Organized Order in Ornamentation

Lena Gieseke
Film University Babelsberg Konrad Wolf

Paul Asente
Adobe Research

Jingwan Lu
Adobe Research

Martin Fuchs
Stuttgart Media University

ABSTRACT

Decorative ornamentation involves a careful balance between accent and order. Existing techniques leave artists either with tedious manual processes or the uncontrolled automatic generation of rather homogeneous patterns that lack creatively-placed visual highlights. We present a method to close this gap, offering the control and quality of manual creation, and the efficiency and accuracy of computation. At the core of our system, customizable and modularly combinable element placement functions fill a space automatically under global design constraints. We provide a set of example placement functions that implement order based on design principles for ornamentation such as balanced element distribution and symmetry. To create structural hierarchies and to guide an ornament to the space it fills, we allow artists to direct the connectivity of elements with drawn strokes. Artists can also draw guides to create vector fields, which organize the ornament along streamlines. Path planning automatically routes around obstacles while aligning the ornament to their borders. Our method combines high-level control mechanisms like taking guidance from example images to low-level control like placing single elements as visual accents and making local edits within the computed ornament. By automating tedious tasks and offering familiar input mechanisms like drawing, we enable artists to focus on the creative intent.

CCS CONCEPTS

• **Computing methodologies** → **Non-photorealistic rendering**; • **Applied computing** → Media arts;

KEYWORDS

Paint Systems, Interaction Techniques, Modeling Interfaces, Procedural Modeling

ACM Reference format:
Lena Gieseke, Paul Asente, Jingwan Lu, and Martin Fuchs. 2017. Organized Order in Ornamentation. In *Proceedings of CAe'17, Los Angeles, CA, USA, July 28-29, 2017,* 9 pages.
DOI: 10.1145/3092912.3092913

CAe'17, Los Angeles, CA, USA

DOI: 10.1145/3092912.3092913

1 INTRODUCTION

Throughout all cultures and times artists have used ornaments to embellish the world around them. One common class of decorative ornaments, as discussed by Wong et al. [1998], creates an underlying perception of order by placing individual components repetitively and in a balanced way. But ornaments also include hierarchical structures with visually dominant elements and connections as accents. These often do not follow the underlying order of the ornament, breaking an otherwise too-homogeneous appearance. Additionally, ornaments adapt to the space they fill by aligning to its boundaries.

It takes an experienced artist to balance the contrast between carefully chosen visual accents and creating a sense of order by applying compositional rules and complementing the space, as shown in the examples in Figure 1.

While artists are indispensable for the creative task of creating an overall layout and placing accents, executing structuring rules and completing the ornament to a cohesive whole is tedious and worth automating. We propose a hybrid technique that gives artists artistic control through familiar tools like sketching, while computing ordered structures automatically, unburdening the artists from tiresome tasks. We aim to offer both the control and quality of manual creation, and the efficiency and accuracy of computation.

Our contributions are:

- an optimization strategy that incorporates customizable and modularly combinable placement functions putting global design constraints explicitly under artist control,
- a ready-made set of placement functions that fulfill design principles for ornamentation [Wong et al. 1998] with a balanced element distribution and symmetry constraints,
- the control of element connections through the translation of visual input into connection strategies under the given global design constraints, combining elements and connections into a cohesive whole,
- the use of path planning to efficiently route the ornament around obstacles
- the incorporation and combination of control mechanisms at all scales, ranging from taking high-level guidance from example images down to placing single elements and making local edits within the computed ornament while maintaining its procedural nature.

The feedback of designers using our system confirms the appeal, efficiency and further potential of our methods.

Figure 1: The top row shows historic and the bottom row commercial examples of hierarchically ordered ornaments. They balance accent and order and complement the space they fill. See end note for image sources [A - J].

2 RELATED WORK

We focus the following discussion of related work on procedural methods, as the core aspect of our system is its procedural nature.

The pioneering work of Prusinkiewicz and Lindenmayer [1990] applies L-systems to algorithmically model plant growth. Various extensions [Parish and Müller 2001; Prusinkiewicz et al. 2003] demonstrate their expressiveness for different applications. However, since the execution process is inherently hierarchical, L-systems have difficulty supporting artistic control mechanisms that range from global to local scale.

Wong et al. [1998] introduced a programmatically controllable procedural system that employed a greedy rule-based strategy to generate floral ornaments. We take inspiration from their work but focus on enabling a usable tool by adding artist-friendly control mechanisms in a unified way. We generalize and technically improve their space-finding algorithm, enabling the explicit enforcement of ornamental principles and unburdening artists from implementing them for each pattern individually. Their method can only control the connections of elements by writing code; we add to their work by giving intuitive design options for directing connectivity. This, combined with the option to place single elements freely, enables ornamentations with hierarchical structures that go beyond repetitive patterning. Anderson et al. [2008] adapted Wong et al.'s work by placing discrete elements along a user-given curve. However, they solely decorate the regions adjacent to the curves, offering only limited control and design options. Etemad et al. [2008] pick up a rule-based strategy for a dynamic recreation of Persian floral patterns, focusing on animation, not on controllability.

Beneš et al. [2011] offer certain global control on the procedural process by dividing a target space into user editable guide shapes. The shapes determine what types of patterns grow in different areas. The connections between the shapes are manually specified by the user and in turn guide the connections between elements. In our approach, element connections are automatically derived from visual guides. Other systems provide global control on the procedural process by interpreting the modeling task as a probabilistic inference problem [Ritchie et al. 2015, 2016; Šťava et al. 2014; Talton et al. 2011], or optimize a packing problem under specific constraints [Chen et al. 2016]. They all control the overall shape of the resulting ornaments, but do not permit the hierarchical or element-level local controls we support, such as specifying the locations of individual elements.

Various example-based approaches [Bradley et al. 2013; Ijiri et al. 2008; Ma et al. 2013, 2011] give artists indirect control over the resulting pattern by defining an exemplary element arrangement a priori. They extract the spatial relationship between elements and attempt to reproduce the relationship in their synthesis results. But their methods do not allow placing accents with single elements, which can break up the perception of a homogeneous texture and helps to create a compelling ornament.

Ijiri et al. [2008] use sketch-based user input to create global control structures such as an underlying vector field to guide procedural growth. We also make use of vector fields generated from artists' sketches, but we use them to establish element connectivity and to guide to growth of a model. Vector fields are further employed in the specific context of procedural street modeling [Chen et al. 2008a] and micrography [Maharik et al. 2011]. They have been integrated into formalized grammars, such as a vector-field guided shape grammar [Yuanyuan Li et al. 2011] and L-system rules [Šťava et al. 2010]. These systems produce patterns that are more homogeneous than the decorative ornaments that we aim for. Xu and Mould [2015] trace shortest paths in vector fields to generate branches for botanic tree modeling. Aiming for a different application, we support global design goals and, in addition, plan growth paths of ornaments around layout obstacles.

Our different types of user input are inspired by painting-tool-like methods in procedural modeling [Chen et al. 2008b, 2012; Emilien et al. 2015; Měch and Miller 2012; Palubicki et al. 2009], in particular by the flexible *Deco* procedural engine [Měch and Miller 2012] upon which we constructed our implementation. We extend these methods by combining the ability to follow input curves while taking the whole environment into consideration, applying the chosen design principles when placing elements on the paths as well as when automatically filling the remaining space.

None of the work discussed so far integrates artist control on an element and connection level once the pattern is computed. There are procedural techniques that enable low-level control on the results themselves, developed in the context of architectural designs [2008], tree modeling [Pirk et al. 2012] and the creation of natural scenes [2015]. The move operator from the latter is similar to ours but their system is optimized for chaotic arrangements, which contrasts to our organized design goals for ornaments.

3 SYSTEM OVERVIEW

Since our approach extends the technique introduced by Wong et al. [1998], we briefly summarize their approach. A procedural model is created with artist-defined elements and a set of growth rules that handle the selection of elements, their appearance and connections. Their technique finds tentative places for elements by testing them against constraints in the procedural model, and, where suitable, placing elements in the found spaces, optionally connecting them to existing elements. Possible ornament designs are technically restricted only by this iterative creation logic. Wong et al. find the next space to fill by computing the medial axis of a

shape stencil using the Manhattan distance, then inflating circles centered on points on the axis until they collide with geometry proxies or the stencil shape. The new element is placed at one of the circles of maximal radius (Figure 3, top row) and connected to the closest existing element. The system then iterates.

Our approach performs a similar greedy iterative process but generalizes it by using placement functions. Wong et al.'s technique of placing the next element into the largest possible space is one possible placement function, but there are many others. As we aim for direct interactions, performance is crucial, so we furthermore modified their implementation to allow interactive performance (see Section 9).

Figure 2 shows an overview of our technique. As a first step, the system can be configured to express global design goals by using algebraic placement functions. Higher values of the functions indicate preferred locations for element placement (see Section 4). Placement functions have a variety of potential inputs, such as a stencil defining the areas to fill, or a desired type of symmetry.

The artist specifies these design constraints through our interface and can also provide other input such as sketched paths to guide the connections between placed elements (see Section 5). Based upon the input, we configure and combine a set of ready-made placement functions that implement the artist's intent. The artist can also specify exact locations for certain elements by directly placing them in the space to be filled.

Our automated placement system then repeatedly evaluates the placement functions to find the locations with maximal values, and inserts elements into the output ornament accordingly. The artist can at any time interrupt the process and make changes using editing techniques like moving or deleting existing elements. The system immediately adapts to these local changes.

The following sections give more details on our process. For coherency, all our models were designed by the same artist and thus share visual traits specific to the artist's individual style. By presenting comparable models with similar growth rules, we aim to highlight the variety of possible designs implemented by the system, not the model's hard-coded rules.

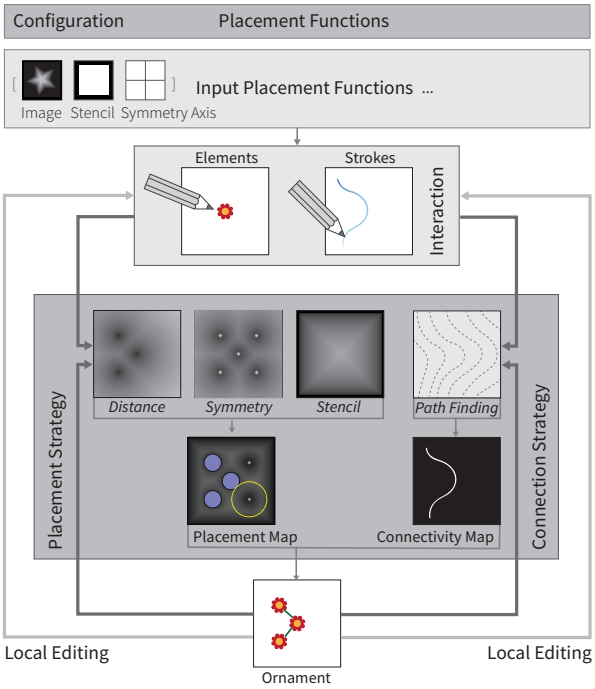

Figure 2: Our technique can be configured to incorporate global design goals such as symmetry. The artist can optionally place specific elements and draw desired connectivity. At any time during the process, the artist can insert, delete and move elements on the canvas and the system adapts to the change.

4 PLACEMENT STRATEGY

Artists start by specifying global design goals for the automatic placement of elements. Some of these goals, like desired growth direction, take additional input through lower-level mechanisms like sketching.

Our set of supplied global designs aims to fulfill more explicitly underlying aesthetic principles of ornamentation. Wong et al. [1998] thoroughly discuss these but their method only indirectly and uncontrolledly implements them. They summarize the aesthetic principles of ornamentation as repetition, balance and conformation to geometric constraints. We support balance and repetition through symmetry constraints and facilitate conformation to geometric constraints by element connection strategies (see Section 5).

4.1 Placement Functions

We strive to support a wide variety of design goals with few limitations. For this, we modularize the creation of global order for placing elements through the definition of placement functions. A placement function $p : \mathbb{R}^2 \rightarrow \mathbb{R}$ takes higher values at preferred locations. p is updated after every placement and its values decrease. Once $\max\{p\}$ reaches or falls below 0, the ornament is completed.

We provide a number of fundamental placement functions, and the system combines them into one overall placement function based on the user-specified design goals. The functions can make use of user-supplied input like shapes, images, and paths. They can also use internal data structures like the locations of the centers of placed elements, and a map of rasterized proxies for them.

Our supplied placement functions implement:

- A stencil function accepting a binary stencil that defines the area to be filled.
- Symmetry functions for different types of symmetry.
- Image data functions accepting a grayscale image that controls the desired element placement based upon image brightness.
- Path functions supporting element placement along paths.

In addition, users with scripting experience can readily extend the system by adding new placement functions. We provide a number of functional building blocks like *min*(), *max*(), *translate*(), and *rotate*(), which can be combined using the usual mathematical function operations and can be extended with custom code. These new

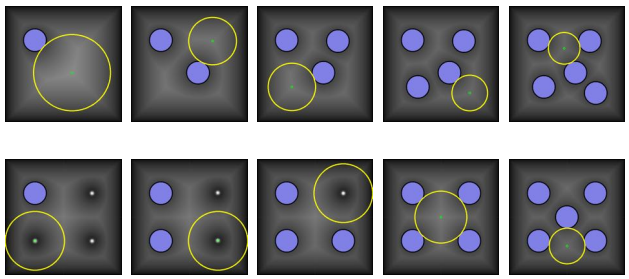

Figure 3: The evolution of the placement function with an initial manually-placed element. The placed element proxies are in blue and the green dot indicates the proposed next location, with the permissible maximal radius in yellow. The gray levels in the map correspond to placement preferences, with brighter values corresponding to preferred positions. The top row shows a placement without explicit symmetry, the bottom row with explicit four-way symmetry.

functions have the same access to user input and internal data structures as our supplied placement functions.

While placement functions provide the framework for unifying and combining many sorts of design constraints, their presence is completely invisible to the end users, who specify constraints and guidance in conventional ways like sketching and choosing among options. The system feeds their input to the placement functions and combines them to implement the desired constraints.

The following subsections briefly discuss the construction of the ready-made placement functions and how they interact to support design goals. For a complete formal description of the functions please refer to the supplemental materials.

The Stencil: The stencil function is a simple function that takes as input the value 1 for areas to be filled and 0 elsewhere.

Symmetry: Wong et al. place elements by maximizing the distance between the new element to both the stencil border and previously placed elements. Figure 4, left, shows that this strategy when applied to a symmetrical stencil can lead to an ordered, highly symmetric ornament. In the example, the algorithm places elements on the symmetry axes of the rectangular stencil shape. The elements partition the space so that following insertions maintain the symmetry. As a result of the greedy search for maximal spaces, larger elements are placed before smaller elements, creating a visible order hierarchy that is characteristic for many ornamental styles.

However, if the artist pre-places an element off-center, or a non-symmetrical stencil is used, or models contain randomized characteristics, applying this strategy without modification leads to unorganized patterns, as shown in Figure 4, center. Figure 3, top row, shows the sequence of element insertions for this case. To solve the issue, we allow the artist to explicitly express symmetry in a way that can integrate pre-placed elements for any stencil, as shown in Figure 4, right, and Figure 3, bottom.

The desired symmetry is supported by a placement function that implements a set of heuristics that work as follows:

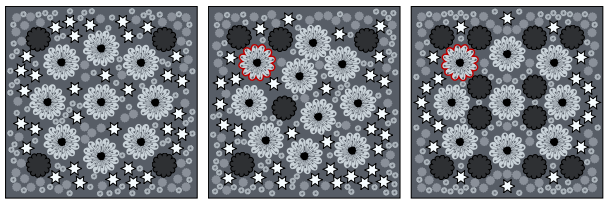

Figure 4: Greedily placing elements to maximize the distance to the frame and previously placed elements can sometimes generate tilings with high symmetry (left). With a single artist-placed first element, marked in red, this approach breaks down (center). By making the desired symmetry explicit, we can give artists the option to manually place elements while maintaining the symmetry (right). Unless otherwise noted, none of our results were edited locally after the computation.

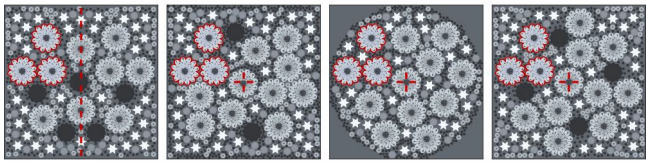

Figure 5: Symmetry generation examples for different symmetry types with the same pre-placed elements (red). From left to right, reflection across an axis, reflection across the center point, three-way rotational symmetry using a three-way symmetric stencil, four-way rotational symmetry.

(1) If an element was placed at some location (x, y), we prefer filling its transpositions under the symmetry transformation, placing the new elements in mirror or rotational symmetry. Hence, the placement function should have the highest values at transformed locations of previously placed elements.
(2) If the placement at such a location is impossible (e.g., because the symmetry set is already complete), we have some freedom. In addition to keeping a maximal distance to the stencil, we prefer placing the next element at a location that permits future elements to be placed at symmetry-creating locations without collisions with existing elements.
(3) Finally, we exclude placements that are too close to existing elements or outside the stencil.

The bottom row of Figure 3 shows the mechanism working for four-way mirror symmetry. With the chosen settings, the first three insertions fulfill symmetries of the pre-placed circle according to heuristic 1 and the next two circles are placed following heuristic 2. Our interface allows the user to choose among various symmetry types; Figure 5 shows several results.

Controlling Placements with Image Data: To show the flexibility of our system, we provide a placement function controlled by an artist-specified example image, as shown in Figure 6. This function uses the brightness of the image to prioritize placement order and

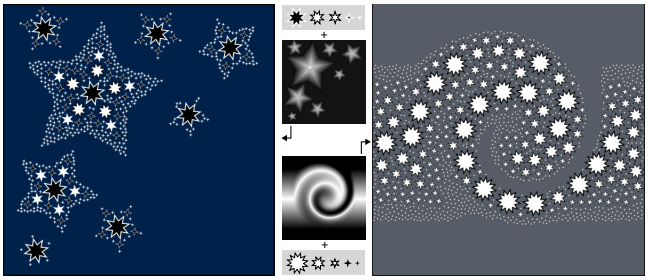

Figure 6: Demonstrating the versatility of placement functions, gradient images in the middle are used to guide the ornaments, with the size and type of placed elements determined by the gradient's brightness.

also uses it to guide the selection of elements. A cut-off for the size leaves the black parts of the example empty.

Paths: To place elements along an artist-specified path we interpret it as a stencil that sets all points off the path to 0. This forces the greedy placement process to put elements along the path according to all other given global design constraints, for example with symmetry constraints, as shown in Figure 7, center and right. After the potential positions on the path are exhausted, our algorithm completes the remaining background. These paths also affect element connections, as described in the following section.

5 CONNECTION STRATEGY

Spatial relationships between elements are often expressed by geometric curves or patterns that connect nearby elements. These interconnections add additional levels of order to an ornament in a structured way. Furthermore, global layouts, such as a frame around a text box, can be achieved by appropriate interconnections. In addition to singular global structures, such as specific frames (see Figure 7, center), many complex real-world ornaments adapt themselves in their entirety to global designs, such as in Figure 8, left, where the whole space is structured with multiple lines.

In the work of Wong et al. [1998], the connections between elements arise from individual, model-specific growth rules. Our method also includes these programmatic growth rules but we add direct visual controls to define connectivity by drawn paths or sketched vector fields. This approach permits greater control than related techniques, as the artist-defined placement functions also guide the positioning of the elements on the paths.

5.1 Placement on a Single Path

Once all elements have been placed along an artist-specified path according to the global design constraints, we sort the elements according to their distance to the path's starting point and connect neighboring elements to express the path. This connection strategy also works for self-intersecting paths as shown in Figure 7, right, where a simple proximity-based strategy might fail.

5.2 Connecting Elements using Vector Fields

A single user-defined path is not sufficient to enforce the connectivity in a large area. While artists could carefully fill the entire space with paths by hand, we resolve this tedious task by letting them sketch a vector field, which structures a plane by storing orientation information at each plane position. The field can be created from a rough sketch or from an example image. We extrapolate sketched guides to dense vector fields by applying the method of Maharik et al. [2011]. For images, we compute the gradient and rotate it by 90° to find the orientation of its isolines. The streamlines of the vector field form a natural, dense and connected organization of the ornament space. Streamlines are found by picking a seed point and tracing out a path with vector field integration.

In order to construct the ornament, we either pick a maximal-length streamline or start at an artist-given seed point. The elements are placed on the traced streamline as if it were a user-specified path (see Section 5.1). Then, we mask the area around it with a stencil (see Section 4.1) and zero out the corresponding areas in the vector field. We repeat this process with the next-longest streamline, or the next seed, and iterate. Similarly to placing larger elements earlier than smaller elements, following longer stream lines before shorter ones contributes to the hierarchical space organization.

Figure 8, left and right, show the synthesis results guided by vector fields constructed from sketched input and from an example image, center. A vector field can also be applied, if so desired by artists, to guide model specific growth characteristics. Figure 8, right, shows background elements that are only allowed to grow in the direction of the underlying vector field.

6 RESOLVING COLLISIONS

As ornament designs are often part of a complex layout, an artist can specify geometric constraints either with stencils or by drawing obstacles. This might lead to an intersection of the defined paths and obstacles. To enforce the perception of an ornament adapting to the space it fills, we guide element paths around intersecting obstacles. We interpret this as the classical shortest-path problem between the start and the end of the path with the image pixels as nodes and 8-connections to non-obstacle pixels as edges. Using Euclidean distance between the adjacent pixels as edge weights leads to a short path. However, this path may not follow the user-intended shape (see Figure 9, left). Therefore, we add the distance between each node and the direct path, which ignores obstacles, to the movement costs from that node to the goal; see Figure 9, center and right. While the result may not be smooth, the size of the elements of the ornament in comparison to the jaggedness of the path at the pixel level empirically compensates for the roughness.

7 LOCAL EDITING

At any time during the computation process, or after the ornament is completed, the artist can directly interact with the elements of the pattern and their connections. Artists are familiar with manual interactions such as placing an element and moving it around on the canvas as part of their everyday workflow. By adopting these methods we narrow the gap between procedural modeling and manual creation. For a demonstration of local editing, please refer to the supplemental video.

Internally, we keep a directed graph of the ornament in which nodes represent elements and directed edges their connections. After each interaction the graph structures are updated. Connections

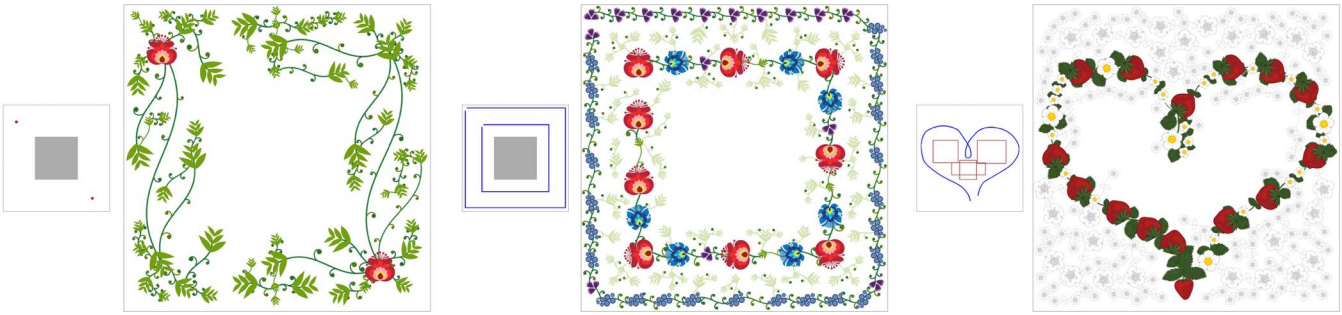

Figure 7: Space is filled in an ornamental manner to complement an artist's input. Stencils are indicated in gray, drawn obstacles in red and paths in blue. Left, the red flowers were manually placed. Center and right, elements are connected along artist-specified paths, shown in blue. Note that the paths were drawn quickly by hand or with a rectangle tool and the spacings are not optimized.

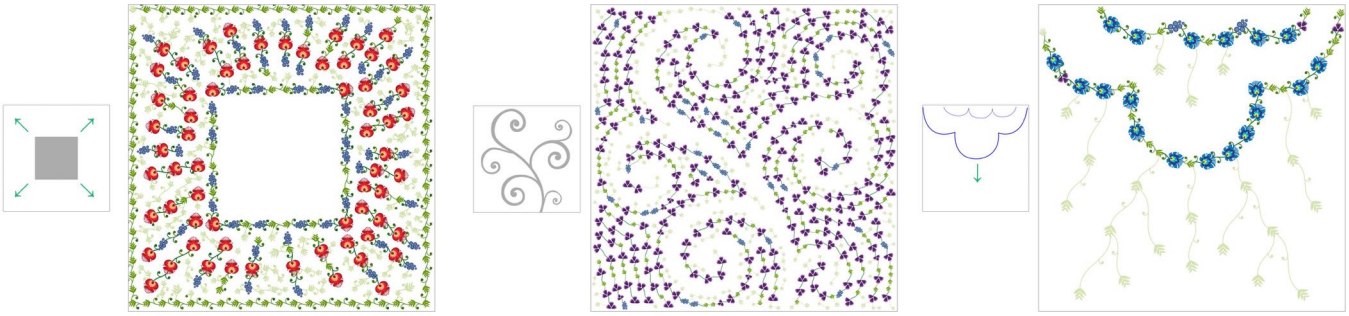

Figure 8: These examples demonstrate the application of vector fields, the directions of which are indicated by the green arrows. Left, the streamline of the field are traced iteratively. Center, a complex vector field is computed based upon contours from an input image. Right, drawn paths define the connectivity of the foreground elements, but the vector field guides connections in the background, not with streamlines but by only allowing limited growth in the direction of the field.

Figure 9: Path planning around obstacles. Minimizing just the geometric path length makes the ornament avoid the obstacle (black line), but the path follows the obstacle's shape only on the upper side (left). We prefer it to follow it on both sides, which we achieve with modified edge costs that pull the path towards the straight line between the endpoints (center). The path is pulled in between the lower obstacles, aligning the path better to the obstacles (right).

that violate rules and orphaned elements are detected and reconfigured. Therefore the model adapts itself to the changes and keeps itself consistent with the growth rules of the model. This retains the powerful interaction capabilities of a procedural model, such as changing specific element characteristics for all samples at once.

Specifically, for the local editing we offer deleting elements and/or their connections and picking up and moving single elements. New elements can also be added. Artists can move or add elements freely without influence from the underlying placement function, and we eliminate overlaps by deleting any elements or connections that intersect. As the artist moves or inserts an element we adapt the ornament to the changes interactively, recomputing its connections according to the rules of the model — for example connecting to the currently closest element. If elements are deleted the space remains empty, but to keep the model intact connections between remaining elements and the now deleted element are reconnected according to the connections rules (Figure 10, left).

8 DESIGNER FEEDBACK

To validate our approach, we performed a study to collect high-level feedback on our methods and evaluated their general strength and weaknesses. Because design tasks take considerable expertise, we sought primarily qualitative feedback from designers, in accordance with evaluations of similar techniques [Kazi et al. 2012; Nakagaki and Kakehi 2014].

Eight self-identified professionals took part in the study. Six were students with a course of study in audio-visual media. Four

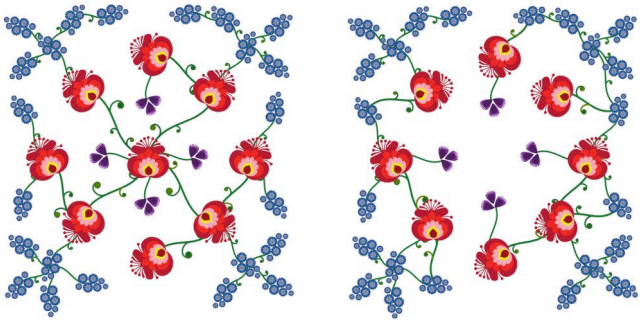

Figure 10: After the manual deletion of the red flower in the middle, the stems that were connected to it are re-connected and aligned to the new connections. This is the only figure in which our results were edited locally after the computation.

Figure 11: In the top row three results computed with the technique from Wong et al., in the bottom three results from our technique with a fourway-symmetry and no manual adjustments. The images in each set differ because this model has many randomized features, such as the shape of the flowers or the stems. This randomization leads to drastically different results for Wong et al.'s technique. As they offer no control mechanisms, the only option an artist has is to execute the algorithm until an favorable design is archived. During the feedback session, 6 out of 8 designers preferred our results.

of the participants rated their design knowledge as *Intermediate*, three as *Advanced* and one as *Expert*.

The study took about an hour and consisted of a brief explanation of ornaments, a task comparing our results to related work, a tutorial session with our tool, three tasks to complete with our method, and an optional task with Illustrator for the participants that rated their Adobe Illustrator knowledge at least *Intermediate*.

8.1 Evaluation

Before knowing about our methods we had the participants compare three ornaments computed with our technique with three of Wong et al.'s [1998] as baseline (Figure 11). For the computation we used Wong et al.'s original implementation to embed the algorithm in our framework. We chose to compare visual quality and not the usability of the methods, as most designers, having no or little programming skills, would not be enable to use Wong et al.'s system. Instead we compared the use of our system to Adobe Illustrator, which is a preferred tool for designers practicing the art.

For the visual comparison, our results were generated with underlying horizontal and vertical symmetry and had no local editing applied. Six of the eight participants preferred our results based upon their more symmetrical appearance and more ordered structure. The explanations for favoring Wong et al.'s [1998] results were "...more balanced in terms of the colours" and "I prefer result set 1 [Wong et al.'s], as there is less order". Neither method specifically handles color, so for both the color distribution is uncontrolled. The second comment indicates that it might be worthwhile to make the degree of order adjustable – another participant later said, in contrast to this, "More symmetry would have been great...".

The survey included 17 Likert-scale questions about our tool in general, specific methods, and a comparison to Illustrator (answered by the 6 of the 8 participants who had sufficient Illustrator knowledge). The quantitative evaluation shows an overall approval of our system (Figure 12) but its results are less expressive due to the limited number of participants. The Likert-scale questions were mainly intended to motivate further comments on the topics in an open-ended fashion. Additionally, we asked the participants "What did you like about the tool?", "What did you dislike about the tool?", "Any ideas for new/missing features?", and "Any further comments?". We clustered the answers by the number of times that a specific topic was mentioned.

For positive feedback the most common comments were that our methods save time (6 times) and are easy to use (4 times). There was praise for the general concept (2) and that it enabled an artist to explore designs (2). For specific methods, local editing (2) and the application of a vector field (2) were mentioned, with one comment saying "In particular I enjoyed the flow fields as it felt that they allow me to orchestrate the picture on a higher level."

In terms of negative feedback, most arose from missing feature implementations (6) and the consequential lack of control (4). For missing features there were many requests for convenience features (6) such as undo functionality, grid alignment or better previews of actions. Further control of element and path characteristics, such as their size, was also desired (4).

From our analysis of the Likert-scale numbers and the open-end answers, we conclude that our methods were well-received overall. They are effective at saving creation time and effort, and the participants all agreed that they are fun to use, even more so than the known tool Illustrator. Our results hold up well in terms of visual quality when compared to Illustrator, with the mean of the responses slightly preferring the tool's results to their Illustrator work. All negative comments were in regard to missing feature implementations; no one questioned our overall concepts. Adding more specific functionalities would also improve the controllability of the results. Nonetheless, one participant even said "I also liked that it gave results that probably would not have been my first choice, but might serve as inspiration for further exploration. Like looking at nature, processes out of our control can give new input into our own designs."

Please find the full study and the responses in the supplementals.

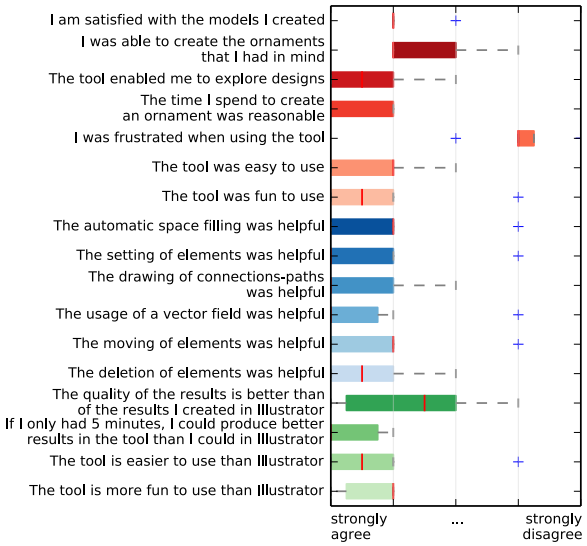

Figure 12: Quantitative evaluation of the Likert-scale questions. A box represents the first to third quartiles, with the red line indicating the median and the whiskers including a range of [(Q1-1.5 IQR), (Q3+1.5 IQR)], with IQR being the interquartile range. Values outside of that margin are indicated with blue + markers. The questions are categorized as applying to the tool in general, shown in red, to specific methods, shown in blue, and in comparison to Illustrator, shown in green.

9 PERFORMANCE DISCUSSION

Our placement strategy has an innermost loop that repeatedly finds the next optimal insertion location after each element has been placed. To accelerate the process, we keep a rasterized version of the placement cost in memory and store it as an image pyramid P, in which each pixel stores the maximum value of four pixels on the next-lower layer as well as the coordinate of the maximum pixel on the source layer. Thus, its single pixel at the top equals $\max\{p\}$, with p being the placement function.

Whenever one of the building blocks of p needs updating, we identify the changed region, track it through all its transformations, and recompute p only for the affected area A. Then, we update P, starting from the bottom with the pixels in A, and recursively work our way to the top until no more updates are required or $\max p$ has been recomputed.

With this strategy, placing the first few elements is slow, as the entire function changes, potentially changing the location of the maximum. Subsequent elements are placed faster and faster, resulting in a total runtime of a few hundred millisectonds to fill 512^2 pixels without symmetry constraints, a few seconds with symmetry, and up to several minutes for the most complex examples. A discussion of the asymptotic runtime is provided in the supplementals.

10 LIMITATIONS

Element groups in symmetric locations are occasionally constructed from different element types, especially if the elements are rather small. This limitation arises from our placement functions being processed on a discretized pixel grid. Accordingly, elements can not be placed at exact locations in the plane, but only at integer pixel coordinates. The example of three-way-rotational symmetry in Figure 5 shows the limitation: some of the smallest elements are not in perfect symmetry to copies of the same type, but to smaller elements. The reason for this lies in the way the placed element groups are constructed: while the first element may fit well into a particular place, the next may be partially occluded due to rasterization artifacts, and therefore a smaller element takes its place instead. The same problem arises from models including variability, such as the strawberry model, in which the leaves' types and sizes are randomized (Figure 7, right).

Aesthetics, even in the special case of ornaments, are subjective. We define our set of example placement functions to create ornamentation based on design principles found in related work. Nonetheless, perceptions differ and, as mentioned in the analysis of the expert feedback, while one participant prefers less ordered results, another one strives for absolute uniformity. Resolving this issue would require even more controllability, ultimately for all characteristics at all times during the process. This would have to be balanced against decreasing the ease of usage.

11 FUTURE WORK

In this work, we addressed the problem of placing elements according to global design constraints on their position and size. As future work, further visual properties could be constrained, such as the orientation of individual elements to better satisfy symmetry. Controlling the overall color distribution would also be worthwhile.

Our greedy method of placing the next element based on the placement function does not consider connections when computing the next space to fill. Here a global optimization or distribution strategy, taking the connections of the elements also into consideration, might be an alternative, but possibly at the cost of the interactive performance we aim for. Further research on this is called for, but beyond the scope of our paper.

After thorough testing of the local editing feature and a preliminary run-through with a designer, we deliberately chose the 'what you see is what you get' principle, as changes, for example to element positions, that were not directly triggered by the artist are hard to anticipate and the system would lose controllability. Exploring this trade-off is future work. We also would like to explore an idea proposed by one study participant regarding moving elements: rather than deleting elements that overlap the new location, push the elements around as with a mass-spring system. This would be interesting in combination with our global design constraints and could be especially promising with an underlying vector field, letting the elements be pushed in a meaningful direction.

The implementation of the procedural ornaments themselves requires programming skills. Hand in hand with our contributions regarding the usability of the models, it would be equally worthwhile to investigate a more artist friendly creation processes for the underlying procedural models.

Lastly, Li et al.'s work [Yuanyuan Li et al. 2011] applies their grammar in 3D space, a desirable extension for which we are aiming in the future.

12 CONCLUSION

Our technique defines a general ornamentation framework that brings user interaction to a task that is currently either fully automated or fully manual. Our uniform approach supports control on various abstraction levels. It puts global design constraints such as symmetry explicitly under artist control, interrelated with visually specified input such as strokes, down to control at the individual element level.

Our technique confirms the great potential of integrating the artistic control mechanisms that artists use every day into a procedural system. We hope it will inspire others to explore this direction further for example for the creation of the procedural models themselves.

IMAGE REFERENCES

[A] Manuscripts and Archives Division, The New York Public Library. 1450 - 1475. Historiated initial and another coat of arms. (1450 - 1475). http://digitalcollections.nypl.org/items/510d47da-e47a-a3d9-e040-e00a18064a99

[B] Owen Jones. 1867. *Examples of Chinese ornament selected from objects in the South Kensington museum and other collections.* London: S. & T. Gilbert. http://archive.org/details/examplesofchines00jone

[C] The Miriam and Ira D. Wallach Division of Art, Prints and Photographs, The New York Public Library. 1882. Valentine cards utilizing decorative design, depicting fowers, hearts, butterflies and a tree. (1882). https://digitalcollections.nypl.org/items/510d47db-bc92-a3d9-e040-e00a18064a99

[D] Spencer Collection, The New York Public Library. 1910. Front doubleur. (1910). http://digitalcollections.nypl.org/items/8a6be0f9-3d78-b15e-e040-e00a180602c7

[E] William Morris. 1883. Strawberry Thief Printed Textile. (1883). https://commons.wikimedia.org/wiki/File:Morris_Strawberry_Thief_1883_detail Source: Planet Art CD of royalty-free PD images - William Morris.

[F] Colourbox. 2011. Frame with roses, Vector. (2011). https://www.colourbox.com/vector/frame-with-roses-vector-1286656

[G] Colourbox. 2016. Big set of hand drawn oral elements, Vector. (2016). https://www.colourbox.com/vector/oral-elements-vector-14556631

[H] Colourbox. 2013. Illustration of frame in Ukrainian folk style, Vector. (2013). https://www.colourbox.com/vector/frame-vector-6826661

[I] Izabela Rejke. 2011. Traditional Polish Folk Design. (2011). http://rejke.deviantart.com/art/Traditional-Polish-Folk-Design-192417774

[J] Colourbox. 2013. Ornamental khokhloma oral postcard with seamless stripe, Vector. (2013). https://www.colourbox.com/vector/ornamental-khokhloma-oral-postcard-vector-8445572

REFERENCES

Dustin Anderson and Zoë Wood. 2008. User driven two-dimensional computer-generated ornamentation. In *Advances in Visual Computing*. Springer, 604–613. http://link.springer.com/chapter/10.1007/978-3-540-89639-5_58

B. Beneš, O. Šťava, R. Měch, and G. Miller. 2011. Guided Procedural Modeling. *Computer Graphics Forum* 30, 2 (2011), 325–334. DOI: https://doi.org/10.1111/j.1467-8659.2011.01886.x

Derek Bradley, Derek Nowrouzezahrai, and Paul Beardsley. 2013. Image-based Reconstruction and Synthesis of Dense Foliage. *ACM Transactions on Graphics* 32, 4, Article 74 (2013), 10 pages. DOI: https://doi.org/10.1145/2461912.2461952

Guoning Chen, Gregory Esch, Peter Wonka, Pascal Mueller, and Eugene Zhang. 2008a. Interactive Procedural Street Modeling. *ACM Transactions on Graphics* 27, 3 (2008), Article 103: 1–10.

Weikai Chen, Xiaolong Zhang, Shiqing Xin, Yang Xia, Sylvain Lefebvre, and Wenping Wang. 2016. Synthesis of filigrees for digital fabrication. *ACM Transactions on Graphics* 35, 4 (2016), 98.

Xuejin Chen, Boris Neubert, Ying-Qing Xu, Oliver Deussen, and Sing Bing Kang. 2008b. Sketch-based Tree Modeling Using Markov Random Field. *ACM Transactions on Graphics* 27, 5 (2008), 109:1–109:9. DOI: https://doi.org/10.1145/1409060.1409062

Yu-Sheng Chen, Jie Shie, and Lieu-Hen Chen. 2012. A NPR System for Generating Floral Patterns based on L-System. *Bulletin of Networking, Computing, Systems, and Software* 1, 1 (2012). http://bncss.org/index.php/bncss/article/view/7

Arnaud Emilien, Ulysse Vimont, Marie-Paule Cani, Pierre Poulin, and Bedrich Benes. 2015. WorldBrush: Interactive Example-based Synthesis of Procedural Virtual Worlds. *ACM Transactions on Graphics* 34, 4, Article 106 (2015), 11 pages. DOI: https://doi.org/10.1145/2766975

Katayoon Etemad, Faramarz F. Samavati, and Przemyslaw Prusinkiewicz. 2008. Animating Persian Floral Patterns. In *Proceedings of the Fourth Eurographics Conference on Computational Aesthetics in Graphics, Visualization and Imaging (CA'08)*. Eurographics Association, 25–32. DOI: https://doi.org/10.2312/COMPAESTH/COMPAESTH08/025-032

Takashi Ijiri, Radomír Měch, Takeo Igarashi, and Gavin Miller. 2008. An Example-based Procedural System for Element Arrangement. *Computer Graphics Forum* 27, 2 (2008), 429–436. DOI: https://doi.org/10.1111/j.1467-8659.2008.01140.x

Rubaiat Habib Kazi, Takeo Igarashi, Shengdong Zhao, and Richard Davis. 2012. Vignette: Interactive Texture Design and Manipulation with Freeform Gestures for Pen-and-ink Illustration. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*. ACM, New York, NY, USA, 1727–1736. DOI: https://doi.org/10.1145/2207676.2208302

Markus Lipp, Peter Wonka, and Michael Wimmer. 2008. Interactive Visual Editing of Grammars for Procedural Architecture. *ACM Transactions on Graphics* 27, 3, Article 102 (2008), 10 pages. DOI: https://doi.org/10.1145/1360612.1360701

Chongyang Ma, Li-Yi Wei, Sylvain Lefebvre, and Xin Tong. 2013. Dynamic Element Textures. *ACM Transactions on Graphics* 32, 4, Article 90 (2013), 10 pages. DOI: https://doi.org/10.1145/2461912.2461921

Chongyang Ma, Li-Yi Wei, and Xin Tong. 2011. Discrete Element Textures. *ACM Transactions on Graphics* 30, 4, Article 62 (2011), 10 pages. DOI: https://doi.org/10.1145/2010324.1964957

Ron Maharik, Mikhail Bessmeltsev, Alla Sheffer, Ariel Shamir, and Nathan Carr. 2011. Digital Micrography. *ACM Transactions on Graphics* 30, 4, Article 100 (2011), 12 pages. DOI: https://doi.org/10.1145/2010324.1964995

Radomír Měch and Gavin Miller. 2012. The Deco framework for interactive procedural modeling. *Journal of Computer Graphics Techniques (JCGT)* 1, 1 (2012), 43–99. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.462.6752&rep=rep1&type=pdf

Ken Nakagaki and Yasuaki Kakehi. 2014. Comp*Pass: A Compass-based Drawing Interface. In *CHI '14 Extended Abstracts on Human Factors in Computing Systems (CHI EA '14)*. ACM, New York, NY, USA, 447–450. DOI: https://doi.org/10.1145/2559206.2574766

Wojciech Palubicki, Kipp Horel, Steven Longay, Adam Runions, Brendan Lane, Radomír Měch, and Przemyslaw Prusinkiewicz. 2009. Self-organizing Tree Models for Image Synthesis. *ACM Transactions on Graphics* 28, 3, Article 58 (2009), 10 pages. DOI: https://doi.org/10.1145/1531326.1531364

Yoav I. H. Parish and Pascal Müller. 2001. Procedural Modeling of Cities. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '01)*. ACM, New York, NY, USA, 301–308. DOI: https://doi.org/10.1145/383259.383292

Sören Pirk, Ondrej Stava, Julian Kratt, Michel Abdul Massih Said, Boris Neubert, Radomír Měch, Bedrich Beneš, and Oliver Deussen. 2012. Plastic trees: interactive self-adapting botanical tree models. *ACM Transactions on Graphics* 31, 4 (2012), 1–10. DOI: https://doi.org/10.1145/2185520.2185546

Przemyslaw Prusinkiewicz. 1990. *The algorithmic beauty of plants.* Springer-Verlag, New York.

Przemyslaw Prusinkiewicz, Faramarz Samavati, Colin Smith, and Radoslaw Karwowski. 2003. L-system Description Of Subdivision Curves. *International Journal of Shape Modeling* 09, 01 (2003), 41–59. DOI: https://doi.org/10.1142/S0218654303000048

Daniel Ritchie, Ben Mildenhall, Noah D. Goodman, and Pat Hanrahan. 2015. Controlling Procedural Modeling Programs with Stochastically-ordered Sequential Monte Carlo. *ACM Transactions on Graphics* 34, 4, Article 105 (2015), 11 pages. DOI: https://doi.org/10.1145/2766895

Daniel Ritchie, Anna Thomas, Pat Hanrahan, and Noah D. Goodman. 2016. Neurally-Guided Procedural Models: Learning to Guide Procedural Models with Deep Neural Networks. *arXiv preprint arXiv:1603.06143* (2016).

O. Šťava, B. Beneš, R. Měch, D. G. Aliaga, and P. Krištof. 2010. Inverse Procedural Modeling by Automatic Generation of L-systems. *Computer Graphics Forum* 29, 2 (2010), 665–674. DOI: https://doi.org/10.1111/j.1467-8659.2009.01636.x

O. Šťava, S. Pirk, J. Kratt, B. Chen, R. Měch, O. Deussen, and B. Beneš. 2014. Inverse Procedural Modelling of Trees. *Computer Graphics Forum* 33, 6 (2014), 118–131. DOI: https://doi.org/10.1111/cgf.12282

Jerry O. Talton, Yu Lou, Steve Lesser, Jared Duke, Radomír Měch, and Vladlen Koltun. 2011. Metropolis procedural modeling. *ACM Transactions on Graphics* 30, 2 (2011), 1–14. DOI: https://doi.org/10.1145/1944846.1944851

Michael T. Wong, Douglas E. Zongker, and David H. Salesin. 1998. Computer-generated Floral Ornament. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '98)*. ACM, New York, NY, USA, 423–434. DOI: https://doi.org/10.1145/280814.280948

Ling Xu and David Mould. 2015. Procedural Tree Modeling with Guiding Vectors. *Computer Graphics Forum* 34, 7 (2015), 47–56. DOI: https://doi.org/10.1111/cgf.12744

Yuanyuan Li, Fan Bao, Eugene Zhang, Yoshihiro Kobayashi, and Peter Wonka. 2011. Geometry Synthesis on Surfaces Using Field-Guided Shape Grammars. *IEEE Transactions on Visualization and Computer Graphics* 17, 2 (2011), 231–243. DOI: https://doi.org/10.1109/TVCG.2010.36