorithm.

Prototype implementation

In the *miWFC* prototype, direct manipulation was implemented using a separate panel; see Figure 6. On the left hand, all tools can be selected: the pencil tool (shown in red), the brush tool, and below those, two user-defined template tools, one for their creation and one for their placement. The pencil tool is accompanied by a drop-down menu to select with which tile to paint. When painting,

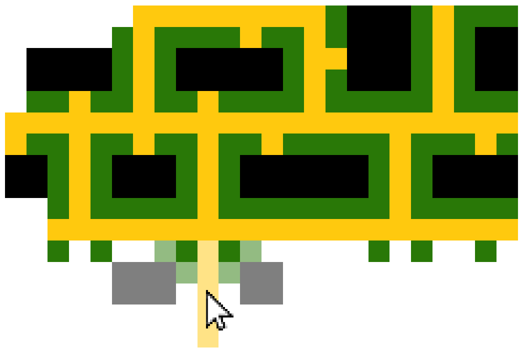


Figure 3: Semi-transparent overlay previewing the results of collapsing to a yellow tile at the cursor. ***(Input image: Village [4])***

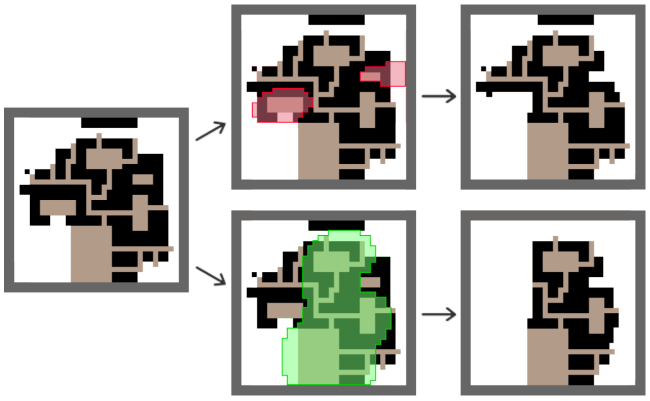


Figure 4: Manually 'un-collapsing' cells: (top) using the brush to reset areas; (bottom) using the brush to lock and preserve an area. ***(Input image: LessRooms [4])***

all tiles available at the hovered cell are shown on the right hand, as shown in Figure 6, with the tile currently selected for painting highlighted in green for clarity. In addition, the panel also provides the ability to place (and revert to) markers, further facilitating quick iterations and experiments.

The brush tool is accompanied by a button to apply the mask(s) to the output, one to reset the mask, and a slider for the brush size (Figure 7a).

The template creation tool basically works as a one-cell-wide brush tool, to select the desired template cells. The template placement tool has a drop-down to select the desired template, one button for deleting the selected template, and one for rotating it prior to placement (Figure 7b).

6 TILE MANIPULATION

Among other tasks, the initialization in step 1 of the WFC Algorithm 1, identifies all tiles in the input, and calculates their weights, according to the relative frequency they occur there. Furthermore, depending on the desired settings, identified tiles may also be considered under rotational or mirroring transformations. Although this may not always be desirable, it is often useful for certain tiles representing, for example, a road corner or a chair, as their multiple orientation might be desirable.

For each cell in the output, the WFC Algorithm 1 keeps track of which tiles are still available. At each iteration, the tile selection mechanism (step 4) selects one tile at random, using the tile weights

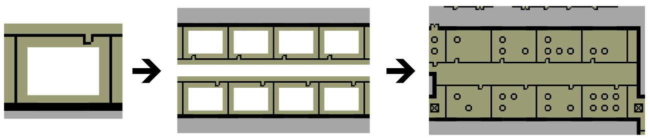


Figure 5: The template (left), output after placing the template multiple times (middle), and the final output after the algorithm filled the blanks (right). ***(Input image: FloorPlan [4])***

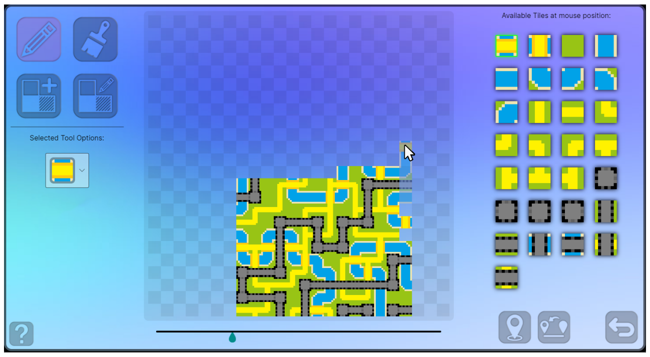


Figure 6: The direct manipulation panel of the *miWFC* prototype. *(Input image: 3Bricks [4])*

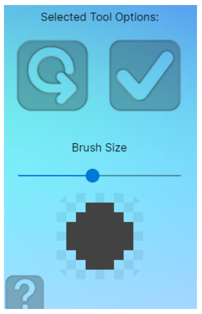
calculated at initialization, to regulate the chance of being selected: the more a tile occurs in the input, the higher its weight, and the higher the chance it will appear in the output.

6.1 Weight manipulation

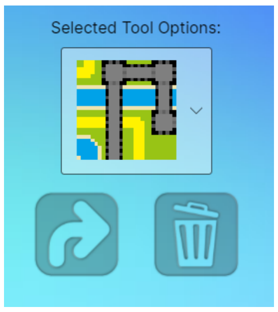
Tile selection is, ultimately, determined by weights and, as long as these are kept fixed, the output will mostly resemble the input. However, by providing the ability to manipulate those weights, one can enable a much larger variety of output, still inspired in the input, but exhibiting very disparate ratios among their tile occurrences.

The fact that weights are simple numeric values, provides users a simple and intuitive control over the relative tile frequencies desired in the output, a larger value making the tile appear more frequently in the output. This can have a significant impact on the algorithm's output, as illustrated in the example of Figure 8, which uses five different tiles: Crossed, Straight, Empty, T-junction and Corner.

However, it might also happen that a user wishes tile weights to dynamically vary over the output space. This boils down to indicating which weight value to use for each tile, at each cell, during step 4 of Algorithm 1. A convenient way of doing this is to graphically 'brush a heatmap' over the output, expressing how these weight values should spatially vary. This allows indicating, for example, a smooth gradient for a given tile weight (see Figure



(a) Brushing tools



(b) Template placement tools

Figure 7: Direct manipulation tool sub-menus.

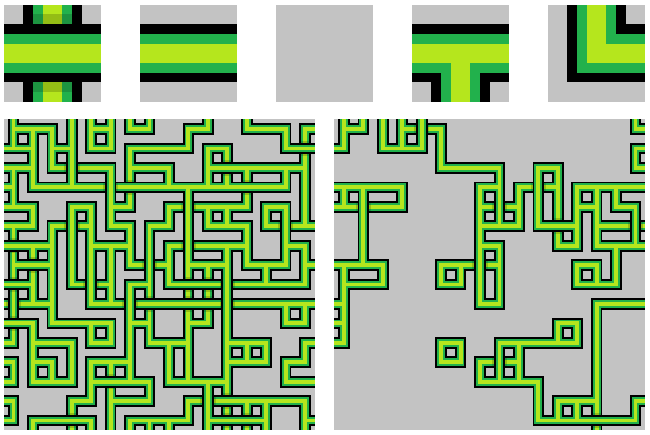
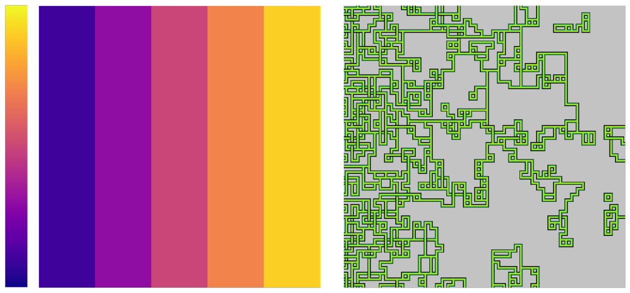
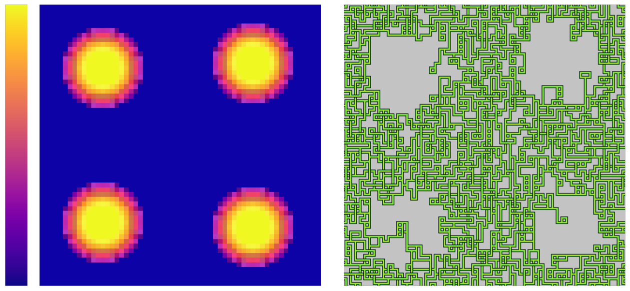


Figure 8: Weight manipulation for the Knots tile set (top) [4]. Example outputs for (left) equal weights for each tile, and (right) increased weight for the empty tile.



(a) The empty tile gets gradually more used, from left to right



(b) The empty tile dominates in the desired spots

Figure 9: Spatially variable weight manipulation by inputting different heatmaps. *(Input image: Knots [4])*

9(a)), or some zones of exceptional high frequency for it (see Figure 9(b)).

6.2 Transformation manipulation

Toggling tile transformations, whether rotational or mirroring, allows for additional control over the appearance of the output. There is a maximum of eight such transformations possible for a single tile.

However, not every tile is affected by every type of transformation, so often some tiles will yield fewer variants. For example, the Empty tile in Figure 8 is affected by none of these transformations, while the T-junction and Straight tiles are only invariant to mirroring.

While not always as obvious as the tile weight manipulation (in Figure 8), toggling these transformations per tile can also have a considerable impact on the output, as illustrated in Figure 10. On the left hand, the Straight tile has rotations disabled, so the output can exhibit no long vertical pipes; on the right hand, instead, it is the Corner tile that has all transformations disabled: as a result, only one corner tile is available instead of all four, effectively lowering its ratio relative to the other tiles.

6.3 Pattern exclusion

In the original WFC algorithm, all tiles initially extracted from the input image are unconditionally taken into account. Yet, allowing the user to exclude individual tiles from the available pool can largely expand the expressive power of WFC, even though it means that the output thus generated will not fully comply to the initial ‘input specifications’. This control allows users to perform more creative experiments, based on the same input image.

Experience tells that it is hard to create a good and clean input for WFC in one go. Therefore, iteratively excluding a tile and assessing how the output looks like is a very effective and powerful method of fine-tuning a good input texture. By definition, the excluded tile(s) will not be found in the output. So this output can be saved, as a customized or improved variant, for further reuse as a future input for the algorithm.

Prototype implementation

We implemented these features in our *miWFC* prototype through an interactive panel, in which a number of settings can be found for each tile identified in the input, see Figure 11.

For each tile, its large bold weight is depicted on the bottom left. This value can be tweaked using the plus and minus to the right of the tile. Through modifier keys, larger increments can be used (Ctrl: ±10, Shift: ±50). Finally, for each tile, two buttons allow toggling the inclusion of tile rotations (top left) and of tile flipping (bottom left) transformations. These buttons are not available for tiles unaffected by each type of transformation. When transformations are turned

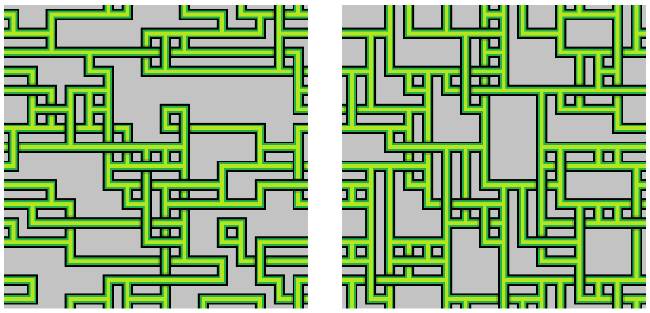


Figure 10: Transformation manipulation for the Knots tile set [4]. Example outputs for disallowing transformations (left) on the Straight tile, and (right) on the Corner tile.

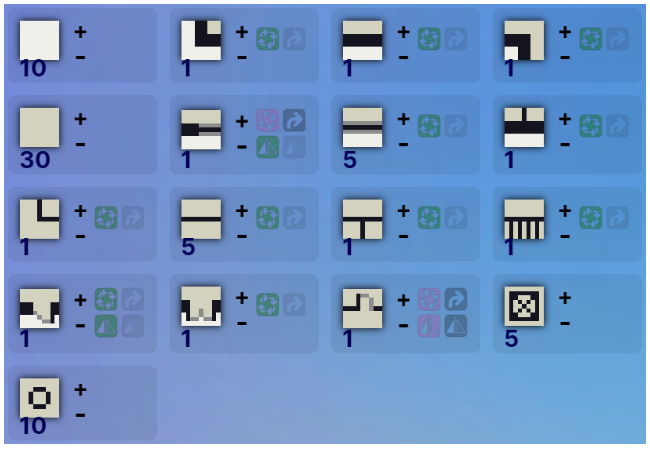


Figure 11: Tile manipulation panel of the *miWFC* prototype. *(Input image: FloorPlan [4])*

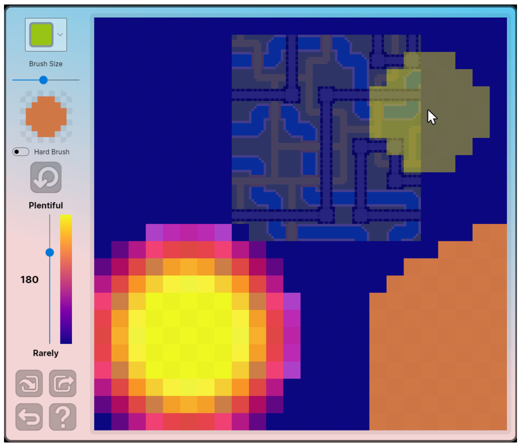


Figure 12: Spatially variable weights editor panel of the *miWFC* prototype. *(Input image: Castle [4])*

off, the grey buttons to the right help set the only position allowed for the tile in the output.

Editing spatially variable tile weights can be done in a separate panel (Figure 12). After choosing the desired tile, the user can freely ‘brush’ weight values as an overlay on the current output. Other options include: setting the desired weight value from a colour-coded scale, setting the brush radius, toggling between soft- and hard-edged brush, as well as exporting/importing already made heatmaps.

7 EVALUATION

So far, we have evaluated the intuitiveness of our approach, implemented in the *miWFC* prototype system. For this, we approached artists and designers from game development studios and online creative platforms. Participants were asked a set of tasks over the course of three user tests. These tests were completed by a total of 45 user study participants, and provided predominantly positive feedback. For verification, participants uploaded their created output, accompanied by a caption motivating their design.

7.1 History navigation evaluation

Our first evaluation topic focused on the navigational controls of *miWFC*, shown in the panel of Figure 2. We asked participants to *"Create a part of a plausible city map, by controlling the steps of the generator (hence and forth) until you are satisfied with the result."*. The majority rated the *miWFC* history navigation features quite positively (Figure 13 (left)). Participants were particularly enthusiast about the use of history navigation markers on the timeline (shown in Figure 2), and gave it a quite positive grade (Figure 13 (right)).

A final 'sandbox' task was included to optionally explore any *miWFC* features without constraints. Albeit optional, only a single participant did not take it, a good sign of the interest sparked among users. In general, the *miWFC* prototype was very well received by the participants, who gave valuable feedback, and expressed interest in participating in subsequent test sessions.

7.2 Direct manipulation evaluation

This evaluation was centred around the direct manipulation methods of *miWFC*, shown in the panel of Figure 6, which are most notably based on the painting and brushing mechanisms. To encourage exploration and creativity, participants were given large freedom. The tasks regarded the creation of a city layout using direct manipulation (given input image ColoredCity [4]). Overall, these tasks yielded mostly positive grades (Figure 14).

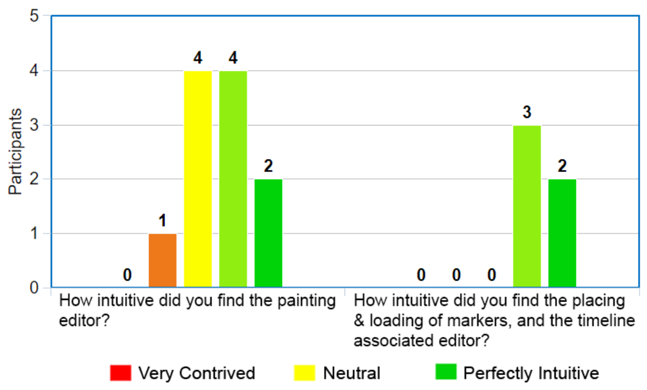


Figure 13: Scores on history navigation

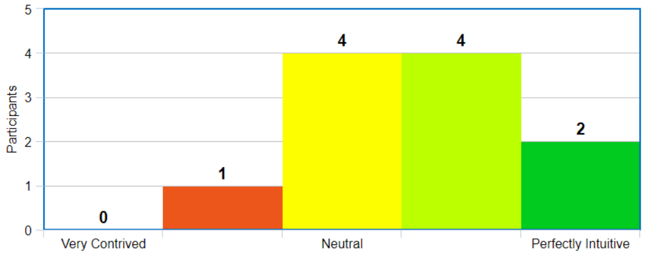


Figure 14: How intuitive did you find the painting editor?

7.3 Tile manipulation evaluation

Our third evaluation focused on four distinct types of tile manipulation provided by *miWFC*. The tasks regarded altering tile weights both statically and dynamically (Figure 15 (left)), tile exclusion (Figure 15 (right)) and tile transformation manipulation (Figure 16). The effect of tweaking tile weights was quickly apparent to all participants, who went on to experiment with their fine control. Overall, albeit intricate, tile manipulation tools were considered intuitive, with mostly positive results.

8 CONCLUSION

Wave Function Collapse (WFC) is a powerful and promising PCG algorithm, but it lacks interactive features and intuitive control, indispensable for the work of most creative professionals. In this paper, we presented a novel approach that converts WFC into a powerful mixed-initiative PCG method.

This approach, properly called *miWFC*, provides abundant interactive features and intuitive control mechanisms for artists and designers, to effectively assist them in their creative work. Among other features, *history navigation* explores the iterative nature of the WFC algorithm, and the animation of its 'flood progression', to promote trial-and-error experimentation over a timeline, by means of an intuitive undo mechanism. *Direct output manipulation* overrules, whenever desired, the automatic WFC cell and tile selection steps, replacing them by the explicit user selection, and allowing

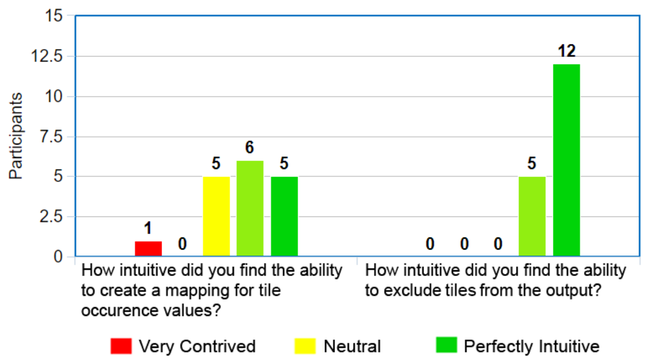


Figure 15: Scores on dynamic weights & pattern exclusion

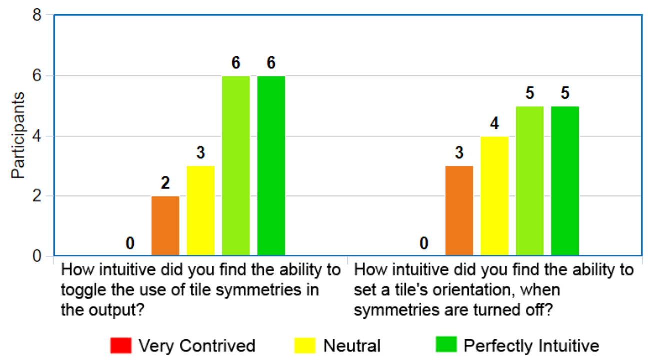


Figure 16: Scores on tile transformation manipulation

for an effective steering of the content generation towards the actual design intent. *Tile manipulation* permits overruling both the weights and the orientations of the tiles identified on the algorithm input. In addition, tile weights can be interactively made to vary throughout the output space, further supporting the expression of a designer's vision. This, in turn, provides users with a fine-grained method for tweaking the global appearance of the output, steering it away from the input whenever desired, in a controlled manner. All these features were implemented in our prototype system *miWFC*, and validated through user tests performed among the end user community, who considered them rather intuitive and useful to express their creative intent.

We believe various other WFC interactive features can be designed, aimed at supporting the creative expression of artists and designers, and empowering them to configure and explore a given generative space. Like in the present work, most likely, an important challenge ahead lies in the design of interactive techniques to encapsulate the inherent complexity of other current WFC extensions. Among the latter, one can include the 'multiple input images' introduced by Karth and Smith [6], extracting patterns from the fusion of "positive" input(s), and excluding undesired patterns from "negative" input(s). Other challenging WFC extensions that would deserve attention are the (re)definition of tile adjacency constraints, the effective definition and use of 3D tiles [11, 1], and the WFC generalization to graph-based domains [8].

The *miWFC* prototype developed in this research project can be downloaded at its repository[1], for anyone to experiment with. As an open-source project, its source code is available under the usual conditions, for others to see, modify or perform further developments.

REFERENCES

[1] Levi van Aanholt and Rafael Bidarra. 2020. Declarative procedural generation of architecture with semantic architectural profiles. In *Proceedings of CoG 2020 - IEEE Conference on Games*. IEEE.

[2] Alberto Alvarez, Steve Dahlskog, Jose Font, Johan Holmberg, Chelsi Nolasco, and Axel Österman. 2018. Fostering creativity in the mixed-initiative evolutionary dungeon designer. In *Proceedings of the 13th International Conference on the Foundations of Digital Games* (FDG '18) Article 50. Association for Computing Machinery, Malmö, Sweden, 8 pages. ISBN: 9781450365710. DOI: 10.1145/3235765.3235815.

[3] Darui Cheng, Honglei Han, and Guangzheng Fei. 2020. Automatic generation of game levels based on controllable wave function collapse algorithm. In *International Conference on Entertainment Computing*. Springer, 37–50.

[4] [SW] Maxim Gumin, Wave Function Collapse Algorithm version 1.0, Sept. 2016. URL: https://github.com/mxgmn/WaveFunctionCollapse.

[5] Daniël Karavolos, Anders Bouwer, and Rafael Bidarra. 2015. Mixed-initiative design of game levels: integrating mission and space into level generation. In *Proceedings of 10th the International Conference on the Foundations of Digital Games*.

[6] Isaac Karth and Adam Smith. 2021. WaveFunctionCollapse: content generation via constraint solving and machine learning. *IEEE Transactions on Games*, PP, (May 2021), 1–1. DOI: 10.1109/TG.2021.3076368.

[7] 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* (FDG '17) Article 68. Association for Computing Machinery, Hyannis, Massachusetts, 10 pages. ISBN: 9781450353199. DOI: 10.1145/3102071.3110566.

[8] 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 (Aug. 2019), 1–4. DOI: 10.1109/CIG.2019.8848019.

[9] Antonios Liapis, Georgios N. Yannakakis, and Julian Togelius. 2013. Sentient sketchbook: computer-aided game level authoring. In *Proceedings of the 8th Conference on the Foundations of Digital Games*, 213–220.

[1]https://github.com/ThijmenL98/miWFC

[10] Paul Merrell and Dinesh Manocha. 2010. Model synthesis: a general procedural modeling algorithm. *IEEE Transactions on Visualization and Computer Graphics*, 17, 6, 715–728.

[11] Adam Newgas. 2021. Tessera: a practical system for extended wavefunctioncollapse. In *The 16th International Conference on the Foundations of Digital Games (FDG) 2021*, 1–7.

[12] Mijael R Bueno Perez, Elmar Eisemann, and Rafael Bidarra. 2021. A synset-based recommender method for mixed-initiative narrative world creation. In *Interactive Storytelling – Proceedings of ICIDS 2021* (LNCS 13138). Springer, Cham, 13–28.

[13] 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* (FDG '19) Article 17. Association for Computing Machinery, San Luis Obispo, California, USA, 9 pages. ISBN: 9781450372176. DOI: 10.1145/3337722.3337752.

[14] Ruben Smelik, Tim Tutenel, Klaas Jan De Kraker, and Rafael Bidarra. 2010. Integrating procedural generation and manual editing of virtual worlds. In *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*, 1–8.

[15] Ruben M Smelik, Tim Tutenel, Rafael Bidarra, and Bedrich Benes. 2014. A survey on procedural modelling for virtual worlds. *Computer Graphics Forum*, 33, 6, 31–50.

[16] Ruben M Smelik, Tim Tutenel, Klaas Jan de Kraker, and Rafael Bidarra. 2011. A declarative approach to procedural modeling of virtual worlds. *Computers & Graphics*, 35, 2, 352–363.

[17] Ruben M Smelik, Tim Tutenel, Klaas Jan de Kraker, and Rafael Bidarra. 2010. Interactive creation of virtual worlds using procedural sketching. In *Eurographics (short papers)*, 29–32.

[18] Gillian Smith, Jim Whitehead, and Michael Mateas. 2010. Tanagra: a mixed-initiative level design tool. In *Proceedings of the Fifth International Conference on the Foundations of Digital Games*. ACM, 209–216. DOI: 10.1145/1822348.1822376.

[19] Oskar Stålberg. 2022. Art by Oskar Stalberg. (2022). https://oskarstalberg.tumblr.com/.

[20] Oskar Stålberg. 2022. Bad North - A Minimalistic, Real-Time Tactics Roguelite with Vikings. (2022). https://www.badnorth.com/.

[21] Oskar Stålberg. 2022. Townscaper. (2022). https://oskarstalberg.com/Townscaper/.

[22] Boris the Brave. 2022. DeBroglie Documentation. (2022). https://boristhebrave.github.io/DeBroglie/index.html.

[23] Sean P. Walton, Alma As-Aad Mohammad Rahat, and James Stovold. 2020. Mixed-initiative procedural content generation using level design patterns and interactive evolutionary optimisation. *ArXiv*, abs/2005.07478.