Hierarchical Semantic Wave Function Collapse

Alaka, Shaad; Bidarra, Rafael

DOI
10.1145/3582437.3587209

Publication date
2023

Document Version
Final published version

Published in
Proceedings of the 18th International Conference on the Foundations of Digital Games, FDG 2023

Hierarchical Semantic Wave Function Collapse

Shaad Alaka
s.alaka@student.tudelft.nl
Delft University of Technology
Delft, The Netherlands

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

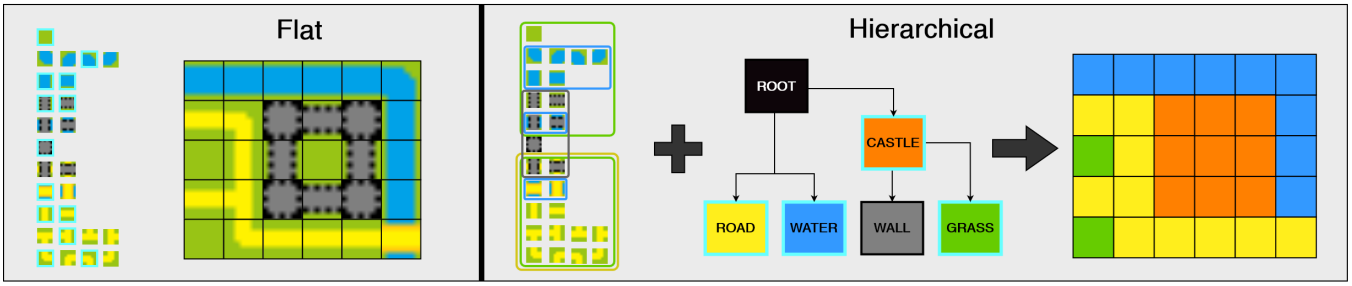

Figure 1: With the flat tile-set of conventional WFC, its output is always at detail level; with a hierarchically structured tile-set, the output can be more conveniently configured and edited at meta-tile level (castle tile-set [1]).

ABSTRACT

There are few proposals to improve the interactivity and control of wave function collapse (WFC) in a mixed-initiative setting. Moreover, most WFC algorithm variants operate on an simple, unstructured set of tiles. This limitation on the level of control provided to designers hampers their creative work in various ways. We propose Hierarchical Semantic WFC, a generalized approach to WFC that organizes its tile-set into a hierarchy akin to a taxonomy induced by the relation 'consists-of'. In such a hierarchical structure, abstract tiles (i.e. non-leaf nodes) can represent the first sketchy intentions of a designer (e.g. forest, urban, desert,...) This allows a designer to interactively collapse a given area into abstract tiles, while subsequently, (and repeatedly, if desired) WFC can *resolve* each area into a variety of particular instances, by further collapsing it into (a valid combination of) its children tiles (whether leaves or not). We identify how this subtle tile-set change affects the whole WFC algorithm, describe a number of novel exploratory and interactive functions that this enables, and showcase these with a variety of examples generated with our prototype implementation. We conclude that these new mixed-initiative content generation methods can considerably reduce design iteration times and improve the assistance given to designers in expressing their creative intent.

KEYWORDS

procedural content generation, wave function collapse, mixed-initiative, object semantics

FDG 2023, April 12–14, 2023, Lisbon, Portugal

ACM ISBN 978-1-4503-9855-8/23/04.
https://doi.org/10.1145/3582437.3587209

ACM Reference Format:
Shaad Alaka and Rafael Bidarra. 2023. Hierarchical Semantic Wave Function Collapse. In *Foundations of Digital Games 2023 (FDG 2023), April 12–14, 2023, Lisbon, Portugal.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3582437.3587209

1 INTRODUCTION

Since its inception in 2016, WFC has become a popular choice for procedural generation of textures, objects and other grid-like structures [1, 4]. Several variants have since then been proposed, extending its functionality and convenience. In particular, proposals to improve the interactivity and control of a WFC-based generator are promising and much welcome. Among them, interactive generators that offer mixed-initiative editing possibilities can be very convenient, effectively supporting artists while amplifying their creative freedom [14]. One such approach was demonstrated by the miWFC prototype[1] [8].

To the best of our knowledge, however, all such WFC algorithm variants operate on a flat set of tile choices per cell. This implies that each tile contains all the concrete and final detail about what it exactly represents. This flat structure implicitly blurs much of the semantics intentionally represented in those tiles by designers, who typically include in them multiple semantic "hints"; for example, the tile-set of Figure 1 includes tiles such as a horizontal straight road that crosses water, a road left-turn on grass, a straight wall of a castle on grass, etc.

Directly using such a detailed tile-set in a mixed-initiative context can be overwhelming for artists, as they will have to consciously think about each of these semantic bits of information before they can place any tile. This quickly becomes awkward and rather impractical, as shown on the left of Figure 1, showing (with a cyan border) all the tiles required to create that output.

It is well known that artists prefer to sketch, in order to quickly express the simple idea they have in mind [14, 19]. For example,

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

they would do this by quickly drawing a wall, or brushing a coastline. Ideally, after that, a generative system could assist and fill in with the details of a possible solution. In this way, the general idea gets captured, while the details can be left up to the machine and, possibly, re-worked. However, in current mixed-initiative WFC approaches, designers are unable to freely sketch, refine and re-evaluate their ideas in a step-wise, creative fashion. On the one hand, this is due to the lack of a semantic vocabulary for sketching and refinement with tiles. On the other hand, to modify and re-evaluate some generated output, one should distinguish what was manually placed with some intent, from what was simply procedurally generated [15]. But standard WFC does not keep track of this information.

We propose a solution to these shortcomings by introducing (i) the notion of *meta-tile*, an abstract tile that represents a semantic group of tiles, along with (ii) a graph-like structure that is able to represent the hierarchy among them and the constraints between them.

Together, these elements provide a solid basis for a variety of novel features that empower designers' expressiveness, allowing them to *focus on what they want, rather than on how to make it.* In addition, some of these features significantly reduce the amount of laborious operations during content creation and editing, allowing designers to iterate more rapidly. This is clearly illustrated on the right side of Figure 1, in which four different tiles are enough for the creation of an identical output. Moreover, for the hierarchical case, the artist can focus on their idea, rather than on how each tile fits onto another tile. This core idea purposefully exploits a fundamental property of mixed-initiative WFC: tile placement on the grid is context-sensitive, and setting this context by means of tile layouts allows one to enforce desired outcomes.

2 RELATED WORK

In 2016, Maxim Gumin unleashed the WFC algorithm, publishing a repository containing his initial implementation [1]. Since then, WFC has had a profound impact on technical artists and game developers, getting adopted, adapted and used in commercially published and upcoming projects (Caves of Qud, Townscaper, Matrix Awakens). The repository has become a hub of anything related to WFC, linking to research, derived works, alternative implementations, etc. [3, 4, 7, 9, 11]. Years before, Merrel had published the conceptually identical Model Synthesis algorithm, though it did not catch on as much as WFC did, possibly due to its lower accessibility, main 3D focus and computing requirements at the time [10].

For completeness, we provide here a generic description of how WFC operates, to set the context for this work. WFC accepts as input a set of tiles, adjacency constraints between them, and produces as output a connected graph (in this case, assume e.g. a grid consisting of cells). Tiles may be placed into cells, and each cell keeps track of which tiles is it still allowed to contain, given the constraints induced by its neighbouring cells. Initially, every cell of this grid is empty and can potentially contain any tile (hence the analogy with 'wave function'). As the algorithm progresses (see Algorithm 1), the following three steps are iterated:

Algorithm 1 Basic WFC algorithm

```
initialize algorithm (building tile and constraint tables)
repeat
    Choose next cell to collapse
    Choose which tile to collapse it into
    Collapse and propagate constraints
until Every grid cell is collapsed or a conflict occurred
```

(1) A cell is chosen to be 'collapsed': for this some heuristic can be used, e.g. the cell allowing the least amount of potential tiles (i.e. lowest entropy);
(2) a tile is chosen to collapse that cell into: for this, one of the tiles is chosen among those allowed for that cell (possibly weighted by its occurrence frequency in the input);
(3) This cell collapse spawns a propagation wave starting at its neighbours, which will eliminate tile choices for the cells it hits according to the adjacency constraints. The wave stops propagating when there are no more changes occurring to the tile choices.

This cycle repeats until either all cells have been collapsed, producing a grid of tiles that satisfies the constraints given as input or, alternatively, when a conflict is reached. The latter happens if a cell ends up without any allowed tile choices after a propagation wave hits it, rendering the current grid instance unsolvable.

As noted elsewhere [8], in this algorithm, both the cell to collapse and the tile to collapse it into could potentially be chosen by a human. This fact provides the basis for any interactive, mixed-initiative WFC editor, in which the user may directly select, on the output grid, which cell(s) to collapse into some selected legit tile. In this setting, WFC can automatically validate that action, and further propagate any changes across the output grid.

Being able to more accurately capture designer intent is desirable because it will speed up the overall design process. This has been done so far in a variety of ways. In one such example, an interactive world editor [14] was augmented to capture design semantics, introducing semantic constraints such as e.g. preserving line of sight between world entities [13]. In another project, the designer can input a rough design specification that guides a procedural generator for room layouts [12].

Adding mixed-initiative interactivity to WFC to make the generation process more spatially controllable has been proposed with miWFC, an interactive editor that allows you to place/overwrite tiles with a brush, create snapshots, regenerate marked parts, and spatially alter the tile probabilities [8]. In addition, other proposals of WFC-based interactive editors have been made, various links are available on the WFC Github repository [1]. Other types of mixed-initiative interactivity have been proposed that do not involve spatial control. Karth and Smith [5], for example, propose to allow the designer to intuitively adapt constraints by providing positive and negative examples of tile combinations; to fulfil them, a back-and-forth process progresses towards the generated result. There are also applications of mixed-initiative WFC interactivity within a game. In Townscaper, for example, Oskar Stålberg practically shows the concept of a single meta-tile that is used for painting towns, while WFC takes care of the propagation [17]. All

these approaches demonstrate the flexibility of WFC when it comes to involving humans in the generative process.

There have also been some efforts at incorporating hierarchies in procedural content generation in general [6], e.g. by using a hierarchy of rules [2]. In particular, one approach that comes close to this idea is rule-based layout solving [18], which also uses classes of objects in order to populate areas marked with those class constraints, though it does not allow for a hierarchy that goes deeper than one level. In another example, a 2D game map consisting of tiles is divided into chunks and then clustered, in order to find high-level tiles, which are similar in spirit to the meta-tiles presented in this work, though again only representing a single level of depth [16].

3 METHOD

The core problem is the fact that the basic WFC algorithm is agnostic to semantics when used as backbone for an interactive editor. This problem manifests itself in two distinct ways:

(1) Semantic concepts, both concrete (e.g. 'on grass') and abstract (e.g. 'urban'), get blurred together, resulting in an output of information-dense tiles.
(2) It makes impossible to track the initial semantic intent a designer had when choosing a tile, because this information is in the designer's mind and not stored in the output.

Both problems hinder designers: due to (1), they are required to consider all the semantics associated with a tile before being able to choose it, even though they may only have some specific semantic intent in mind; because of (2), the semantic decisions that a designer ends up making get lost in the output, even though this information would be useful in subsequent editing iterations or other applications.

Therefore, our main goal with augmenting the WFC algorithm is to enable it to distinguish tiles with bunched up semantics and the relevant relations between them, so that they can be reasoned over, stored and used separately.

3.1 Hierarchy of semantic nodes

There are many different ways to represent semantic concepts and how they are related. One such way is in the form of a graph, in which the semantic concepts are represented by nodes, while the edges represent relations between them.

It is neither feasible nor useful to model all possible relations between semantic concepts, in the context of a virtual environment editor. Inherent to WFC, of course, is the "can-be-adjacent-to" relation, which is kept by the algorithm and used for its propagation step.

For our purposes of creating a hierarchy, it will suffice to represent the "consists-of" relation, which is unidirectional by definition. In a real-world setting, recursion does not occur with this relationship, which eliminates cycles in the graph. Finally, we will consider that every node can be reached from one particular node, called the *root node*. The root node represents the state of any fully non-collapsed cell, which can potentially collapse into any other node. Nodes that have no outgoing edges are *leaves*. This structure is very similar to a tree, except that a node x may have multiple parents (i.e. each parent may somehow "consist-of" of x). This is a type of directed acyclic graph (DAG), and we will refer to it as such, as it also admits the use of terminology such as "root", "leaves", "parents" and "children" in the same fashion as a tree; see Figure 2.

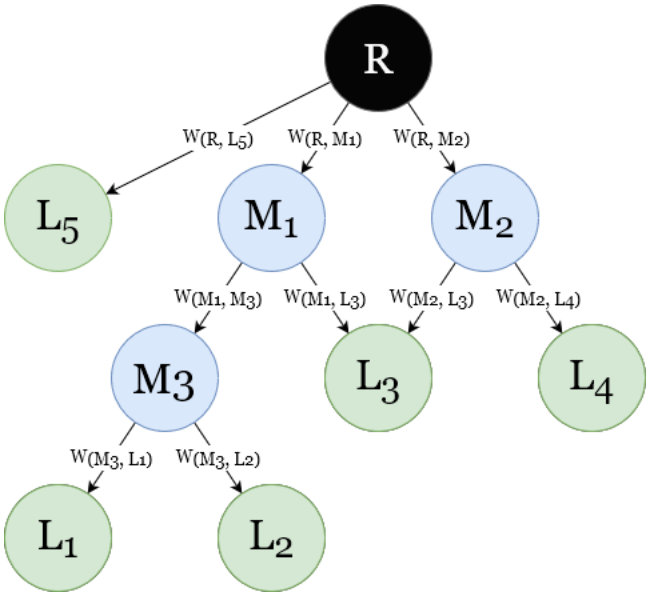

Figure 2: General hierarchical structure of the proposed DAG. Green corresponds to leaf nodes, blue to meta-tile nodes, and black to the root, representing the initial meta-tile on every cell.

These choices were made in order to make the structure resemble a hierarchy; the further a node is from the root, the more detailed and specific it is in terms of semantics, taking on the semantics of all its predecessors. This has benefits for both the designer, due to the correspondence to detail levels, and for the algorithm operation itself, as we will see.

The DAG leaves represent the most detailed concepts; in a way, they correspond to the usual tiles used as input to the standard WFC algorithm, with all the semantics bunched up, which we will refer to as *leaf-tiles*. In contrast, all other nodes are also represented by a new type of tile, which will be called a *meta-tile*. Just like leaf-tiles, meta-tiles may also be involved in constraint definitions, between each other or with leaf-tiles. Finally, as indicated in Figure 2, every DAG edge, between parent node x and child node y, has a weight associated with it, which can be interpreted as "which fraction of x consists of y".

3.2 Algorithm extension

The DAG, with its meta-tiles and weights, allows us to extend the WFC algorithm according to the following:

Initialization and input All cells on the grid are initialized to the root meta-tile of the DAG. The input is defined as a set of single-colored meta-tiles and leaf tiles, a DAG that specifies how the meta-tiles flow into the leaf-tiles, and a set of adjacency constraints that may be specified among/between them.

Intermediate collapse states When collapsing a cell automatically, only the direct children of the DAG node representing the tile that is currently on that cell can be chosen as replacement of that tile. This allows a cell to collapse multiple times, with the nodes in the DAG acting as intermediate collapse states. A cell can no longer collapse further when it reaches a leaf-tile.

Child choice Which child of a meta-tile x gets chosen for replacing it is determined by the weights on the DAG edges from x to its children.

Adjacency constraint inference For a correct propagation, the usual adjacency constraints used by the WFC algorithm are now augmented with the additional constraint information that is inferred from the sub-graphs corresponding to each of the tiles being tested.

Partial propagation and leaf reachability In certain cases, a meta-tile may no longer be allowed to collapse into some of its children. This must be correctly propagated; since all adjacency information eventually trickles down to the leaves, it is sufficient to omit a tile if there exists no path from its node to any leaf.

Collapse paths The collapse path that a cell takes in the DAG is recorded and stored in the output. Such collapse paths can be used to backtrack to the higher level semantics a tile originated from.

3.3 Overlapping model and generality

The novel notion of a tile hierarchy generalizes to the WFC overlapping model, to other types of adjacency models and even to using other types of constraints, as its method has no hard dependencies on either 2D grids or the simple tiled model.

There are, however, some subtleties brought in by the overlapping model, which influence one particular ability: the specification of meta-meta or leaf-meta adjacencies in the input. Because of the overlapping zones of the patterns, these may become ambiguous. Therefore, a choice has to be made for using a fitting tile for that transition, if such a pattern was not already present. While it is possible to use a fixed color for this, it is probably preferable to allow a designer to specify transitional tiles by filling these gaps only for the patterns they want to be involved. Note that this is not required if the appropriate patterns are already present. In that case a meta-tile can immediately act as a convenient wild-card, which allows the creation of patterns that only focus on the features of interest, or that allow for querying certain features on the grid.

Other than that, specializing the notion of semantic hierarchies for the overlapping model also brings exciting opportunities that should be further explored, such as using actual patterns for the meta-tiles instead of single colors, which could be done by tweaking the adjacency constraint inference. Figure 4 shows our proof of concept prototype, using hierarchical semantic WFC with the overlapping model.

4 EDITING FACILITIES

The algorithm extensions introduced in the previous section support several novel design workflows, enabled by a few new editing facilities, which can be made available to designers in an mixed-initiative editor. In this section, we show the usefulness of each of them, by means of examples based on the simple input DAG of Figure 3.

Note that we are addressing two types of designers here, since the newly enabled facilities benefit both groups:

(1) The *input designer*, who defines the tiles and the constraints between them, and how their corresponding nodes are related in the DAG.
(2) The *environment designer*, who creates an environment design specification using the input, and then generates a final output for this.

In what follows, we indicate (in the subsection title, between brackets) which of the two designer groups is addressed with each editing feature.

Furthermore, in order to consistently and clearly refer to the tiles and the constraints among them, we use the symbol <-> to denote an adjacency constraint between two tiles, allowing them to be adjacent, and we use SMALL CAPITALS when referring to some tile. For example, LAND <-> WATER denotes that adjacency is allowed between the meta-tile representing "land", and the meta-tile representing "water". In this paper, we will not handle any uni-directional adjacency constraints, so all presented constraints are assumed to be bi-directional.

4.1 Meta-tile constraints [1]

Since the input now supports constraint definitions between meta-tiles, or between meta-tiles and leaf-tiles, we can let the input designer add additional constraints with very little effort. Examples:

- You can specify that LAND can be adjacent to WALL, which will ensure that any tile in the sub-graph of WALL can have adjacent any tile in the sub-graph of LAND.
- One can say that a HOUSE collapses into WALL and FLOOR, and then specify WALL <-> FLOOR. Building on the above point, this implies that when you place a WALL meta-tile somewhere, it can either collapse into a lone WALL, or get a FLOOR meta-tile attached to it, which would cause it to turn into a HOUSE, since FLOOR must be surrounded by WALL.
- This feature can also be used for transitions; the designer could specify a new meta-tile DENSE-FOREST that has the same children as FOREST, except with a much larger weight on the edge pointing to TREE, and then only specify FOREST <-> DENSE-FOREST, which means that it will always try to surround itself with FOREST first when automatically collapsing.

Without this feature, the input designer would have to tediously define the constraints between all the possible leaf-tiles, while now entire sub-graphs in the DAG can be addressed with one constraint. This also makes it much easier to add e.g. one more sibling tile, as it will simply acquire all constraints from the parent meta-tile. Lastly, meta-tile constraints enable a basic form of semantic transitions, allowing one to morph the distribution of contents gradually over an area.

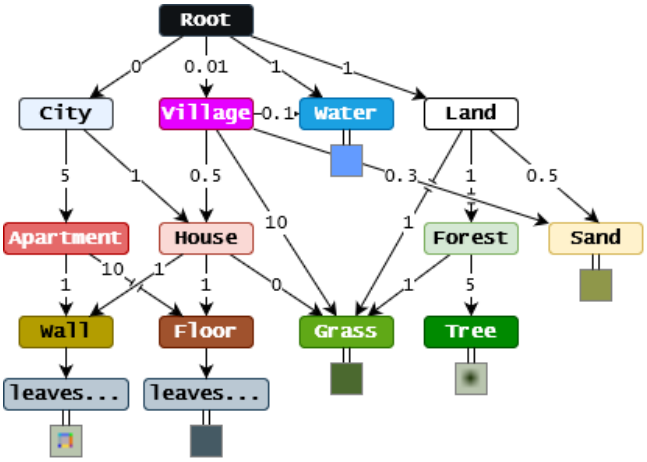

Figure 3: The DAG for the example use case. The nodes are colored according to their meta-tile. The leaf nodes have the leaf-tiles they correspond to attached to them.

4.2 Meta-tile weights [1]

When the designer brushes over de canvas using meta-tiles, the rest of the detailed work can be carried out through automatic WFC. However, the weights that influence which tiles will then be chosen can still be interactively adjusted by the designer within the editor. This allows for control over the distribution/density of tiles. Examples:

- The designer can control, in the DAG, the relative frequency of occurrence for the children of meta-tile HOUSE, by adjusting the weights on their edges. Note that both CITY and VILLAGE "consist of" HOUSE; yet, thanks to the different edge weights, HOUSE can be made more common in VILLAGE than it will be in CITY.
- Likewise, CITY would likely contain more APARTMENT than HOUSE. Although both consist of, say, WALL and FLOOR, HOUSE can have a much higher FLOOR weight, causing WFC to generate larger structures for it.
- The amount of TREE tiles in FOREST can be controlled through the corresponding edge weights. One can design multiple FOREST tiles with different densities. Without this, the designer would not be able to differentiate between semantic groups through the distribution of tiles, or easily tweak the distribution of tiles within some area representing a semantic concept.
- The weights can also be used to completely deactivate a tile from appearing in automated generation, by putting them to zero. This can be used to prevent the generation of some tile that, for instance, is needed under some meta-tile as a filler to facilitate a plausible generation process. For instance, HOUSE allows GRASS to prevent it from running into a contradiction when the shape of an area that was painted with HOUSE is not fully filled with WALL or FLOOR, but in general we don't want grass to generate under HOUSE unless necessary.

4.3 Meta-tile placement [2]

The designer can now collapse a group of cells on the grid to a meta-tile, e.g. with a brush, which allows for a spatial expression of semantics. Moreover, for this the designer can choose a meta-tile at any level of detail, and at any time. Examples:

- A designer wishes to have a village in some location, without wanting to be specific about how the houses look or are distributed, therefore VILLAGE will be placed.
- One of the houses in this VILLAGE area should have a specific shape. The designer can "zoom in" and draw that shape with a HOUSE brush within the previously collapsed VILLAGE area.

Without this, the designer would have to either tediously describe each of the houses or none of them, unable to choose where to be specific. This basic feature clearly expresses the usefulness of meta-tiles in a mixed-initiative context, turning the iterative design process into a declarative semantic specification, rather than focusing on a detailed tile painting.

4.4 Storing the collapse paths [2]

Storing the collapse paths allows tracing back the meta-tiles in the DAG and highlighting or selecting areas that correspond to some specific meta-tile, also allowing for the bulk-execution of certain semantically meaningful operations. Examples:

- The designer can choose to select all cells with tiles that have HOUSE in their collapse path DAG, highlighting all houses (but not any apartments).
- Some editing operations can be performed on such a selection, for example, clearing, as well as regenerating, in order to obtain different houses.

The operation of regenerating (groups of) meta-tiles is impossible without the present algorithm extensions; even if one would be able to select all the generated structures that correspond to e.g. houses, there would be no way to constrain the generation process to generate semantically equivalent structures. With meta-tiles, though, one restricts tile collapsing to remain within the corresponding sub-graph.

4.5 Automated collapsing

Currently WFC is mostly used as a procedural generator in practice, without necessarily involving a human in a mixed-initiative generation process. Still, hierarchical semantic WFC brings benefits even if we limit the scope to automated generation:

- We can now use the chosen tile's depth in the tree as a cell selection metric, which allows for level-by-level collapsing
- The weights on the tree allow for a hierarchical form of density control
- By using a tile's depth as selection metric, we can make adjacency constraints between meta-tiles at the same level more meaningful, allowing for the automatic generation of transitions among them.

5 RESULTS

We implemented the hierarchical semantic WFC method described so far, and made it the backbone of an editor that offers the new editing facilities identified in Section 4. In this section, we briefly

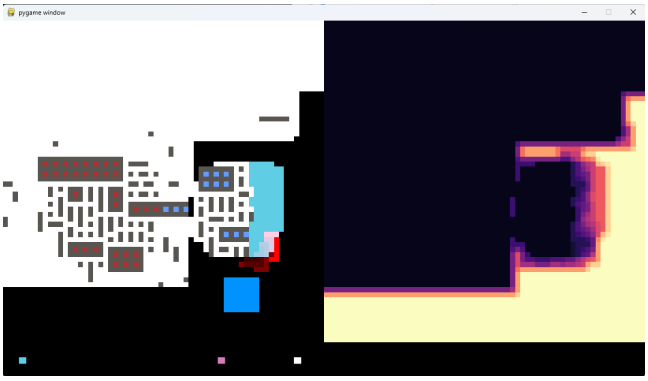

Figure 4: A screenshot of the current pygame-based prototype editor while collapsing meta-tiles into leaf-tiles, combining the overlapping model with hierarchical semantic WFC.

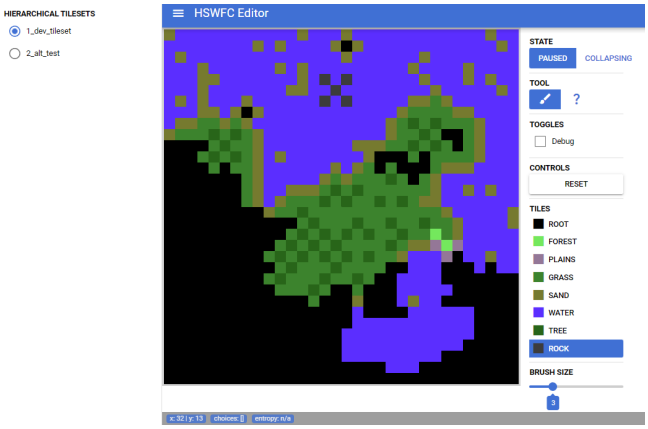

Figure 5: A screenshot of the current web-based prototype editor, paused while it was auto-collapsing.

describe our prototype implementations, and then focus on demonstrating the usefulness of those editing features within a use case, consisting of interactively designing a top-down level for an open world game.

5.1 Prototype implementations

Two prototypes were conceived: one for development and testing[2], and another one that is intended for interactive use by designers[3]. The former one was written in Python, using data structures from `numpy`[4] for fast vectorized operations on matrices, and using the fast deque from `collections`[5] to accelerate the algorithm, and the graphical user interface (GUI) is rendered using `pygame`[6] by means of blitting. This version of the prototype is responsible for the outputs shown throughout this section. It has several debugging features, such as being able to record and visualize propagation waves and show what tiles were disallowed because of it, showing the canvas entropy on a second panel, at the right; see Figure 4. It also has some more advanced editing features, such as the ability to overwrite meta-tiles with their ancestors and (un)propagate this accordingly, which can act as an eraser or re-generation tool depending on how it is used.

The second prototype is much more user friendly in its design from the ground-up, and is intended as testbed for benchmarking the usefulness of the novel editing facilities for designers; see Figure 5. It is built in Quasar[7], and uses a webworker to run the algorithm in a background thread. The algorithm was built in Javascript, using `mathjs`[8] as a convenient substitute for `numpy` and a fast deque implementation[9] for the propagation queue. A demo of a development version is available online for everyone to try out[10]. This demo will be kept up to date with the latest developments.

Both prototypes allow simultaneous editing and automated collapsing, which ensures a fast and responsive user experience while editing the environment. At the moment, both prototypes implement a version of the simple-tiled WFC algorithm on a 2D grid. In addition, the Python prototype also implements the general WFC overlapping model, as shown in Figure 4. This was mostly done to confirm that the notion of semantic tile hierarchies generalizes to other forms of WFC as well.

5.2 Use case

Level and environment design is a crucial task in video game development. Various game development tools exist with first-class support for the incremental creation of such environments. In many games (e.g. Caves of Qud, Legend of Zelda, Rimworld, Minecraft) these environments are laid out on a grid of tiles or voxels, and this feature very well suits the output of WFC.

As a use case, we consider here a map editor for a top-down 2D game, where the designer wants to rapidly sketch out a world with the key elements that should be present, and then recur to hierarchical semantic WFC to detail and fill the gaps iteratively. First, we evaluate the generation of a map from a sketched input. For this, we use the tile-set and DAG of Figure 3.

In Figure 6 the west-side adjacency matrix for this particular tile-set is shown (you need one for each cardinal direction). Note that LAND <-> WALL LEAVES was specified, which according to the DAG will allow any of GRASS, TREE, FOREST and SAND to be adjacent to the children of WALL. In fact, if the designer wishes to add more tiles and makes sure that LAND will be the ancestor of this tile, it will automatically receive these adjacency constraints with the house walls as well. This example clearly illustrates the power of meta-tile constraints in action.

In Figure 7 a design specification has been sketched out by an environment designer. Note how the meta-tile painting feature was used here; LAND, VILLAGE and FOREST are all meta-tiles, while WATER is a leaf tile, also showing how the designer can fluently incorporate both on the same grid in their design, thanks to adjacency inference. The meta-tiles allow the designer to loosely specify the spatial layout of the environment. What will end up being generated can

[2]Python prototype: https://github.com/Archer6621/HSWFC-editor-pygame
[3]Web prototype: https://github.com/Archer6621/HSWFC-editor
[4]https://numpy.org/
[5]https://docs.python.org/3/library/collections.html
[6]https://www.pygame.org/news
[7]https://quasar.dev/
[8]https://mathjs.org/
[9]https://www.npmjs.com/package/denque
[10]Web prototype demo: https://archer6621.github.io/hswfc-editor-dev/

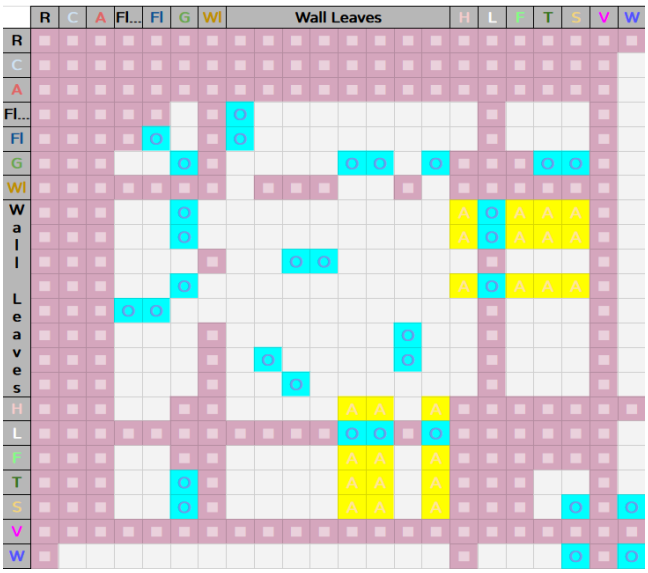

Figure 6: The west-side adjacency matrix that accompanies the DAG from Figure 3. Blue denotes an input adjacency, yellow an augmented adjacency from a meta-tile constraint, and purple an upwards cascaded adjacency.

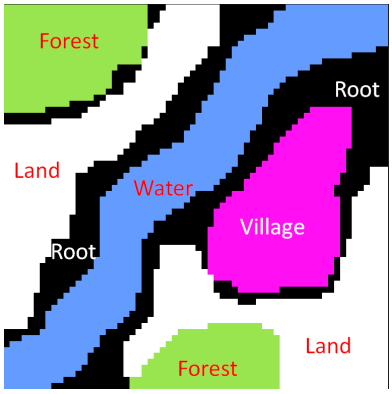

Figure 7: A design specification painted in the prototype editor implementation of hierarchical semantic WFC. Note that a mix of meta-tiles (e.g. VILLAGE) and leaf-tiles (e.g. WATER) was used here. Recall that the black tile corresponds to ROOT.

vary, but will be constrained by what is in the sub-tree of those meta-tile nodes, as shown in the DAG of Figure 3.

Subsequently, the designer can choose to resume automated collapsing, which will continue collapsing the meta-tiles downwards until all cells on the grid contain a leaf-tile. As this is a stochastic process, the design specification can result into multiple concrete manifestations, as shown in Figure 8. This process can be controlled to an extent by adjusting the weights on the DAG. In the right output, HOUSE was made slightly less common in VILLAGE by adjusting the weight on the edge connecting them to be 0.02 instead of 0.1, resulting into a less dense village. This shows how the meta-tile

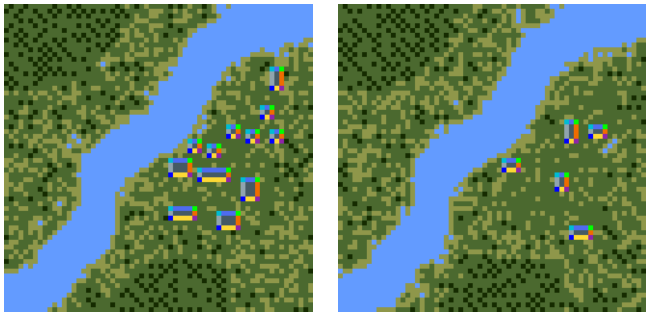

Figure 8: Two variations of environment generation from the design specification of Figure 7, but with different weights for the VILLAGE meta-tile.

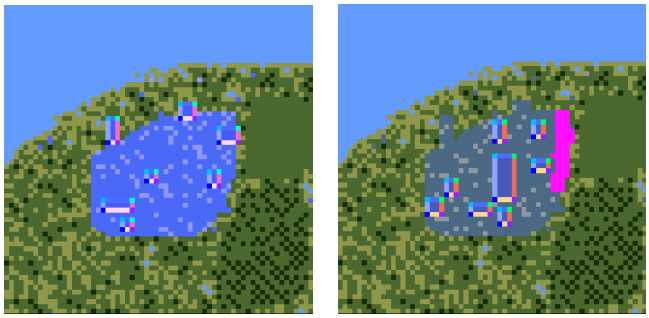

Figure 9: Regenerating the village highlighted in blue, after being able to select it through the collapse paths. The new still-generating village on the right clearly has a different layout.

weights are useful in controlling the distribution of entities based on their semantics.

Ultimately, after sketching and resuming the automated collapse for several iterations, the designer ends up with an output they are mostly satisfied with as seen in figure 9, except for the particular layout of the village. The designer can choose to select all tiles that have VILLAGE in their collapse path (highlighted in blue), and simply regenerate only the village until a layout is found that is satisfactory.

The usefulness of this last feature extends beyond the designer's scope, as other systems can make use of this semantics to identify separate objects/bodies as well. For example, the village in our example contains HOUSE meta-tiles, which can be easily used (in our hypothetical game) to add interactivity to all houses. For this, one could analyze cells' collapse paths and state that the player can interact with any tile that "originated" from HOUSE in some specific way. Another example would be to connect all houses with a road network using a dedicated algorithm. For that, all clusters of tiles originated from HOUSE can be treated as a node, as they would represent the houses in this network.

In Figure 10, various output examples are shown, demonstrating a variety of scenarios. One can see for instance that leaving areas unpainted (i.e. at the root tile) basically makes the generator decide what happens there, though one still has the freedom to tweak the

outgoing edges from the root node to make this suit the use case. Also note how in (v) the designer can constrain the output in a variety of ways by drawing certain shapes with the WALL meta-tile. Finally, in (vi) we can see the designer mixing various detail levels seamlessly, showing how you only need to specify the elements that matter to you, and leave the rest of the environment to the generator.

6 DISCUSSION

From the interactive examples in the previous section, we can identify a first clear benefit of our hierarchical semantic WFC approach: environment designers can focus on what they want to place, rather than having to worry about the detailed semantics blurred in each leaf tile. For instance, with the appropriate input, they can just place the semantic embodiment of a "house", which WFC will then collapse to the required tiles based on the context.

Secondly, the probabilities of "choosing" some tile can be tweaked thanks to the weights in the DAG structure. This gives control over the distribution and proportions of the contents of any semantic group. Designers can tweak this to express they want more of some concept (e.g. tree in a forest) in some area, and less in another (e.g. tree in a city). Without a hierarchical structure, this would be non-trivial.

Thirdly, the input designer has now control over the semantic composition and clustering of entities under each meta-tile, including the specification of their weights and of the constraints among them. This does not only save time, it also clarifies the intent of some constraint (e.g. use simply LAND <-> WALL, instead of GRASS <-> WALL, GRASS <-> WALL-CORNER, FOREST <-> WALL-HORIZONTAL-STRETCH, etc).

While promising in many respects, there are still challenges ahead to consider. For one, a sensible input remains an essential requirement for any WFC implementation, either created by the designer or obtained from third parties. The semantic power of an editor using WFC is limited by the quality of the input, and the designer would have to make amendments if the input is deemed insufficiently powerful or does not fully cater to their needs.

Furthermore, hierarchical semantic WFC certainly requires a more complex input, needing a hierarchy specification in the form of a DAG. While there are known methods for deriving an input tile-set and constraint-set from an image for the original WFC implementation, there are no known methods for deriving a semantic hierarchy from an image, hence to even get started with an editor that uses this algorithm, an appropriate input needs to be crafted by a designer first. This burden can be reduced slightly by using existing WFC inputs that have definitions for the constraints and tiles, and then simply expand the hierarchy with additional constraints involving meta-tiles in hindsight.

7 CONCLUSION

We have shown that extending WFC with a semantics-based hierarchical structure and meta-tiles enhances its potential as a driving algorithm for an interactive environment editor. This is achieved by facilitating novel editing features otherwise impossible or very tedious. These novel features allow a designer to express semantics much more clearly, use this semantics as a powerful input to WFC, and ultimately, maintain this semantics in generated environment, throughout its iterative refinement.

The implemented prototypes have proven that these extensions can be realized, and are effective without any deterioration of algorithm performance. We are presently working on up-scaling the approach to an environment editor that can be used by designers, with which we will perform a number of user studies to assess the design experience improvement, in terms of expressiveness and iteration time.

Looking further ahead, there are several interesting directions for future work. First, we would like to look into how the collapse chains in the output cells can be used for detecting semantic objects and specifying higher level constraints between them. Another interesting direction would be to examine more closely the editing workflow enabled by our method, which allows a designer to work on several scales at once, switching between detailed and global views at will.

It is definitely worth further exploring the integration of a semantic tile hierarchy notion with other WFC implementations (e.g. overlapping model, graphs), and identify additional promising features, both for automatic and for mixed-initiative content generation.

One could also look into making the automated generation step more useful, e.g. by allowing certain meta-tiles to have their own procedural generator, e.g. taking a number of contextual parameters for influencing the child-choice weights, in order to steer towards more desirable output. Lastly, it could be very interesting to investigate which other kinds of semantic relations could be modelled between the nodes in the DAG, e.g. relations that are based on proximity, where one type of tile is likely to occur close by another type of tile.

REFERENCES

[1] Maxim Gumin. 2016. *Wave Function Collapse Algorithm.* personal repository. https://github.com/mxgmn/WaveFunctionCollapse

[2] Peter Kan, Andrija Kurtic, Mohamed Radwan, and Jorge M Loaiciga Rodriguez. 2021. Automatic interior Design in Augmented Reality Based on hierarchical tree of procedural rules. *Electronics* 10, 3 (2021), 245.

[3] Isaac Karth and Adam Smith. 2021. WaveFunctionCollapse: Content Generation via Constraint Solving and Machine Learning. *IEEE Transactions on Games* PP (05 2021), 1–1. https://doi.org/10.1109/TG.2021.3076368

[4] 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* (Hyannis, Massachusetts) *(FDG '17)*. Association for Computing Machinery, New York, NY, USA, Article 68, 10 pages. https://doi.org/10.1145/3102071.3110566

[5] Isaac Karth and Adam M. Smith. 2019. Addressing the Fundamental Tension of PCGML with Discriminative Learning. In *Proceedings of the 14th International Conference on the Foundations of Digital Games* (San Luis Obispo, California, USA) *(FDG '19)*. Association for Computing Machinery, New York, NY, USA, Article 89, 9 pages. https://doi.org/10.1145/3337722.3341845

[6] Jassin Kessing, Tim Tutenel, and Rafael Bidarra. 2012. Designing semantic game worlds. In *Proceedings of PCG 2012 - Workshop on Procedural Content Generation, co-located with the Seventh International Conference on the Foundations of Digital Games*. ACM, Raleigh, NC, 40–48.

[7] 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. *Proceeding of IEEE Conference on Games* 1, 1 (08 2019), 1–4. https://doi.org/10.1109/CIG.2019.8848019

[8] Thijmen SL Langendam and Rafael Bidarra. 2022. miWFC - Designer empowerment through mixed-initiative Wave Function Collapse. In *Proceedings of the 17th International Conference on the Foundations of Digital Games* (Athens, Greece) *(FDG '22)*. Association for Computing Machinery, New York, NY, USA, Article 66, 8 pages. https://doi.org/10.1145/3555858.3563266

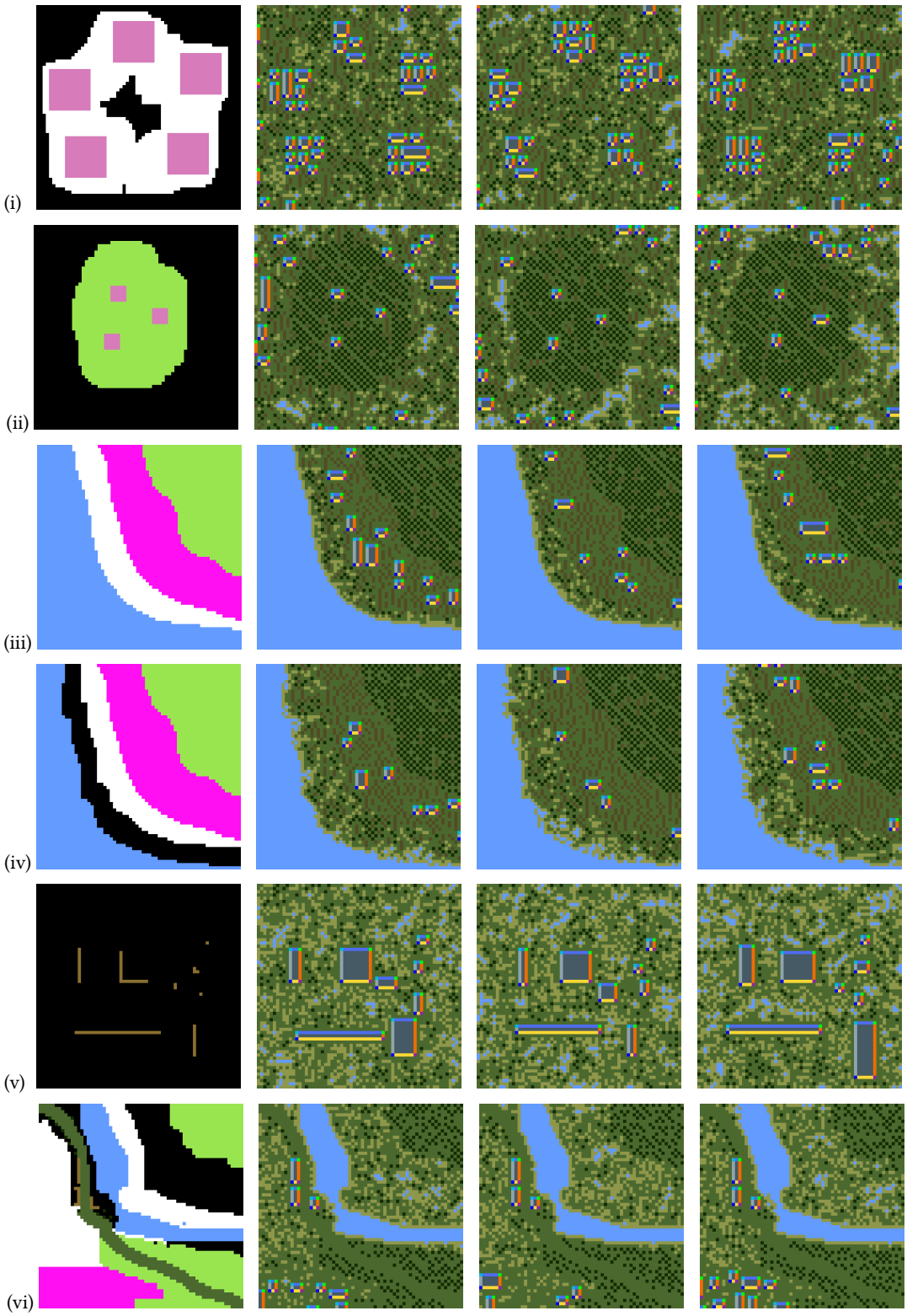

Figure 10: More examples of meta-tile paintings with three variations of their generated output, from top-to-bottom: (i) Clusters of HOUSE surrounded by LAND, causing houses to generate only inwards; (ii) houses in forest, (iii) shoreline village and forest, (iv) shoreline with margin, (v) various wall configurations, (vi) mixed detail: grassy path along some forest, walls, a river, and a village at the bottom.

[9] Alain Lioret, Nicolas Ruche, Etienne Gibiat, and Cédric Chopin. 2022. GAN Applied to Wave Function Collapse for Procedural Map Generation. In *ACM SIGGRAPH 2022 Posters* (Vancouver, BC, Canada) *(SIGGRAPH '22)*. Association for Computing Machinery, New York, NY, USA, Article 59, 2 pages. https://doi.org/10.1145/3532719.3543198

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

[11] Tobias Nordvig Møller, Jonas Billeskov, and George Palamas. 2020. Expanding Wave Function Collapse with Growing Grids for Procedural Map Generation. In *Proceedings of the 15th International Conference on the Foundations of Digital Games* (Bugibba, Malta) *(FDG '20)*. Association for Computing Machinery, New York, NY, USA, Article 28, 4 pages. https://doi.org/10.1145/3402942.3402987

[12] Konstantinos Sfikas, Antonios Liapis, and Georgios N. Yannakakis. 2022. A General-Purpose Expressive Algorithm for Room-Based Environments. In *Proceedings of the 17th International Conference on the Foundations of Digital Games* (Athens, Greece) *(FDG '22)*. Association for Computing Machinery, New York, NY, USA, Article 64, 9 pages. https://doi.org/10.1145/3555858.3563262

[13] Ruben Smelik, Krzysztof Galka, Klaas Jan de Kraker, Frido Kuijper, and Rafael Bidarra. 2011. Semantic Constraints for Procedural Generation of Virtual Worlds. In *Proceedings of the 2nd International Workshop on Procedural Content Generation in Games* (Bordeaux, France) *(PCGames '11)*. Association for Computing Machinery, New York, NY, USA, Article 9, 4 pages. https://doi.org/10.1145/2000919.2000928

[14] 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 (2011), 352–363.

[15] Ruben M. Smelik, Tim Tutenel, Klaas Jan de Kraker, and Rafael Bidarra. 2010. Integrating procedural generation and manual editing of virtual worlds. In *Proceedings of PCG 2010 - workshop on Procedural Content Generation, co-located with the Fifth International Conference on the Foundations of Digital Games*. ACM, Monterey, CA, 8 pages. http://graphics.tudelft.nl/Publications-new/2010/STDB10d

[16] Sam Snodgrass and Santiago Ontanon. 2021. A Hierarchical MdMC Approach to 2D Video Game Map Generation. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment* 11, 1 (Jun. 2021), 205–211. https://doi.org/10.1609/aiide.v11i1.12794

[17] Oskar Stålberg. 2021. Townscaper. Raw Fury, Steam, Epic Games Store, GOG, Nintendo Switch, xBox, App Store, Google Play. https://oskarstalberg.com/Townscaper/

[18] Tim Tutenel, Rafael Bidarra, Ruben M. Smelik, and Klaas Jan de Kraker. 2009. Rule-based layout solving and its application to procedural interior generation. In *Proceedings of the CASA'09 Workshop on 3D Advanced Media in Gaming and Simulation*. Utrecht University, Amsterdam, The Netherlands, 15–24. http://graphics.tudelft.nl/Publications-new/2009/TBSD09a

[19] Barbara Tversky and Masaki Suwa. 2009. *Thinking with Sketches*. Oxford University Press, Oxford, Chapter 4. https://doi.org/10.1093/acprof:oso/9780195381634.003.0004