EUROGRAPHICS 2021
H. Rushmeier and K. Bühler
(Guest Editors)

*Volume 40 (2021), Number 2*
*STAR – State of The Art Report*

# A Survey of Control Mechanisms for Creative Pattern Generation

Lena Gieseke[1], Paul Asente[2], Radomír Měch[2], Bedrich Benes[3], and Martin Fuchs[4]

[1]Film University Babelsberg Konrad Wolf, [2]Adobe Research, [3]Purdue University, [4]Hochschule der Medien

**Abstract**

*We review recent methods in 2D creative pattern generation and their control mechanisms, focusing on procedural methods. The review is motivated by an artist's perspective and investigates interactive pattern generation as a complex design problem. While the repetitive nature of patterns is well-suited to algorithmic creation and automation, an artist needs more flexible control mechanisms for adaptable and inventive designs. We organize the state of the art on the pattern design features they enable, such as repetition, frames, curves, directionality, and single visual accents. Within those areas, we summarize and discuss the techniques' control mechanisms for enabling artist intent. The discussion is led by the questions of how input is given by the artist, what type of content the artist inputs, where the input affects the canvas spatially, and when input can be given in the timeline of the creation process. We categorize the available control mechanisms on an algorithmic level and categorize their input modes based on exemplars, parameterization, handling, filling, guiding, and placing interactions. To better understand the potential of the current techniques for creative design and to make such an investigation more manageable, we motivate our discussion with how navigation, transparency, variation, and stimulation enable creativity. We conclude our review by identifying possible new directions that can inspire innovation for artist-centered creation processes and algorithms.*

**CCS Concepts**
• ***Computing methodologies*** → ***Computer graphics;*** • ***Human-centered computing*** → ***Interaction design process and methods; Interaction design theory, concepts and paradigms;*** *Systems and tools for interaction design;*

## 1. Introduction

Pattern generation in computer graphics provides a large amount of precise and quickly-generated digital content. However, despite more than three decades of research, supporting artists with meaningful digital tools for creative content generation, for example, those needed for ornamental pattern designs (Figure 1), is an ongoing challenge. Most solutions focus on singular features and control mechanisms, such as example-based controls or brushes, on an algorithmic level. Little attention, however, has been paid to overall creative workflows, which need to strike a balance, giving users needed power without burdening them with unwanted details. Often, techniques that are claimed to be artist-controllable turn out not to be so.

This may be a consequence of research often being executed without direct and continuous collaboration with artists and without the support of large-scale user studies. Algorithms and methods are often developed to further the state of the art, such as building on recent progress in deep learning research, with little understanding of artists' needs and how mechanisms can be validated.

Recent surveys have covered 3D generation in great detail, focusing on world building [STBB14; ADBW16], terrain modeling [GGP*19] and game-specific approaches [HMVI13; TYSB11]. In this report, we review recent advances in 2D pattern generation.

The underlying regularity of 2D pattern designs is based on a repetitive and balanced distribution of elements, usually following hierarchical structures. These characteristics can be efficiently implemented by procedural approaches that arrange elements in space according to generative rules [ŠBM*10], but developing these rules can be tedious, non-inspiring, repetitive, and challenging. The primary motivation for inverse procedural models is to free the artist from these tasks. Computational generation techniques can ease these problems and also perform in a more precise and less error-prone way than a human artist. However, the creative demands of tasks like laying out space-specific designs and placing visual accents must also be considered. Procedural models must be augmented and different approaches must be unified to combine the control and quality of manual creation with the efficiency and accuracy of computation [GALF17].

We review recent advances in 2D pattern generation and discuss procedural models, data-driven generation, and design-specific pattern generation. As theoretical grounding, we classify control mechanisms and their characteristics from the perspective of an artist, from global to local and from automatic to manual, with varying levels of abstraction. As a basis for a discussion of the ca-

**Figure 1:** *Historic examples for creative pattern designs and ornamentation. Places of origin from left to right, top to middle row: France, China, USA, UK, Poland, Egypt, UK. Bottom row: recent commercial examples. This figure is adapted from [GALF17], image sources: [1-11].*

pabilities of control mechanisms in the creative process and their potential for innovative creation, we review the literature on creativity and summarize aspects that can help to understand these capabilities in the context of computer graphic publications.

We organize contemporary techniques by design areas and the visual features they enable. We further group related work by commonly-used control types. Then, we specifically analyze the artistic controls offered by each technique. We conclude the review with a discussion of the means for enabling creativity for the different control mechanism types. With this survey, we hope not only to meaningfully categorize and summarize the state of the art but also to contribute to a shared vocabulary and to establish a foundation for incorporating artists and their creative tasks into algorithmic content creation pipelines.

## 2. Terminology

The following describes how we will be using terms relevant to our topic.

**Pattern** is a generic term for any type of repeated, often regular, arrangement [Oxf17].

**Artistic** refers to a task with an outcome that potentially has meaning and value beyond aesthetics and practicality. In addition to formal skills that depend on a given domain, an artistic task usually requires creative thinking as well as intuition, emotion and sensual considerations, for example.

**Creative** refers to a task that intentionally produces a novel, non-standard outcome, as further discussed in Section 4.

**Design space** refers loosely to the range of visual results a technique can create. For example, Perlin noise [Per85] has a rather restricted design space of noise images, only differing in few characteristics. Drawing with a pen can result in many different designs, thus resulting in a larger design space.

**Expressiveness** refers to the size, the variability and the openness of a design space as in detail discussed in Section 4. Expressiveness is commonly used in the context of creative controls, but often without a clear understanding of its meaning.

**Canvas** constitutes the area in which the output is generated, similar to a canvas in a painting context.

**Shape** refers to the external boundary or outline on the canvas or of an object without any restrictions on the form.

**Procedural** refers to the production of output by running an algorithm, such as a rule-based system.

**Data-driven** is the production of output based on given data.

**Parameterized** refers to offering separately controllable characteristics. It can be used with procedural pattern representations, or data-driven approaches.

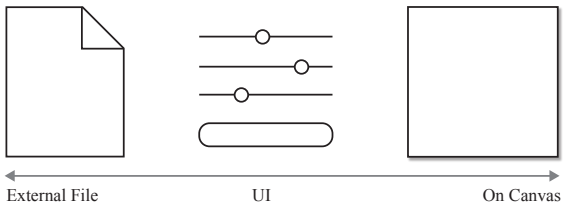
## 3. Taxonomy of Control Mechanisms

The following taxonomy lays groundwork for our later evaluation of control mechanisms of the state of the art. It is difficult to derive the discussion of means for enabling creativity directly from the related work, as its authors have followed different motivations and have emphasized various aspects when describing their work and results. To classify the work in an objective and unified manner, we analyze general characteristics of control with digital creation tools and relate the actual presented control mechanisms to them. Based on this analysis, we then review the related work in Section 7.

### 3.1. Control Characteristics

A creation process can be described by answering the questions of *how*, *what*, *where*, *when* and *who*. These characteristics can be discussed in various creation contexts and could even be translated to traditional media such as aquarell on paper.

**How (User Interaction)** is a control executed or an input given by an artist? How detached is it from the visual result on the canvas?

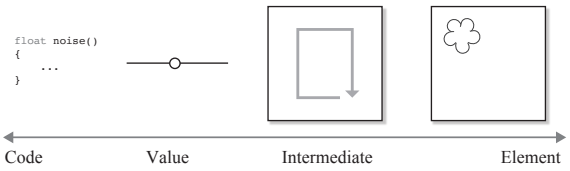



- *File*: The control is externally given, such as with code or a configuration file.
- *Separate UI*: A separate UI is given through which an artist gives input and activates states. Separate UIs are often in close proximity to the canvas, carefully designed and easily usable. However,

because they detach the work from the actual output, separate UIs have a level of indirection. An artist must actively translate the interaction with the separate UI to the resulting output on the canvas.

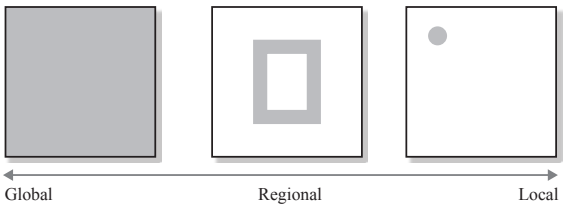
- *On canvas*: Controls are executed directly on the output canvas. Most of them require activating or selecting a tool in a separate UI, such as selecting a pen for drawing on a canvas. There are cases where controls cannot clearly be classified as either separate UI or on canvas. A pen, for example, can have different characteristics that an artist needs to set in the UI. The adjustment of settings should be as seamlessly integrated into an on-canvas tool as possible (e.g., with choices appearing as tool tips).

**What (Content)** does an artist give as input? What is the level of abstraction of the content that an artist works with?



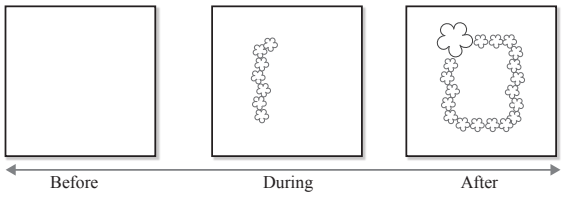

- *Code*: Input is a syntactically structured formal language.
- *Value*: The input is a single value, chosen from a range - for example, with a slider.
- *Intermediate*: The input is visual but abstract, such as sketching a mask or arrows for directionality. Artists still have to interpret how the input affects the result.
- *Element*: The input is a component of the resulting pattern.

**Where (Canvas)** does the input have an effect and what is the area of influence?




The input can have *global* influence, *regional*, e.g., on a drawn curve or *local*, e.g., on one specific element.

**When (Timeline)** is the input given and at what time in the creation process is the control executed?




Input can be given *before* the creation process, *during* it when parts of the results are already visible, or *after*, when the result is visible and can be adjusted retrospectively.

**Who (User)** describes the skill set needed to provide the input.


This category can be in part derived from the above characteristics of *how* and *what*. In most general terms, this category can be classified as the skill set of a *Programmer*, for giving, e.g., code as input, requiring analytical-formal and logical thinking and the ability to abstract. On the other end of the spectrum, the skill set of an *Artist* is needed, e.g., for drawing, requiring intuitive-visual and spatial thinking and the ability to create. The *who* category is listed here for completeness. The discussion of needed competencies, skills and mindsets, including the accompanying psychological and artistic aspects, is out of the scope of this survey.

### 3.2. Control Mechanisms

We now classify the control mechanisms as they are described in the state of the art. These low-level mechanisms define what an artist can or must work with and are specified for each reference in Table 1. Because this survey focuses on algorithms, UI specifics, such as the layout of buttons, are not considered. For each mechanism we summarize the *how*, *what*, *where* and *when* characteristics over all reviewed publications and with that show in Table 3 connections and capabilities of different control mechanisms and potential trade-offs between approaches.

#### Initialization

- *System Configuration*: The required overall setup of the system, such as computing caches or training a model. This is usually a one-time investment.
- *Task Initialization*: A non-creative task that has to be executed each time in order to produce an output, such as selecting the specific optimization algorithm.

#### Exemplars

- *Image*: An example image that should be matched in its entirety. Examples are usually pixel data.
- *Element Arrangement*: An example element arrangement that should be matched in its entirety. Elements are usually separate shapes and might carry additional data.
- *Element*: One specific, individual asset that becomes in the result part of a whole. Elements can be shapes or pixel data.

#### Parameterization

- *Visual Output*: Parameters that can adjust visual features directly in the output.
- *System/Generation*: Parameters that influence the output indirectly, such as parameters for an optimization algorithm or constraints.

#### Handling

- *Visualization*: Any type of visual interface that goes beyond the standard UI elements, such as sliders and buttons.

- *Image-Based*: Images as indirect control input, such as pixel data masks.
- *Sketch-Based*: Sketches and curves defining indirect control elements on the canvas—for example, the drawing of a mask with a pen tool.

**Filling**

- *Shapes*: A space to fill, e.g., a specific shape).
- *Masking*: Areas within the shape to fill that should remain unaffected.
- *Curves*: A one-dimensional curve or path to be followed with elements of the pattern. The curve is given as a whole before the generation starts.

**Guiding**

- *Brushing/Strokes*: A curve, usually created by mouse movements or with a stylus pen, that is followed with output elements—often understood as brushing.
- *Directions*: Visual elements such as intermediate curves, arrows or output components that define directions for the design to follow (e.g., with an underlying vector field).

**Placing**

- *Element Placement*: The direct placement of components on the canvas as part of the final result.
- *Element Drag & Drop*: Drag and drop of components on the canvas within the existing result.

## 4. Means For Enabling Creativity

Many content generation tasks in computer graphics involve creative considerations. Considering creativity is especially relevant for investigating control mechanisms because the controls provide the means for creative creation. However, creativity is ill-defined and involves insights from various disciplines, making it a notoriously difficult topic to address.

On the one hand, there are efforts to develop algorithms that perform creatively themselves. On the other hand, there is the goal of supporting human creativity with digital tools, which is the focus of this survey. Enabling human creativity is an established field in the human-computer interaction (HCI) community and it is based largely on the pioneering work of SHNEIDERMAN [Shn07] about *Creativity Support Tools*. This topic has recently found renewed attention [FMD18; FMR*19; RMF*20], asking for more rigorous evaluation in regard to claimed creativity support. Among other aspects, FRICH et al. [FMD18] ask for "clearer definitions of creativity". This shows that the HCI community already strives for the type of survey this STAR proposes. However, for this survey, rooted in the field of computer graphics, we review potentially relevant characteristics of underlying algorithms that enable artists to be creative. We include interface design aspects but they are not the focus of this survey. However, the need to bridge between developing the core algorithms and the interface in order to support creative and artistic intent has been voiced by past research [DHF*17; Ise16; Sal02], and this work contributes to this effort.

CHERRY et al. [CL14] presented an evaluation technique called the quantifiable *Creativity Support Index* (CSI), which has found its way into the graphics community [SLD17]. The index measures how well a tool enables creativity based on a psychometric survey. The development and validation of the measurement dimensions—namely, *exploration*, *expressiveness*, *immersion*, *enjoyment*, *results worth effort*, and *collaboration*—are based on user tests. CHERRY et al. [CL14] quantified the specific phrases participants used to describe a creative process. However, a clear definition of terms like *exploration* and *expressiveness* is missing and the meaning of a statement such as, "I was able to be very creative," is left open.

A quantified user study is often not feasible, for example for this survey and for assessing the state of the art. Furthermore, doing so would not always be meaningful because supporting creativity is not a goal for most methods. However, most methods do offer carefully developed control mechanisms. We propose a discussion of how the control mechanisms of each technique, as presented by its authors, enable creativity. Based on the given information, we reflect on the potential for the means for enabling creativity in a meaningful way, even if creative control was not necessarily the authors' intention.

We derive a working basis of such means for enabling creativity in regard to control mechanisms in the following review of relevant references. Our classification is meant as a step toward understanding the creative control options within the current state of the art. In terms of measurement dimensions, it can be seen as an evaluation on algorithmic level and a subset of the more general and user-study-based classification with the *Creativity Support Index*.

### 4.1. Discussion Basis

WEISBERG [Wei06] states that a creative person "intentionally produces a novel product" (p.70) and explicitly decouples a possible value of a product from it being the result of a creative process (p.63). BODEN [Bod10] described *novelty* as a surprising product, one that the creator did not directly anticipate (p.30).

The integration of *intention* in describing a creative process is crucial for the development of meaningful algorithms. Weisberg explains that a painter who accidentally stains a painting—a stain that is later applauded by the art world as an innovative technique—cannot be considered to have achieved a creative result. Hence, algorithms need to enable artists to follow their intentions with transparent and controllable mechanisms.

Weisberg argues that a creator needs domain-specific knowledge and expertise in order to come up with something novel or surprising. He rejects the common perception of creativity as being an "unfathomable leap of insight" and advocates its systematic accessibility.

Weisberg's argument applied in the context of control mechanisms leads to requiring techniques to enable artists to fully understand the domain they work with. Cause and effect of interactions as well as the overall options to control the output must be transparent and navigable.

However, there also must be a large design space for a creator to explore. Such an increase of possible options is a core aspect of many common creativity techniques, such as brainstorming, and must also be used for the development of digital tools [TMNY04].

Design spaces need to be big but also meaningful and well-framed for the domains they represent. They must provide space to delve into without the danger of getting lost. Therefore in a system that offers design options, all options need to make sense, while "enabling someone to see possibilities they hadn't glimpsed before" [Bod10]. For the discovery of unseen results, various stimuli can be given, for example well-designed constraints.

The target audience is also an influencing factor in enabling creativity. Each skill level requires its own unique type of support [CL14]. In the following discussion, this survey does not explicitly investigate the appropriateness of a technique for different skill levels unless it specifically distinguishes itself from a related work, as, for example, in BENEDETTI et al. [BWCS14]. Instead it focuses on the general suitability of a technique to create the design goal with a reasonable training curve for an average artist. Also, for this first investigation, we focus on support for a single artist, but it is worth mentioning that enabling collaborative creative work would be a meaningful aspect to consider in the future.

To sum up the brief review in the context of control mechanisms, we consider the categories *navigation*, *transparency*, *variation* and *stimulation* in the context of control mechanisms. We understand variation as the size of the design space within the context of the technique. For the exploration of different designs we distinguish between the general controllability necessary for navigating a design space ("there are many different roads in the landscape"), and the transparency of that navigation and the understanding of cause and effect when using the tool ("I have the map to the landscape and know how to get from one point to another"). However, at this point, these categories can be seen as somewhat loose and experimental, aiming toward a better understanding of requirements for creative controls. For these characteristics, there is no clear translation into quantifiable metrics, such as timings or error rates, which are standardized measurements for productivity [CL14; Shn07].

**Navigation**

The means of navigation describe whether a creation processes is efficiently manageable as well as the extent of the controllability.

- *Interactive:* Refers to the lack of noticeable delays when executing controls and computing results. Lengthy, non-creative configuration requirements are also potentially distracting. Hence, a thorough analysis should consider the whole process an artist has to go through to produce a result.
- *Number of Controls:* Indicates how flexible and controllable a technique is, e.g., by counting the number of different controls that can be adjusted for one output. Ideally, this category would refer to the ratio of visual features of the possible output that are relevant to humans to controllable features. This would ensure that the controls cover all necessary features and that they complement each other. However, the identification of generally describable, perceptually relevant visual features is out of the scope of this survey and left to future work.
- *Navigation History:* Describes the ability to go back and forth in one's own creation process, such as using an eraser.

**Transparency**

The means of transparency describe how clear the understanding of cause and effect within the system are.

- *Control Domain:* Refers to how well controls are mapped to visual features and how well they cover the possible design range of each feature. A high-quality control should not have any overlapping effects with other controls.
- *Control Communication:* Describes how well controls (e.g., with a visualization and/or little abstraction) represent their effects on the result. For artist-centered tools this could mean that controls should be visual and directly on the canvas.

**Variation**

Variation indicates how visually different the results can be.

- *Size of the Design Space:* This is limited if all results look rather similar to each other and are part of a specific design class. A large design space of one technique allows, for example, combining different texture classes such as stochastic and structural creation.
- *Openness of the Design Space:* Refers to the limitlessness of possible designs and that there is no attachment of the technique to a specific design class. An open design space enables an artist to come up with a distinctive individual style. Different artists can create inherently different and unique results with the same tool if it has a open design space.

We do understand that a clear definition of the available different design classes is needed. However, these solely depend on the design task.

**Stimulation**

The means of stimulation indicate how well an artist can enter a pleasurable and stimulating workflow.

- *Immersion:* How natural and enjoyable the usage of a system feels. An immersive technique needs navigation to be fluent, controls to be intuitive, and the design space large enough to not to hit its boundaries while using the tool.
- *Stimuli:* The support to find surprising results—for example, with design suggestions or variations of the input. Options to support stimulation are still underrepresented but on the rise with machine learning techniques. A clear definition of this category is not feasible at this point.

In summary, for handling the ill-defined topic of creativity, we follow the definition of creativity as intentionally producing a novel and surprising product. We derive the means for enabling creative control from recent research results and relate them to control mechanisms. We hope to further a more objective judging of the ability of a technique to support creativity and a detailed comparison of methods and give an first example for that in the following review of the state of the art.

Various aspects of our discussion guides still leave room for interpretation. Knowledge from other disciplines, for example in regard to the perception of visual features, can contribute with valuable insights. We hope that our work inspires such research towards a quantifiable analysis of creative control.

## 5. Design Features

Creative patterns include repetitive and ordered structures that are often considered as *textures*, thus demanding automatic and procedural creation. On the other hand, creative pattern generation might

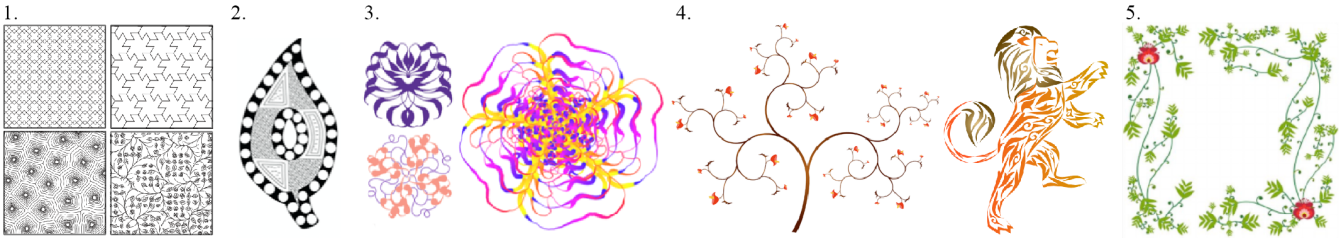



**Figure 2:** *Patterns showcasing the design features 1. Distribution and Repetition, 2. Frames and Hierarchies, 3. Curves, Lines and Brushing, 4. Connections, Branches and Directionality, and 5. Single Accents. Sources, from left to right: [LHVT17], [SP16], [JBMR18], [GJB*20], [SKAM17], [GALF17].*

also use a global layout, adapting to the space it is filling. Furthermore, these patterns often include visual hierarchies and highlights that were placed with creative intent.

Specifically, we categorize the analysis of the state of the art in Section 7 with the design features *Distribution and Repetition*; *Frames and Hierarchies*; *Curves, Lines and Brushing*; *Connections, Branches and Directionality*; and *Single Accents* for creative pattern generation. Here we briefly explain each design feature and give a visual example in Figure 2 for each from the related work.

**Distribution and Repetition** refers to an overall distribution of elements and usually results in a homogeneous pattern with a texture-like quality. A careful composition of the repeating elements can be used to create a perception of balance and order. Compositions are not limited to the repetition of the same element, but different visual qualities such as size or saturation can create various relationships.

**Frames and Hierarchies** form compositions that further structure a design by creating contrasts, e.g., foreground vs. background, and accentuating spatial relationships, e.g., framing.

**Curves, Lines and Brushing** refer to artist-defined curved elements, which are an essential design feature for many creative pattern designs. They can be used, for example, as a base element or frame as well as a distinct visual element. Brushing usually gives an individual, hand-made quality to a pattern.

**Connections, Branches and Directionality** between base elements are often used to further emphasize frames and hierarchies. These structures and directionality are also often used to elaborate and accentuate the form of the space they fill, e.g., by aligning to it, building an pattern-space relationship [ABS06].

**Single Accents** refer to visually dominant elements and structures that might not follow the underlying order of a pattern, breaking an otherwise homogeneous appearance. This design feature is based on the principle of contrasts and accents, which is crucial for the overall visual appeal of a creative pattern design [War96; WZS98; MOT99].

For the analysis of the state of the art, we sort all work according to the above-discussed design areas and with that by the visual features they enable. Within those design areas we then discuss the control mechanisms.

## 6. Models

In the context of computer graphics, generation techniques are usually differentiated into procedural and data-driven approaches. This understanding applies equally to the generation of geometry, animations and texture. Procedural techniques describe the visual output by evaluating an algorithm, while data-driven approaches rely on existing data, such as photographs. However, as the field has developed, the approaches have begun to blend and their advantages have been combined.

This survey focuses on procedural models, but it also integrates and highlights promising or desirable characteristics of suitable data-driven techniques.

### 6.1. Procedural

EBERT et al. [EMP*02] describe procedural techniques as algorithms and mathematical functions that synthesize a model or an effect. Representations based solely on equations are considered the "purest" form of procedural modeling [STBB14]. This approach gained immediate importance in the early days of computer graphics. Analytical methods were able to reproduce natural phenomena such as wood, stone, water, smoke and plants, using only a small amount of code, hence being memory efficient. The main appeal includes compactness combined with with being continuous, scalable and unbound to a specific resolution.

The compactness and efficiency of a procedural model also enable parameterization, resulting in the model being responsive and flexible. Parameters usually represent certain visual characteristics and their prominence. Parameterization brings the crucial benefit of textures remaining editable throughout an entire visual effect production pipeline.

However, the effectiveness of traditional parameterization in helping an artist fulfill design goals is debatable. EBERT et al. [EMP*02] argue that parameterization brings the benefit of a few parameters controlling many details. At the same time, this is potentially problematic for the realization of specific designs because these often require full individual control of all visual elements. Additionally, parameters are often non-intuitive due to representing overly abstract characteristics of the underlying functions and having overlapping effects [BD04; LVLD10; GD10; BŠMM11; LLD12b; LLD12a].

In addition to difficulties in controlling a procedural representation, creating the procedural model itself requires considerable effort, even though it is only a one-time investment. They require translating a visual design into a technical representation and generalizing it. For procedural textures specifically, handling anti-aliasing efficiently can also be challenging. EBERT et al. [EMP*02] includes a valuable and in-detail survey of function-based design principles of procedural models with a focus on textures.

Procedural models are not limited to purely function-based designs. For example, the pioneering work of PRUSINKIEWICZ [Pru90] applies the grammar-based L-system to algorithmically model plant growth, an approach extensively investigated by the computer graphics community.

The classifications of core mechanisms for procedural generation of HENDRIKX et al. [HMVI13] in the context of games and the categorization of SMELIK et al. [STBB14] for virtual worlds are equally appropriate in the context of creative pattern generation. In the following we briefly discuss core mechanisms in the context of creative pattern generation, based on the previously mentioned taxonomies ([HMVI13], [STBB14]).

**Stochastic Models** generate models by using random values. They can either be used in their original form as procedural models or as noise basis functions. Visual features can be added by combining multiple layers of noise in different resolutions. Typical noise functions are lattice value noise, lattice gradient noise (e.g., Perlin noise [Per85]), sparse convolution noise, and spectral noise [EMP*02; LLC*10]. In the context of creative pattern generation, stochastic models build a basis for many designs but their design spectrum and controllability are limited.

**Function-based Models** extend the class of stochastic models by layering and combining a variety of functions to form a visually complex pattern. Typical building blocks are periodic, spline, step, clamp and conditional functions [EMP*02] and are the basis for regular patterns designs.

**Rule-based Models** are part of often quite complex, generation systems that can be context-dependent and/or design-specific. Rule-based models are programs that relate to and partition the space to fill and follow propagation rules. The algorithmic core often handles proxy shapes, while for the result graphical elements, such as vector graphics, are mapped to the proxies. Rule-based procedural models are suitable for creative pattern generation and novel control mechanisms because their iterative generation logic is open and flexible [WZS98; MM12]. They can implement any designs and include any elements. Moreover, within a suitable pipeline, they can potentially take global constraints into consideration and build structural hierarchies.

**Grammar-based Models** form grammatically-correct sentences from individual words, based on a system of rules, and they are related to rule-based models in that the rules are using grammars or rewriting systems. Prominent techniques are L-systems and shape grammars. An emerging subgroup of grammar-based models incorporates *probabilistic* inference into the derivation of correct sentences from a grammar. In recent years, there have been a variety of successful grammar-based approaches for certain aspects of creative pattern generation [BŠMM11; TLL*11; RMGH15]. However, grammars are difficult to set up and to design [ŠBM*10]. Because the execution process is inherently hierarchical, grammar systems have difficulty supporting creative control from a global to local scale.

**Artificial Intelligence Models** represent approaches that go beyond the direct execution of specific rules. For example, they automatically optimize results based on fitness or error functions, or they apply planning steps. HENDRIKX et al. [HMVI13] group this class into genetic algorithms, neural networks and constraint satisfaction and planning. In recent years, machine learning has been introduced into procedural content generation with the same impact as in all other computer science fields [SSG*17]. The potential of machine learning techniques in regard to creative pattern generation and their control mechanisms seems almost limitless.

### 6.2. Data-Driven Models

In contrast to procedural techniques, data-driven methods can be used in two ways in the context of creative pattern generation. First, they describe the processing of limited data, such as the pixels from a photograph. Second, they refer to the output of a method, which is again limited, such as pixel data. Data-driven models traditionally do not include underlying design models, as procedural representations do. Consequently, data-driven approaches are flexible in terms of possible designs and can achieve photorealism by processing real photographs.

At the same time, photographs often include visual features such as illumination effects that are unwanted and difficult to remove. Moreover, further down a production pipeline, pixel data is usually no longer editable. Working with high-resolution images leads to high memory requirements, and without additional algorithms, data is fixed to a given resolution and scale.

Addressing the issue of resolution in example-based synthesis is a well-established field of research that aims to create an infinite amount of pixel data based on a given exemplar. The pyramid-based texture synthesis of HEEGER et al. [HB95] is an early famous example. WEI et al. [WLKT09], as well as BARNES et al. [BZ17] present comprehensive summaries of such example-based texture synthesis techniques, discussing statistical feature matching, neighborhood matching, patch-based and optimization methods. Overall, example-based methods for texture synthesis have achieved similar results in data size, random accessibility and editing and resolution options as procedural textures - but only within specialized contexts and not in an unified manner. Procedural textures offer these capabilities as inherent and combined characteristics.

Data-driven models are numerous and diverse because they need no underlying procedural model. In the following survey of the state of the art for creative pattern generation, we include various data-driven techniques, however an overall classification is not the focus of this work. Relevant techniques include the tiling and distribution of elements and drawing and brush mechanisms.

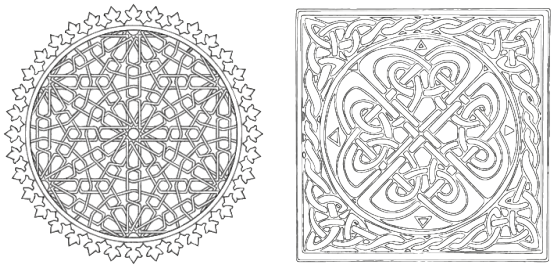


**Figure 3:** *Examples of traditional Islamic (left) and Celtic pattern designs, showing the complexity of possible pattern designs. Image sources: [12, 13].*

### 6.3. Specific Pattern Designs

In this section we review models that output specific pattern designs, which could be the basis for creative pattern generation. The references in this section summarize work that solely focuses on the output, offering little control or support for a creative creation process. Work that produces similar aesthetics but also offers control mechanisms (e.g., [WZS98; CSC12; ZCT16]), is reviewed in Section 7.

Work on the generation of pattern designs is spread over various research communities [Whi10]. For the development of models WHITEHEAD [Whi10] differentiates between two motivations in the context of generating models for ornamentation in games. First, the author identifies the goal to reproduce existing patterns such as Islamic and Celtic designs. Such work is mainly found in the communities of mathematics and computer science. Second, WHITEHEAD identifies the goal of generating novel pattern designs, which is held mainly by algorithmic computer artists. Such designs are usually not executed in an academic context and, beyond the presentation of the results, are unfortunately not well documented. Only a few exceptions, such as the work of TAKAYAMA [Tak16] in 3D-printed ornate shapes or ALVAREZ et al. [AMM19] in randomized abstract texture designs give much information about their underlying algorithms.

For what WHITEHEAD [Whi10] calls mathematical/scientific ornamentation in the context of games, the author describes tiling and symmetry as the most relevant constituting rules. Combined with interlacing parts of the pattern while repeating and tiling elements, these principles are able to systematically describe Islamic [Ost98] and Celtic [Cro93] patterns, moving towards the traditional designs given in Figure 3.

The seminal work of KAPLAN et al. [KS04] presents an algorithmic representation of Islamic star patterns, a topic of ongoing interest [KB18]. Readers further interested in this line of work are referred to the extensive investigation of KAPLAN [Kap02]. ETEMAD et al. [ESP08] and HAMEKASI et al. [HS12] also focus on Islamic flower patterns. Celtic designs were also successfully computed by KAPLAN et al. [KC03] and DOYLE and SEMWAL [DS13].

In addition to Islamic and Celtic designs, a variety of other pattern designs have been algorithmically formalized, such as Gothic window tracery [HF04], M. C. Escher patterns [DLW81; KS04], woodwork [GTS10; GSK12], optical illusions [CYZL14], mazes [PS06] and tile-based patterns [OZH15; Gda17].

## 7. Analysis of the State of the Art

This survey analyzes the control mechanisms in the state of the art for creative pattern generation from the perspective of an artist. Techniques are evaluated as a whole, including required configuration input, what they can create, and performance times. The analysis of the control mechanisms is directly taken from the authors' descriptions.

The analyzed works are categorized by design features they can create. The categories are *Distribution and Repetition*; *Frames and Hierarchies*; *Curves, Lines and Brushing*; *Connections, Branches and Directionality*; and *Single Accents* (Section 5, Figure 2). Within those design areas we investigate how an artist can create such designs and how control mechanisms are clustered for each design feature. Common clusters include example-based, field-based, and data-driven control mechanisms. In Table 1 all design features and discussed publications are cross-linked to the following sections.

If a work belongs to several design categories, it is discussed in detail in the most fitting area and then briefly referenced in other applicable areas. Techniques are considered procedural unless indicated otherwise. As it is the nature of procedural generation to automatically fill a space upon execution, we include the shapes category for all procedural systems in Table 1 even though this might not be mentioned in a publication. If a publication does not specify how the shape to be filled is provided, we consider the control to be code given by a file.

### 7.1. Distribution and Repetition

Designs with a repetitive pattern and a distribution of elements can be viewed as texturing methods. In the following, we further differentiate between stochastic, regular-to-near-regular, and design-specific patterns, as well as element arrangements. Texture generation mainly focuses on creating a repetitive and homogeneous pattern as automatically as possible. These methods usually provide only part of the design space and controllability that is needed for creative pattern generation. They are applicable to sub-parts with a texture-like quality, such as background regions and fills. Because texturing has been the driving force behind much of the development of procedural representations, it produced manifold approaches and noteworthy control mechanisms. Throughout this section, we identify clusters of example-based, field-based, probabilistic interference, and data-driven methods for the design feature of *Distribution and Repetition*.

#### 7.1.1. Stochastic Pattern

Stochastic textures have been the foundation of both research investigations and many complex models. They are generated with noise functions, and LAGAE et al. [LLC*10] present the state of the art for work before the year 2010. Figure 4 shows a typical visual appearance for a stochastic pattern. For controlling the textures, the authors identify three main approaches: indirect access to the noise through controlling the power spectrum (also shown

| | INIT. | | EXEMPLARS | | | PARAM. | | HANDLING | | | FILLING | | | GUIDING | | PLACE. | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Configuration | Initialization | Image | Arrangement | Element | Visual Output | System | Visualization | Image | Sketch | Shapes | Masking | Curve | Brushing | Directions | Element | Drag & Drop |
| *DISTRIBUTION AND REPETITION* | | | | | | | | | | | | | | | | | |
| *Stochastic Patterns* | | | | | | | | | | | | | | | | | |
| GALERNE et al. [GLLD12] | | | × | | | × | | × | | | × | | | | | | |
| GILET et al. [GDG12a] | | × | × | | | × | | | × | | × | | | | | | |
| GILET et al. [GSV*14] | | | × | | | × | × | | × | | × | | | | | | |
| PAVIE et al. [PGDG16] | | | | | × | × | × | | × | | × | | | | | | |
| GUINGO et al. [GSDC17] | | | × | | | × | × | | × | | × | | | | | | |
| KANG et al. [KH17] | | | × | | | × | | | × | | × | | | | | | |
| GILET et al. [GD10] | | × | × | | | × | | | | × | × | | | | | | |
| *Regular to Near-Regular Patterns* | | | | | | | | | | | | | | | | | |
| LEFEBVRE et al. [LP00] | | × | × | | | × | | | × | | × | | | | | | |
| GILET et al. [GDG12b] | | | | × | × | × | | | | | × | | | | | | |
| BOURQUE et al. [BD04] | × | × | × | | | × | | | | | × | | | | | | |
| GIESEKE et al. [GKHF14] | × | × | × | | | × | × | | | | × | | | | | | |
| HU et al. [HDR19] | × | × | × | | | × | | | | | × | | | | | | |
| GUEHL et al. [GAD*20] | | × | × | | | × | × | × | | | × | | | | | | |
| BIAN et al. [BWL18] | | | × | | | | | × | | × | × | | | | | | |
| LI et al. [LBMH19] | | × | | | | × | × | | | | × | | | | | | |
| TU et al. [TWY*20] | | | | × | | | × | × | | × | × | | | | | × | × |
| ZEHNDER et al. [ZCT16] | | | | | × | | × | × | | | × | | | | | × | × |
| CHEN et al. [CZX*16] | | × | | | | × | | | | | × | × | | | × | | |
| *Rule- and Design-Specific Patterns* | | | | | | | | | | | | | | | | | |
| WONG et al. [WZS98] | × | | | | | × | | | | | × | | | | | | |
| LOI et al. [LHVT17] | × | | | | | × | | | | | × | | | | | | |
| TALTON et al. [TLL*11] | × | | | | | × | | | × | | × | × | | | | | |
| RITCHIE et al. [RMGH15] | × | | | | | × | | | × | | × | × | | | | | |
| LI et al. [LBZ*11] | × | | | | | × | | | | | × | | | | × | | |
| ŠT'AVA et al. [ŠBM*10] | | × | | × | | × | × | | | | × | | | | | | |
| TALTON et al. [TYK*12] | | × | | × | | × | × | | | | × | | | | | | |
| *Element Arrangements* | | | | | | | | | | | | | | | | | |
| BARLA et al. [BBT*06] | | × | | × | | × | | | | | | | | | | | |
| HURTUT et al. [HLT*09] | | | | × | | | | | × | | × | | | | | | |
| IJIRI et al. [IMIM08] | | | | × | | × | × | | | | × | × | | × | × | | |
| MA et al. [MWT11] | | × | | × | | × | × | | | | × | | | | × | | × |
| ALMERAJ et al. [AKA13] | | | | × | | | | | | | × | | | | | | |
| LANDES et al. [LGH13] | | | | × | | × | × | | × | | × | | | | | | |
| SAPUTRA et al. [SKAM17] | | × | | | × | | | | | × | × | × | | | × | | |
| SAPUTRA et al. [SKA18] | | × | | | × | | | | | | × | | | | | | |
| *FRAMES AND HIERARCHIES* | | | | | | | | | | | | | | | | | |
| ANDERSON et al. [AW08] | | | | | × | × | × | | | | | × | × | | | × | |
| BENES et al. [BŠMM11] | × | | | | | × | | × | | | × | × | | | | | |
| SANTONI et al. [SP16] | | | | | | × | × | | | × | × | × | | | | | |
| GIESEKE et al. [GALF17] | × | × | | | | × | | | × | × | × | × | × | | × | × | × |
| *CURVES AND BRUSHING* | | | | | | | | | | | | | | | | | |
| CHEN et al. [CSC12] | | | | | | | | | | | | | × | | | | |
| XU et al. [XM09] | | × | | | | | × | | | | × | | × | | | | |
| MERRELL et al. [MM10] | | | | | × | | | | | | | | | | | | × |
| MĚCH et al. [MM12]* | × | | | | | × | × | | | × | × | × | × | × | | | |
| HSU et al. [HWYZ20] | | | | × | × | | | × | | | × | × | | × | × | | |
| JACOBS et al. [JBMR18] | | × | | | × | × | × | × | | | | | | × | | × | |
| LU et al. [LBW*14] | | × | | | × | × | | | | × | | | × | | | | |
| ZHOU et al. [ZJL14] | | × | | | × | × | × | | | | × | | × | | | | |
| KAZI et al. [KIZD12] | | | | | | × | × | | | × | × | | × | × | × | × | |
| XING et al. [XCW14] | | | | | | × | × | × | | × | × | | × | × | × | × | |
| *CONNECTIONS, BRANCHES AND DIRECTIONALITY* | | | | | | | | | | | | | | | | | |
| GUO et al. [GJB*20] | × | | × | | | × | × | | | | × | | | × | | | |
| *SINGLE ACCENTS* | | | | | | | | | | | | | | | | | |
| YEH et al. [YM09] | | | | × | | | | | | | | | | × | | × | × |
| GUERRERO et al. [GBLM16] | × | | | × | | | | × | | | | | | | | × | × |

**Table 1:** *Recent techniques are sorted by design areas and visual features they enable. For each work it is analysed and indicated which specific control mechanisms they offer. *Please note that [MM12] present a procedural modeling engine, which in principle can be programmed to include almost any control type.*

in Figure 4), direct access to its appearance through function parameters, and example-based techniques. The first two approaches are based on specific function characteristics and are hardly generalizable for creative pattern generation. Also, for stochastic patterns, related work usually focuses on defining a new model. Then, the control of that model often directly derives from that specific model.

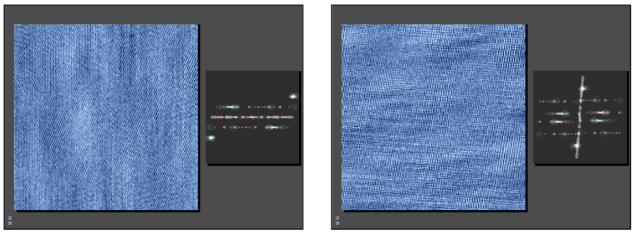


**Figure 4:** *In blue, Gabor noise examples. To their right an interactive visualization of their power spectrum for editing the visual characteristics. [GLLD12].*

**Example-Based** Most example-based stochastic texturing techniques do not offer further artist input beyond the exemplar but focus on performance. LAGAE et al. [LVLD10] match noise bandwidth for isotropic multi-resolution noise in a few milliseconds, given by GILET et al. [GDG12a]. GALERNE et al. [GLM17] introduced an efficient sparse convolution noise based on textons. The example match takes between half a second and five seconds, depending on the resolution.

GALERNE et al. [GLLD12] add to the example fitting an interactive visual editor for adjusting their Gabor noise, taking about two minutes per texture,. Sets of Gaussians represent the power spectrum of the noise, which can be rotated, scaled, translated, and cloned. Due to the abstract visual nature of a power spectrum, its connection to the visual features of the noise is not directly intuitive for artists. Hence, the editor has a strong exploratory nature to it. However, as the editing itself is interactive and visually appealing, it is inviting to do so.

GILET et al. [GDG12a] focus on increasing the expressiveness of their model toward more structural texture designs. For that, they introduce a noise that can approximate arbitrary spectral energy distributions. A straightforward noise-by-example computation takes up to 20 seconds, depending on the number of artist-defined convolution noises. For greater control and expressiveness, a perturbation function and a multi-layer approach are presented and some configurations can be given by artist-defined image maps. Further pursuing the topic of greater expressiveness and structured noise, GILET et al. [GSV*14] introduced a local random phase noise. Adjustable parameters control the visual quality of the noise and the amount of structure in comparison to noise. The authors do not report performance times for the matching step. PAVIE et al. [PGDG16] argue for control mechanisms being more intuitive in the spatial domain instead of the commonly-used editing of the power spectrum and align local random phase noise on a regular grid with a spot noise model based on a random distribution of structured kernels. Artists have interactive control of the spatial structures by modifying the spot functions and their distribution, thus increasing the range of possible designs.

GUINGO et al. [GSDC17] further improve on spatial variation and visual quality. They base their work on an underlying novel noise model and separate the handling of structures. Artists need to configure the fitting, and the performance of matching a $512 \times 512$ input image can take up to one hour with the current implementation not parallelized. KANG et al. [KH17] combine procedural noise with a data-driven tiling. So-called "non-features" are obtained by a noise-by-example method. "Features" such as edges can be edited in the feature image and are combined with the noise based on an artist-controlled ratio. The feature extraction for a $257 \times 257$ input image, and therefore the texture matching, ranges from few seconds to two minutes. GILET et al. [GD10] apply a more general optimization strategy for choosing the parameters of a noise-based model. With a given estimated light source direction in the input, GILET et al. [GD10] can create displacement map textures, with the parameter computation taking from one to three hours. With a given rough approximation of the geometry and a representative pattern patch in the input, even volumetric representations can be created from the exemplar.

### 7.1.2. Regular to Near-Regular Patterns

All the above discussed noise-based methods control a single stochastic procedural model. Even though recent advances greatly increase their expressiveness, the design space of noise-based models is too limited for creative pattern generation. Procedural models featuring regular to near-regular pattern designs (for a definition of the texture spectrum see LIN, HAYS, WU, et al. [LHW*06]) are usually optimized for specific design goals or even support a variety of procedural models within this class of designs. Figure 5 shows several regular to near-regular patterns.

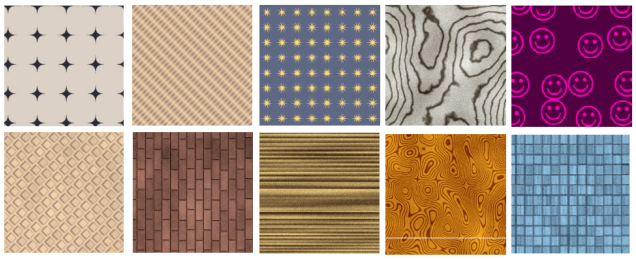


**Figure 5:** *Visual examples for regular to near-regular pattern designs [GKHF14].*

**Example-Based** For brick and wood textures, the early work of LEFEBVRE et al. [LP00] presents example-based control by transferring specific measured properties of an input to corresponding parameters for the procedural representation. The authors describe the matching performance as taking from a few minutes up to an hour. Expanding the design range notably, GILET, DISCHLER, and GHAZANFARPOUR [GDG12b] focus on the interactive creation of procedural semi-structured texture models and their visual features.

Random variations of artist input pattern are generated considering hierarchical spatial relationships, . In order to do so, an artist needs to give multiple exemplary object distributions.

BOURQUE et al. [BD04] allow for the whole procedural texture spectrum with a parameter retrieval technique. For the fitting, artists need to chose a configuration from two types of similarity metrics and two types of optimization strategies. As initialization for the optimization, the authors propose "on the order of 200" pre-computed random choices. The authors report an average optimization time of 12 minutes, not specifying for how many parameters. However, GILET et al. [GDG12a] report more than an hour runtime. With a higher number of parameters the current form of the approach quickly becomes unfeasible. GIESEKE et al. [GKHF14] build up on the work of BOURQUE et al. [BD04] and interpret the parameter matching as retrieval task. Based on pre-computed caches and a perceptually motivated image metric, their technique achieves interactive performance for fitting a $256 \times 256$ exemplar. As a one time investment the caches for the textures have to be computed. An artist has to chose the texture model to be matched. A similar approach offer HU et al. [HDR19] by training convolutional neural networks for the parameter retrieval. A style transfer step in the pixel domain is added for fitting visual details. Next to the given input example, the user has to chose from four high level texture classes. With the pre-computed caches ready, the fitting is interactive, with performances around one second, depending on texture resolution. The style-transfer lies in the range of minutes. Overall, for parameter retrieval methods the parameter count is highly influential on the performance for both visual quality and computation time. Heavy one-time investments must be made to compute the caches or train the network, which then determine the possible design spaces. Artists could use such techniques for computing a reasonable starting point for their creative work.

While focusing on stochastic pattern, the semi-procedural approach of GUEHL et al. [GAD*20] offer a much larger design space than related methods by enabling the combination of patterns. The technique generates textures based on input exemplars and gives an artist the option for manual editing and database browsing to shape the output (Figure 6) At its core the system is based on a noise-by-example approach. The appearance space of the noise can be explored with an interactive 2D map and the browsing of a database of preview images, which are unusual options for example-based modeling. The interactive 2D map is visually abstract and not suitable for reaching a specific output quickly but helpful for exploring the design space. Data-driven details are smoothly combined with the noise and the user can adjust the output with different parameters. The system is interactive with reported synthesis times below one second.

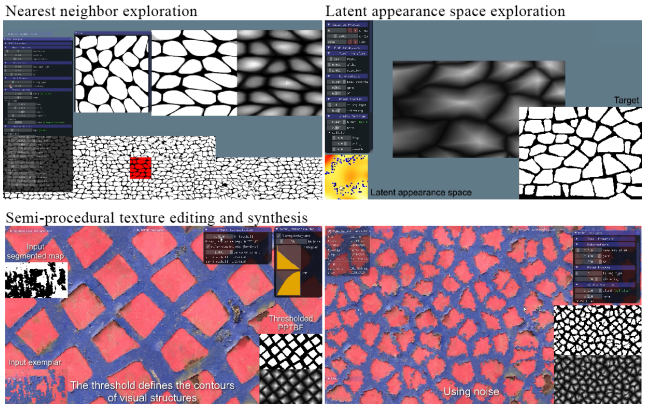



**Figure 6:** GUEHL *et al. [GAD*20] offer various control mechanisms for a semi-procedural generation of stochastic patterns. Examples are from the supplemental video. With the different controls, an artist can work exploratively as well as implement a specific design goal efficiently.

**Data-Driven** By adding appearance as an objective when optimizing for a structurally sound topology, MARTÍNEZ et al. [MDLW15] offer a data-driven example-based approach. Example patterns are given as raster images. The authors report on performances between 1 and 15 minutes for exemplars up to $330 \times 330$. BIAN et al. [BWL18] build upon that work by controlling topology appearance with custom-made vector pattern tilings. The technique is worth pointing out for automatically helping an artist create structurally sound connections between manually drawn tiles. Tiles can be drawn from scratch. The interactive interface gives hints and corrections for the construction of structurally sound designs, such as previews of the pattern on the canvas, snapping to proper correction points, and automatic corrections near tile boundaries. Similarly, LI et al. [LBMH19] optimize for structural soundness when putting decorative elements together to generate quilting patterns. A user inputs the decorative element, a region boundary and configuration values. Then a quilting pattern is generated automatically. There are no further adjustments of the result possible. The performance is below ten seconds.

Similar to the interactive authoring of BIAN et al. [BWL18], TU et al. [TWY*20] merge automation with manual creation. The algorithm synthesizes continuous curve patterns from exemplars made of BÃľzier curves, replicating not only on the position of the elements but also on their connectivity. The authors report matching times of 160 seconds depending on the sampling density. The example-based generation process is supported by various artist-controllable interactions before, during and after the creation process. Connections are continuously re-computed. An artist can also draw the example and connections directly on the canvas from scratch.

ZEHNDER et al. [ZCT16] provide artists with a tool to directly assemble structurally sound curve networks on a three-dimensional surface in 3D. The components of the network are spline curves defined by the artist. Components can be placed manually or are

repeated semi-automatically. The curves can be moved on the surface while having an elastic quality to them, which seems to be a quite engaging task. To prevent structural weaknesses, the system indicates problematic areas and suggests improvements, seamlessly combining the design task with engineering requirements. The performance of the automatic packing highly depends in the number of curves and ranges from seconds up to 15 minutes on average for the presented examples. For filigrees, which are thinly structured repetitive patterns, CHEN et al. [CZX*16] present a mainly data-driven approach. Their method automatically distributes and assembles a set of suitable input elements. The filigrees are mechanically strengthened through the optimization of a packing problem, which must be configured by the artist. Additionally, a directional stroke field can be drawn on the canvas, controlling element orientation and size. Element distribution percentages can be given when multiple elements are combined into one common pattern. The performance in two-dimensional space runs from 6 to 26 seconds.

Within their specific design spaces, the example-based data-driven techniques of BIAN et al. [BWL18], TU et al. [TWY*20], and ZEHNDER et al. [ZCT16] highlight the promising direction of automating filling and repetition, and computing, e.g., structural constraints, while giving an artist enough interactivity for creative exploration.

#### 7.1.3. Rule-based and Design-Specific Patterns

In the following we summarize designs that are based on a specific set of rules or grammars, as for example the ornamental patterns shown in Figure 7 or the tangles pattern in Figure 9. Hence, there are no limits to their expressiveness other than their underlying creation logic. Also, the following section discusses techniques that focus on the filling of global *shapes* and *masks*.

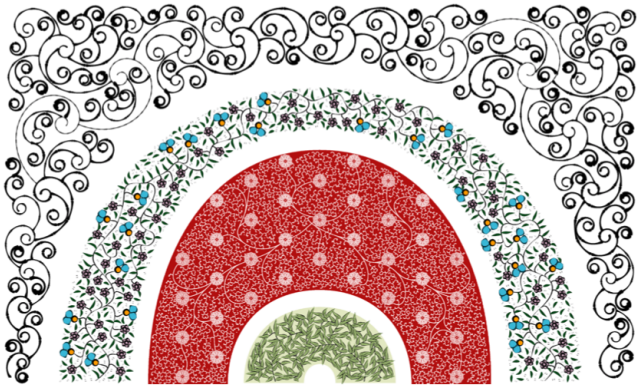


**Figure 7:** *Visual examples for the rule-based iterative space filling model from* WONG *et al. [WZS98].*

WONG et al. [WZS98] introduced a programmable procedural system that employs a greedy rule-based strategy to generate ornamental patterns. A procedural model is created with decorative elements and with a set of growth rules that handle the selection, appearance and connections of the elements. The process iterates, finding 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. All adjustments to the design and layout of an ornament have to be done by writing code. The authors do not report any performance times.

SANTONI et al. [SP16] and GIESEKE et al. [GALF17] present rule-based and design specific patterns. These techniques feature frames and hierarchies and are discussed in Section 7.2). LOI et al. [LHVT17] present a custom-made procedural framework that can create a large variety of element texture designs. The authors aim for designs that are unrelated to their spatial location and the space they fill, calling it stationary. Their programmable method is developed for technical artists and requires programming expertise. Pattern scripts are built with partitioning, mapping and merging operators. These operators enable both global and local design control and the composition of designs. The operator-based technique would enable a node-based interface design, which is not explicitly demonstrated in the article. The execution time for most designs is a few seconds, with some examples taking more than 1 minute. A user study with technical artists carefully evaluates the system's scripting interface, concluding positive results overall.

**Probabilistic Interference** Other systems provide the control to fill an outlining shape by interpreting the procedural modeling task as a probabilistic inference problem.

TALTON et al. [TLL*11] extend grammar-based procedural models by decoupling the growth control from the grammar itself. Their flexible analytic objective functions take images and volumes as global controls. The authors discuss that some experimentation might be needed to achieve a desired design goal, making the approach less transparent. Performance depends on the complexity of the grammar and the number of optimization steps needed. The authors report performance times ranging from a few seconds to several hours. For their examples, the authors manually terminated the optimization iteration. Similarly, RITCHIE et al. [RMGH15] control rule-based hierarchical and iterative procedural models with image-based matching and target volumes. The work improves convergence behavior and final scores. The reported performances range from around 3 seconds to 12 minutes, and the authors show that the number of included primitives scales reasonably. RITCHIE et al. [RTHG16] uses machine learning to improve the performance even further, increasing performance up to 10 times by integrating a neural network and sampling a learned constraint-satisfying approximation. Reported performances are overall below 3 seconds. As interactive performance is the foundation of creative control, this is of great importance.

**Fields** Fields have been used to creatively control the appearance of patterns with great success for several design features, meriting a separate discussion of them as control mechanisms in Section 8.3. In the present context, for repeating structures and filling shapes, LI et al. [LBZ*11] present a shape grammar that is guided by either a vector or tensor field. The field can influence the grammar's translation command, potentially leading to globally pronounced structures. The field can furthermore guide rotation, scaling, and

color parameters. The artist can specify a priori field constraints, such as regular and singular field elements, on the surface to be filled. Once the field is computed, local Laplacian smoothing can be applied. The authors report a synthesis performance for geometric surfaces from less than a second up to 3 minutes. Recently, applying a parameter field and a density field for controlling the appearance, ETIENNE et al. [EL20] present the procedural generation of band patterns as pixel shader in realtime. Both fields can be controlled by image input.

**Example-Based** ŠT'AVA et al. [ŠBM*10] present a context-free L-System that is able to recreate a given two-dimensional vector image consisting of groups of line segments. The algorithm creates similarity groups of these basic elements, computes spatial relationship clusters and iteratively translates these into rules. An artist is required to define a similarity threshold and significance weights for the different clusters, such as element distance or similarity, for example, thus guiding their representation according to the L-system rules. The time needed for the inverse step, depending on the number of elements in the input, is reported to range from a few seconds up to 20 minutes. TALTON et al. [TYK*12] further generalize the idea of inverse grammar generation and interpret it as a probabilistic interference problem. Their system induces a probabilistic formal grammar from a hierarchy of labeled components. The authors demonstrate their technique on scene graphs for geometry models as well as DOM trees for web layouts. The authors do not report any performance times for learning design patterns.

### 7.1.4. Element Arrangements

Element arrangements, such as shown in Figure 8, have individual and unconnected visual entities as their smallest unit. The elements themselves usually come from separate input data, such as vector graphics. In its broadest sense, the underlying distribution models for the arrangements can be seen as a procedural model. Even though there are no generative rules, characteristics of the discrete elements and their distribution can often be parametrized, and changes can be automatically processed and reproduced in the output. When filling a shape with elements or masking areas on the canvas, elements should ideally not be cut and align themselves for evenly filling the shape.

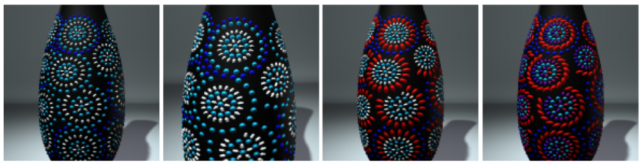



**Figure 8:** *An example for a pattern in 3D made of discrete elements [MWT11]. MA et al. also offer extensive editing options such as the propagation of the edit of a single element to the overall pattern.*

**Example-Based** For an example-based generation of element arrangements, relationships between elements are usually extracted from example arrangements and reproduced for the synthesis. In the following, we only include techniques with some form of user input beyond non-creative system configuration parameters. Much previous work [PH19; CXL19] focuses mainly on point distributions.

BARLA et al. [BBT*06] and HURTUT et al. [HLT*09] focus on example-based element arrangements of stroke-based vector elements. BARLA et al. [BBT*06] synthesize arrangements by matching local neighborhoods to a global seed distribution computed by Lloyd relaxation. Computing arrangements takes up to 10 seconds, and artist input is used in addition to the stroke patterns. Further visual adjustments are possible in a post-processing step. HURTUT et al. [HLT*09] can capture non-uniform distributions and improve the performance to the order of seconds. As a possible artist input, one exemplary shape input and density map are shown, and other input options are discussed in principle. The authors clearly state their focus to be on automation.

IJIRI et al. [IMIM08] combine data-driven texture synthesis with procedural generation. Their technique analyzes a given element distribution with local neighborhood comparisons and synthesizes new arrangements with interactive performance with incremental rule-based local growth. Element attributes that go beyond the positions of the elements and orientation cannot be controlled. Artists have a variety of design options with element orientation modes, an interactive spray tool to define areas to grow in, and a flow field tool to define overall alignments and a boundary tool. Moreover, the reconstructed topology can manually be adjusted. IJIRI et al. [IMIM08]s' work is an early example for combining a data-driven with a procedural approach. Their tools allow artists to work on the canvas directly and to focus on the actual output, even with procedural models.

The technique of MA et al. [MWT11] is based on a sample of a discrete element distribution and an output shape to fill both in two and three dimensions. The exemplar has to contain the actual elements in their domain and cannot be pixel data. In order to fill the output shape with elements, an energy optimization is processed with a novel neighborhood similarity metric. In addition to element positions, the metric includes variable features referring to orientation, geometry, appearance and type, for example. Hence, the metric is capable of reproducing global aggregate distributions that go beyond local element placements. The authors also extended their work to the spatial-temporal domain [MWLT13]. Necessary inputs are the exemplary element distribution, the neighborhood size to consider and the output shape. Further distribution constraints based on element attributes are optional. Examples for the inclusion of a vector field and element drag and drop are given. The authors report performance times between one and ten minutes with a non-optimized implementation.

ALMERAJ et al. [AKA13] base their example-based geometric texture generation technique on how such textures were manually created in a previous user study [AKAL11]. The idea of developing a generation algorithm based on a systematic study of how artists manually create patterns is worth further investigation. The authors identify tiling, structure and randomness as the prominently used creation strategies. The patch-based algorithm, which focuses on eliminating the appearance of regularity, doesn't seem to allow for

any user interaction. The authors do not report on the performance. LANDES et al. [LGH13] present an element distribution technique in 2D as well as 3D that improves on collision-free and anisotropic distributions with spatial relationship measurements. Next to the exemplar, there are several user inputs possible, such as a gradient image for the distribution intensity and parameter for the fitting algorithm. Even though these can control a visual range, their communication is quite abstract. Performances range from seconds to several minutes depending on the complexity of the arrangement.

**Fields** SAPUTRA et al. [SKAM17] optimize a flow-based ornamental packing of elements into a two-dimensional outline. For each element, a predefined spine controls the element's deformation. The artist defines direction guides and optionally fixed elements that control the computation of evenly placed streamlines. Elements are placed and deformed along streamlines. An iterative refinement step optimizes for a dense and balanced filling. An average packing takes about an hour. SAPUTRA et al. [SKA18] build up on their previous work substituting the flow-based packing with a mass-spring system. An additional secondary packing step further fills gaps with simpler shapes. With that the technique achieves denser and more even packings. A user provides primary and optionally secondary elements, the closed shape to fill and a distance for the spacing between elements. Packings take between 2 to 20 minutes, including both, the packing of the primary and secondary elements.

HSU et al. [HWYZ20] present an interactive brushing systems for placing aggregations of elements directly. As the work stands out through its brush-based control mechanisms, we discuss it in Section 7.3.

**Data-Driven** PHAN et al. [PLA*16] offer a data-driven recommendation system for circular ornamentation, employing a machine-learned style and composition feature vector. Based on a custom ring-based layout system that represents, for example, plates and vases and a first decorative element chosen by the artist, the system completes a design. The artist can also chose to incrementally add elements manually, while the system accompanies this by suggesting suitable elements and placements. This work indicates the promising direction of using learned characteristics for tools that stimulate with, e.g., meaningful design suggestions.

### 7.2. Frames and Hierarchies

For the design feature of *Frames and Hierarchies*, results usually consist of different pattern in different areas (Figure 9). Frames are either based on filling a frame-shaped space or on lines and curves. This type of control usually gives an artist more direct visual control than the previously discussed methods, which mainly focus on filling a shape. In addition to the visual output being further structured, the control is put onto the actual canvas.In the following, we cluster this work based on curves, shapes, and masks.

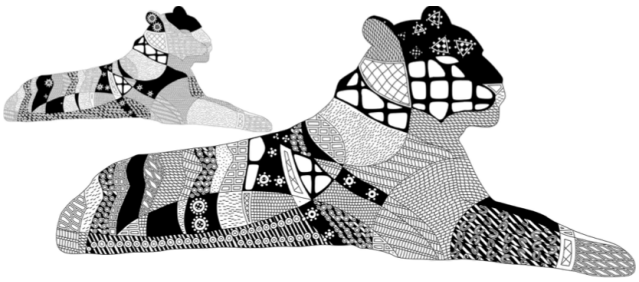


**Figure 9:** *An example for a tangle pattern from* SANTONI *et al. [SP16]. On the left, the automatically generated tangle pattern and on the right the pattern is further edited by an artist.*

**Curves** ANDERSON et al. [AW08] place discrete elements on the sides of an artist-given curve based on techniques from WONG et al. [WZS98]. The artist can input masks and use proxies to control the size and type of elements to be placed next to the curve. Two interfaces exist, the interactive view and the buffer view. The authors do not report a user study or specific performance times but call their system interactive.

**Shapes and Masks** BENES et al. [BŠMM11] offer a complex shape-filling and masking system for procedural L-system models by dividing a target space into artist-editable guide shapes. Seeds for the L-system are interactively given by an artist as a position and orientation. The guide shapes determine what types of patterns grow in different areas. The connections between the shapes are manually specified by the artist and in turn guide the connections between elements, possibly creating frames and emphasizing hierarchies. Based on a mass-spring system, the guides can be intuitively edited as a whole. The authors report 30 minutes up to one hour for inferring the L-systems. The generation times for most pattern examples are less than a second, with up to 45 seconds for only one complex scenario.

SANTONI et al. [SP16] present the design-specific generation of tangles with a stochastic shape grammar. Tangles are repetitive black-and-white hand-drawn patterns made from dots, straight lines, simple curves and circles (Figure 9). The visual elements of Tangle patterns can align to the shape they fill and this alignment can create borders, frames and hierarchies automatically. A tangle generation usually takes a few seconds, with a complex example taking about 3 minutes. The authors present an interactive system for creation based on a parameterized artist interface, including rule re-expansion and sketch-based operator modification. The presented system is a powerful combination of editing operations with procedural generation. The work also includes navigation through the editing history, which is noteworthy as this basic operation needed for creative control is usually overlooked. A user study evaluates the system as accurate, controllable and easy to use after a reasonable training time.

**Curves, Shapes and Masks** Also building upon WONG et al. [WZS98], GIESEKE et al. [GALF17] offer, among other fea-

tures, several mechanisms to create frames and hierarchies for procedural models. The authors specifically focus on the on the generation of procedural ornamentation. At the core of their system, procedural element placement can be combined with custom-made placement functions, which enable global design constraints such as symmetry. An artist can control the overall growth of the pattern as well as the connectivity of the elements by drawing frames and paths directly onto the canvas or by designing a vector field by sketching its directions. While all editing steps are interactive, more complex designs can have computation times up to several minutes, depending on the chosen placement functions.

### 7.3. Curves and Brushing

For the design feature of *Curves and Brushing*, we consider curves and brushing as visual elements, as well as the overall control mechanism, which in turn leads to a distinct visual style. Figure 10 give a visual example of both approaches. For brushing, we further cluster data-driven mechanisms and feature exploration techniques.

Formed curves such as circles, spirals or hearts are essential components for many pattern designs, including ornamentation, and are discussed first. Incorporating an artist-defined curve as the spine of a pattern, CHEN et al. [CSC12] use an interactive L-system to attach decorative spiral designs to the curve given by an artist. XU et al. [XM09] use the space-filling algorithm of WONG et al. [WZS98] in combination with particle tracing in simulated magnetic forces for the generation of decorative curves. Physical properties and the initialization of the particles are the parameters for designing the curves. The computation takes less than five seconds. The authors acknowledge the non-intuitive parameterization of the system and give an example timing of two minutes for finding the parameters of a specific design goal. MERRELL et al. [MM10] generated a set of curves in the same style of a given parametric example curve. A style is defined by local properties, such as tangents and curvatures, that are derived from a local shape analysis. The new curves are computed with a rule-based system that allows artists to interactively edit the result. Interactivity is somewhat diminished by computation times of up to minutes for a curve set.

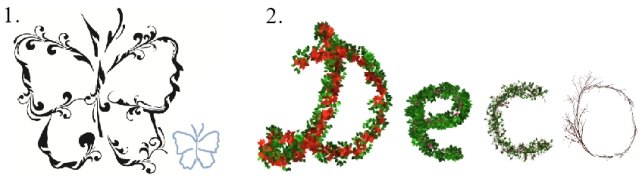



**Figure 10:** *Left, an example for curves used as control mechanism for a data-driven approach [LBW*14] and on the right for procedural pattern generation [MM12].*

For more individual designs, brushing methods create output along curves but do so directly without taking an a priori completed curve into consideration, as if using a spray can or a brush. Brushing techniques usually include a brush diameter, hence the size of the area to be filled along the curve. MĚCH et al. [MM12] present the flexible *Deco* procedural engine and examples of brushing methods for different aspects of generating procedural models, from brushing growth constraints, such as masks, to having a pattern grow along the strokes. This discussion only refers to the actual examples given by the authors. However, the engine opens up and generalizes environments for interactive control mechanisms for various types of procedural models. For the programming of decorative pattern models within the engine, helpful functionalities, such as symmetry objects and control guides, are predefined. All artist control mechanisms have interactive performance. Overall, performance mainly depends on the pattern generation scripts. The engine can load pattern codes as a dynamic library, optimizing performance.

For texture generation, there are various methods that have successfully combined texture synthesis with a brushing interface. For example, LUKÁČ et al. [LFA*15] present an interactive example-based brushing system that processes user-given shapes and directions for the features of the synthesised texture. HSU et al. [HWYZ20] builds on this, presenting an interactive brushing systems for placing aggregations of elements directly. The work focuses on an even distribution and on resolving collisions. An artist can do add, erase and replace operations with the brush and also sketch a density map. The brushing is combined with an autocomplete functionality with element fields to control the automatically filled elements' alignment based on brushed directions. Filling is computed by iteratively optimizing the scale, orientation and position of the elements. Elements can be rigid or be deformed through the packing. The technique is applicable for 2D planes, 3D surfaces and 3D volumes. Performance is below 30 seconds for 2D representations. Overall, this work is a convincing combination of manual creation with automatization for creating creative element arrangements. DAVISON et al. [DSJ19] add to the work with a brushing technique that employs several example arrangements as palettes, which can be freely combined.

More painting-like methods can be found, for example, in procedural botanical modeling [APS08; CNX*08; PHL*09], procedural landscape generation [EVC*15], as part of a procedural water color engine [DKMI13] or for dynamic effects [XKG*16].

Going far beyond simple curve structures, JACOBS et al. [JBMR18] developed the programming and drawing environment *Dynamic Brushes*, in which an artist can create individual procedural brushes for a stylus pen. General programming logic and relevant mathematical functions for creating patterns are translated into a visual programming interface. The evaluation of the system by two professional artists shows that once initial struggles to learn the system were mastered, the artists were able capture their personal analog styles with the procedural brushes. Overall, the authors and the artists open many valuable questions about the usage of current tools and about alternative approaches that seek to seamlessly blend manual and procedural creation processes. *Demystified Dynamic Brushes* [LBM*20] expands on this by giving an artist further options, e.g., with visualizations on the canvas to investigate the linking between the procedural modelling and the visual output. Also, the creation history is recorded and can be navigated. Overall the evaluation of the system with artists indicates

that a tighter bond between visual work and programming tasks make procedural modelling to artist more accessible.

**Data-Driven** LU et al. [LBW*14] create a pattern along a sketched curve with a data-driven approach. Vector pattern exemplars are placed and deformed along the curve using optimized visual soundness. For the exemplars, an artist has to define the start and end point of their spines. The artist can refine results with "add" and "erase" constraints that are drawn on the canvas. The authors report a performance of an average of eight seconds per stroke for the examples given. A related data-driven approach for synthesizing example-based vector patterns along a curve was presented by ZHOU et al. [ZJL14] in the same year. In this work, the authors focus on ensuring a structurally sound output pattern. Artists can input topological constraints, local pattern orientations and a value for variation. Once a pattern is generated, an artist can interactively adjust the underlying curve, with the pattern being updated accordingly. Generation performances are reported from below one second up to a little more than two minutes for complex models.

Creating textures from pen-and-ink drawings, KAZI et al. [KIZD12] present a multifaceted tool, mixing data-driven and procedural modeling. Simple manually-created drawings can be automatically repeated along paths and brush strokes, and can be used to fill regions. Edits of the original drawing can be propagated to all repeated elements. A user study confirms the system's usefulness for efficiently creating repetitive textures while maintaining the natural workflow and artistic control of an artist. XING et al. [XCW14] build upon that work by automatically detecting and suggesting possible repetitions to the artist, aiming for a less regular, more painting-like quality. The presented system also offers various brush options and navigation tools in order to combine automation with artist control. Neither of these two articles report on performance times for the computation of the strokes and edits.

More data-driven painting-like methods can be found, for example, for hand-drawn animations [XWSY15], creating mosaics [Iga10; AGYS14] and data visualization [XHC*18].

**Feature Exploration** Even though not a generating technique in itself, exploration is an important characteristic of a creative process. TODI et al. [TWO16] present a tool for exploring common layout types with sketched input. With the method of CHEN et al. [CFA16], an artist can browse a collection of texture images by sketching highly abstracted pattern features. The represented structural features of reflection, rotation, and translation symmetries adhere to important design principles for visually pleasing patterns. One could imagine a similar intuitive approach for exploring the parameter space of an ornamental procedural representation, for example.

In the context of 3D modelling, TALTON et al. [TGY*09] investigate a collaborative 3D modeling system by crowd-sourcing possible models and feeding the results back to the system for others to explore in a structured manner.

### 7.4. Connections, Branches and Directionality

We now discuss systems that include the design feature of *Connections, Branches and Directionality* in a generally-applicable manner. By comparison, some already-discussed work [CSC12; XM09; MM10], enables only specific branching designs. The general category includes the already-discussed technique of BENES et al. [BŠMM11]. Even though their work focuses on structuring a space with shapes, an artist can also control the connecting points between those shapes. In the following, we identify the mechanism clusters of fields, example-based, and data-driven.

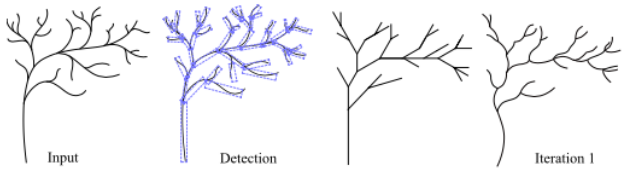



**Figure 11:** *Visual examples for branching structures from* GUO *et al. [GJB*20], who recreate a user sketch (left) with a procedural L-system (right) through inverse modeling.*

**Fields** The already-discussed work of GIESEKE et al. [GALF17] (Section 7.2) enables a sense of directionality by controlling the growth of a pattern through vector fields and directing the connectivity and branching of elements along user-specified paths. Connections between single elements can be designed through drag and drop. Patterns align to the space they are filling by automatically growing around obstacles, implemented with a shortest-pathfinding approach. Of the systems described in Section 7.1.4, IJIRI et al. [IMIM08] employ vector fields to define the overall growth direction and alignment of elements within an example-guided arrangement, enabling the design of directionality. Similarly, SAPUTRA et al. [SKAM17] optimize a flow-based ornamental packing of elements into a two-dimensional outline.

Vector fields are further employed in various other specific procedural modeling contexts, including procedural street modeling [CEW*08], micrography [MBS*11] and botanical models [XM15].

**Example-based** GUO et al. [GJB*20] focus specifically on creating branching structures with an inverse modeling process for inferring a generating L-system. The technique is robust and takes a variety of input designs such as real-world images or hand-drawn sketches of the branching. The inverse modeling employs convolutional neural networks, a tree graph for representing transformations and greedy optimization. Beside the exemplar, a user can input the length of the rules and the frequency of the repetition. Once the network is trained, the inference take a fraction of a second.

**Data-Driven** Even though the energy-brushes of XING et al. [XKG*16] are intended for creating animations, and hence are not included in the table, they could also be used in a pattern generation process. The technique could be used to deform given visuals in an aesthetic manner, perhaps under certain design constraints for pattern generation. Their brushes let an artist roughly sketch directions that shape smoke, swirl, and wind velocity fields, which in turn control the animation of the illustration. An abstracted preview

of the type of brush is given on the canvas letting an artist translate abstract strokes into the desired animation. The user interface gives all basic interactions of a drawing tools, such as layers and undo. In contrast, HU et al. [HXF*19] solve the design of velocity fields by freely sketching all possible scene elements, such as boundaries, obstacles and specific types of flow, offering more freedom but also leading to less-structured results.

### 7.5. Single Accents

The design feature of *Single Accents* breaks with the underlying principle of procedural generation, which is to repeat certain rules and with that visual features. This overall principle is demonstrated in Figure 12 However, for creative pattern generation, it is worthwhile to investigate how to combine the execution of rules with occasionally breaking such rules. For example, GIESEKE et al. [GALF17] claim this to be necessary for ornamentation to break "an otherwise too-homogeneous appearance." Because of the small amount of work matching this design feature, there are no mechanism clusters.

Even though the required functionality can be compared to using the tip of a brush, paint-like procedural modeling techniques such as MĚCH et al. [MM12] often have a more spray-can-like quality and do not explicitly include the option to place single elements. GIESEKE et al. [GALF17] (Section 7.2) explicitly provide local editing options, including deleting and placing single elements and connections and dragging and dropping existing ones. SAPUTRA et al. [SKAM17] also support accent elements in their field-based ornamental packing generation. The system-generated elements automatically flow around the placed accents.

YEH et al. [YM09] allow for separating single elements by combining a manual data-driven design process with procedural-modeling-like editing options. Based on detected symmetries and curvilinear element arrangements in a given vector pattern, an artist can adjust the spacing, location and scale of one element and propagate that change to the all other elements in the group. The authors also offer a brush that recreates recognized element groups. The technique is interactive for the examples shown but interactive performance does not scale to more complex input patterns.

**Figure 12:** *While the pattern exploration technique of* GUERRERO *et al. [GBLM16] focuses on exploring design variations with repetitive visual characteristics, e.g., based on symmetry, as shown on the left, the technique also enables the placement of single elements as shown on the right. The example is from the supplemental video.*

The vector pattern generation technique of GUERRERO et al. [GBLM16] also supports single accents while the artist is exploring design variations. An artist can select and continue with one of the offered alternatives. The system constantly re-selects from large space of relevant variations based on the artist's modifications. The user interface is carefully laid out to offer design variations in an intuitive and efficient manner while not hindering the artist's own workflow. The authors thoroughly evaluate their system quantitatively and qualitatively, including a user study. Overall, participants agreed on the usefulness of technique.

## 8. Discussion of Means for Enabling Creativity

Table 2 compares performance timings as reported by the original authors. As the works discussed here come from a time frame spanning over 20 years, older techniques can be expected to significantly outperform their original implementations on current platforms. Accordingly, we categorize them coarsely into *realtime*, *interactive*, *synchronous*, *asynchronous*, and *offline* to indicate how one would work with the proposed techniques, and how their original application may have been intended. To relate the control mechanisms to the control characteristics, we considered the same publications in Table 3. Due to the diversity of the underlying methods and the different design goals of the considered body of work, we believe this to be a representative summary.

The categories in Table 1 show a highly uneven distribution. The largest category is *Distribution and Repetition*, which focuses on repetitive pattern designs and filling a space, usually automatically. Another large group is *Curves and Brushing*, which lets artist manually create distinct but potentially unstructured designs. However, creative pattern designs, such as the historic and commercial patterns in Figure 1, are rarely uniform or unstructured, so additional control is needed to let artists create structure and working with design rules. The table categories of *Frames and Hierarchies; Connections, Branches, and Directionality*; and *Single Accents* include systems that attempt to address this need. These include algorithms that have a greater understanding of the space being filled and offer global planning to the artist.

In addition, there are domains closely related to pattern generation for which there are few creatively controllable algorithms. The interwoven structures in the Celtic pattern design in Figure 3 could be computed algorithmically while an artist controls the overall design, but systems that generate similar designs automatically [KC03; DS13] have limited controllability.

Table 3 shows that global control is most commonly enabled through intermediate representations, such as example images. At the other end of the spectrum, placing elements as part of the actual output is most often local and manual with little automation supporting the artist throughout the creation process. The dominant control mechanisms in the Parameterization category, *File* and *Value*, both require abstracted input from an artist, such as a text value or the use of a slider. Sketch-based controls move the interaction onto the canvas and can make small-scale adjustments. Spaces to fill, curves to follow, and masking areas are also usually done directly on the canvas, but only influence the output indirectly. Brushing creates the output directly on the canvas, but can only do so in

| | TIMINGS | | | | | |
|---|---|---|---|---|---|---|
| | Configuration | Realtime | Interactive | Synchronous | Asynchronous | Offline |
| | | <1s | <10s | <1min | <1h | >1h |
| *DISTRIBUTION AND REPETITION* | | | | | | |
| *Stochastic Patterns* | | | | | | |
| GALERNE et al. [GLLD12] | | | | | | |
| GILET et al. [GDG12a] | | | < | | | |
| GILET et al. [GSV*14] | | | | | | |
| PAVIE et al. [PGDG16] | | × | | | | |
| GUINGO et al. [GSDC17] | | | | | < | |
| KANG et al. [KH17] | | | | | | |
| GILET et al. [GD10] | | | | | | |
| *Regular to Near-Regular Patterns* | | | | | | |
| LEFEBVRE et al. [LP00] | | | | | | |
| GILET et al. [GDG12b] | | | × | | | |
| BOURQUE et al. [BD04] | × | | | | | |
| GIESEKE et al. [GKHF14] | × | | | | | |
| HU et al. [HDR19] | × | | | | | |
| GUEHL et al. [GAD*20] | | < | | | | |
| BIAN et al. [BWL18] | | | | | | |
| LI et al. [LBMH19] | | | | | | |
| TU et al. [TWY*20] | | | | | | |
| ZEHNDER et al. [ZCT16] | | | | | | |
| CHEN et al. [CZX*16] | | | | | | |
| *Rule- and Design-Specific Patterns* | | | | | | |
| WONG et al. [WZS98] | | | | | | |
| LOI et al. [LHVT17] | | | | | | |
| TALTON et al. [TLL*11] | | | | | | |
| RITCHIE et al. [RMGH15] | | | | | | |
| LI et al. [LBZ*11] | | | | | | |
| ŠT'AVA et al. [ŠBM*10] | | | | | | |
| TALTON et al. [TYK*12] | | | | | | |
| *Element Arrangements* | | | | | | |
| BARLA et al. [BBT*06] | | | | | | |
| HURTUT et al. [HLT*09] | | | | | | |
| IJIRI et al. [IMIM08] | | | | | | |
| MA et al. [MWT11] | | | | | | |
| ALMERAJ et al. [AKA13] | | | | | | |
| LANDES et al. [LGH13] | | | | | | |
| SAPUTRA et al. [SKAM17] | | | | | | |
| SAPUTRA et al. [SKA18] | | | | | | |
| *FRAMES AND HIERARCHIES* | | | | | | |
| ANDERSON et al. [AW08] | | | × | | | |
| BENES et al. [BŠMM11] | | | | | | |
| SANTONI et al. [SP16] | | | | | | |
| GIESEKE et al. [GALF17] | | | | | | |
| *CURVES AND BRUSHING* | | | | | | |
| CHEN et al. [CSC12] | | | × | | | |
| XU et al. [XM09] | | | < | | | |
| MERRELL et al. [MM10] | | | | | | |
| MĚCH et al. [MM12] | | | × | | | |
| HSU et al. [HWYZ20] (2D) | | | | | | |
| JACOBS et al. [JBMR18] | | | × | | | |
| LU et al. [LBW*14] | | | | | | |
| ZHOU et al. [ZJL14] | | | | | | |
| KAZI et al. [KIZD12] | | | × | | | |
| XING et al. [XCW14] | | | × | | | |
| *CONNECTIONS, BRANCHES AND DIRECTIONALITY* | | | | | | |
| GUO et al. [GJB*20] | × | | | | | |
| *SINGLE ACCENTS* | | | | | | |
| YEH et al. [YM09] | | | | | | |
| GUERRERO et al. [GBLM16] | × | | × | | | |

**Table 2:** *A rough categorization of timings in realtime, interactive, synchronous, asynchronous, and offline. Timings are taken from the authors' descriptions. An × is used for work that the authors call interactive without giving specific timings.*

| | | | HOW | | | WHAT | | | | WHERE | | | WHEN | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Total (out of 50) | File | UI | Canvas | Code | Value | Intermediate | Element | Global | Region | Local | Before | During | After |
| | *Setup* | | | | | | | | | | | | | | |
| | Configuration | 13 | 13 | | | 9 | 4 | | | 13 | 2 | | 13 | | |
| | Initialization | 20 | 18 | 3 | | 3 | 16 | 8 | 2 | 19 | 3 | | 20 | | |
| ⟵ *Decreasing Automation* | *Exemplars* | | | | | | | | | | | | | | |
| | Image | 13 | 11 | 1 | | | | 12 | 2 | 13 | 2 | | 13 | | |
| | Arrangement | 13 | 11 | 1 | 3 | 4 | | 3 | 10 | 13 | 4 | | 13 | 1 | |
| | Element | 11 | 7 | 3 | 1 | | | 5 | 6 | 9 | 3 | | 11 | 1 | |
| | *Parameterization* | | | | | | | | | | | | | | |
| | Visual Output | 36 | 26 | 8 | | 5 | 30 | 2 | | 34 | 2 | | 19 | 1 | 19 |
| | System | 22 | 14 | 8 | | | 21 | 1 | | 23 | 4 | | 22 | 1 | |
| | *Handling* | | | | | | | | | | | | | | |
| | Visual UI | 10 | | 4 | 4 | | 1 | 7 | 4 | 7 | 6 | | 5 | 3 | 5 |
| | Image | 11 | 11 | | | 2 | | 9 | | 11 | 5 | | 11 | | |
| | Sketch | 10 | | 1 | 9 | | | 10 | 5 | 6 | 9 | 2 | 8 | 3 | 2 |
| | *Filling* | | | | | | | | | | | | | | |
| | Shapes | 41 | 32 | | 11 | 26 | | 14 | | 40 | 15 | | 41 | 3 | |
| | Masking | 11 | 5 | | 8 | 2 | | 9 | | 8 | 11 | | 11 | 2 | |
| | Curve | 9 | | | 9 | | | 7 | 2 | 1 | 9 | | 8 | 2 | |
| | *Guiding* | | | | | | | | | | | | | | |
| | Brushing | 8 | 1 | | 7 | | | 2 | 6 | 2 | 8 | | 2 | 7 | |
| | Directions | 9 | 1 | | 8 | | | 8 | 1 | 8 | 8 | | 9 | | |
| | *Placing* | | | | | | | | | | | | | | |
| | Element | 9 | | | 9 | | | | 9 | | 1 | 9 | 9 | 2 | 1 |
| | Drag&Drop | 7 | | | 7 | | | | 7 | 1 | 5 | 6 | | 6 | 3 |

**Table 3:** *Prevalence of control mechanisms in the literature: In total, 50 publications are included (the discussed state of the art in Section 7). Please note that the total for each step (how, what, where, when) can exceed the total of that category because the category can be implemented in multiple ways.*

a limited region depending on the brush size, unless used to brush intermediate control lines. All other inputs are typically given before or after the generation of the output. The classification underlines that a focus on one control type, as is often done in computer graphics research, leads to the common trade-off between global automation and local manual manufacturing.

We now discuss various techniques' potential to support a creative workflow, clustered into the most common types of control. This discussion of the means for enabling creativity, namely *Navigation, Transparency, Variation* and *Stimulation* (Section 4), is interpretive and somewhat subjective. However, we strive to combine a realistic discussion of terms like "artist usable" and "creatively controllable" with steps towards defining them more objectively.

### 8.1. Example-Based Control

As the state of the art shows, the most prominently investigated techniques for texturing are example-based and inverse approaches. Example-based control mechanisms provide a *goal-oriented control*. The motivation behind using these techniques is mainly to

generate a specific and predictable output as efficiently as possible. This type of control stands in contrast to exploration. Example-based approaches change the design task into a data-driven image generation task, such as taking a photograph or designing a sample in an application such as Adobe Photoshop or Illustrator. Relevant factors for differentiating example-based techniques are the size of the design space and hence their expressiveness, performance and initialization requirements.

The investigation of example-based techniques shows valuable achievements in goal-oriented control and in increasing design space size within specific contexts. With regard to creative control, crucial steps are increasing variability and improving navigability with interactive performance. However, many techniques still require considerable non-creative effort for an artist, such as working with a power spectrum or predicting how changes in the exemplar, such as element arrangements, affect the output.

TU et al. [TWY*20] and BIAN et al. [BWL18] combine drawing a tile with an automatic tiling system, a promising direction that offers transparent navigation. Also, drawing tiles leads to a design space that is fairly open within the scope of the specific pattern type, which is further limited by fabrication constraints in BIAN et al. [BWL18]. Both techniques offer direct canvas intereactions such as editing options on the tiling itself [TWY*20] and preview and snapping functionalities that make it easy for an artist to create structurally sound patterns [BWL18].

The potential of example-based methods for creative control lies in improving interactive performance, reducing initialization requirements, and experimenting with the spatial influence of the input, such as introduced by GUEHL et al. [GAD*20] with their semi-procedural generation of stochastic textures (Figure 6). Overall, the presented work focuses on global designs, such as the whole canvas and repeating regions. Methods for which regions could be defined, models layered or the placement of single elements integrated constitute valuable directions for example-based methods as creative control.

### 8.2. Shapes and Masks

Sophisticated masks and growth constraints lead to visually interesting and complex designs, such as the packing patterns from SAPUTRA et al. [SKA18]. However, it is not completely predictable exactly how a space will be filled. Because the interactive performance of many of the presented methods is quite limited, even a basic trial and error exploration is hardly feasible. The navigation of the design space becomes cumbersome, and stimulation becomes hindered. The one technique (SANTONI et al. [SP16]) that offers transparent navigation is also the one with a severely restricted design space, but their inclusion of a navigation history stands out from the related work in this survey. In terms of stimuli, the mass-spring system for editing control guides offered by BENES et al. [BŠMM11] and the elastic curves from ZEHNDER et al. [ZCT16] (Figure 13) are promising directions because they are intuitive and enjoyable to use and encourage exploration. Overall, the control mechanisms of ZEHNDER et al. [ZCT16] are a convincing solution for enabling artists to adhere to structural constraints while still offering a fully navigable, transparent and stimulating creative workflow.

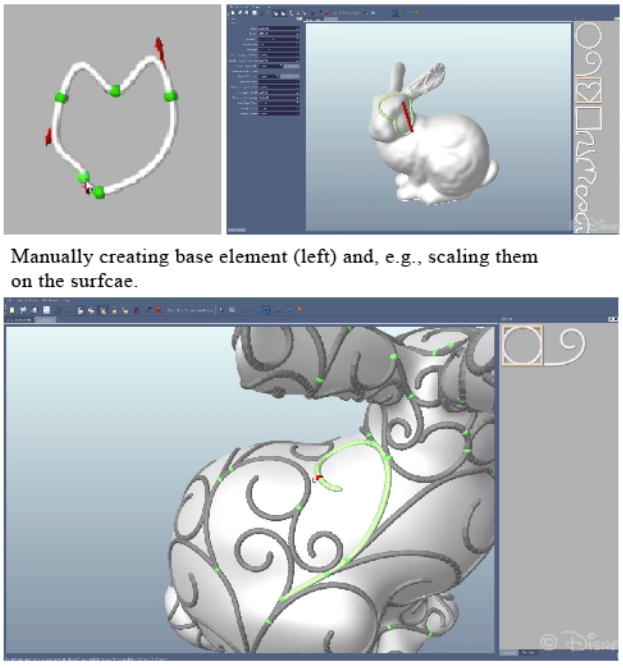



**Figure 13:** ZEHNDER *et al. [ZCT16] offer various control mechanisms, from creating the base elements (top-left) to combining and editing them on the surface, while ensuring structural soundness (bottom). Example are from the supplemental video.*

In terms of control mechanisms for a more complex design goal, shape fillings and masks do not permit hierarchical or element-level local controls or the control of element connections needed by artists who want to generate patterns creatively without having to write code.

### 8.3. Fields

Fields (in the context of this survey, usually vector fields) constitute a powerful tool for combining an automatic procedural filling with the option to design regions individually and with control often directly given on the canvas. As HSU et al. [HWYZ20], SAPUTRA et al. [SKAM17] and GIESEKE et al. [GALF17] show (Figure 14), the streamlines of a field can structure how a space is filled. Using a vector field to control, e.g., directionality, can greatly decrease the manual work needed to fill a space in its entirety because a few curves can define the entire field. Other global design choices, such as an overall growth direction or the alignment of elements, are simple to translate from a vector field to procedural generation rules.

The discussed work shows that fields allow for greater visual variation by opening the design space and providing transparent control for filling a space automatically. While fields are still a level of abstraction away from the pattern itself, they are intuitive to understand, e.g., through a visual representation. Their abstraction

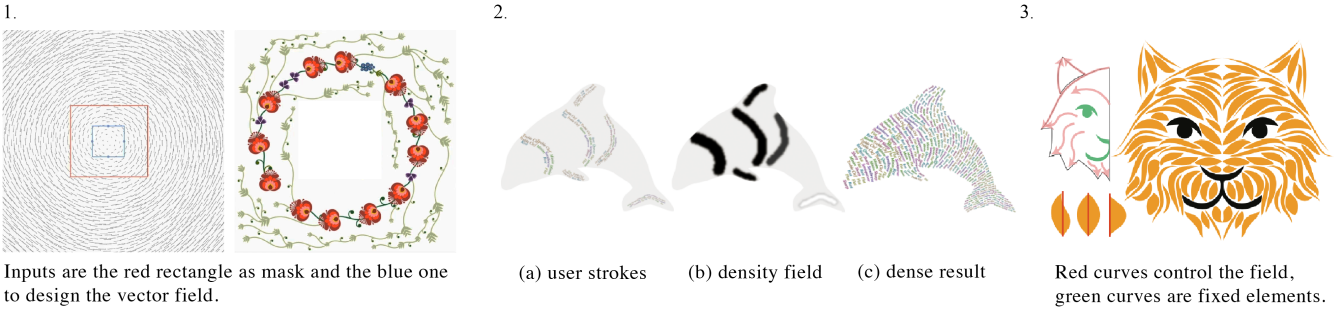



**Figure 14:** *Different approaches to design vector fields for creative pattern generation. 1.* GIESEKE *et al. [GALF17] (example from the supplemental video), 2.* HSU *et al. [HWYZ20], 3.* SAPUTRA *et al. [SKAM17].*

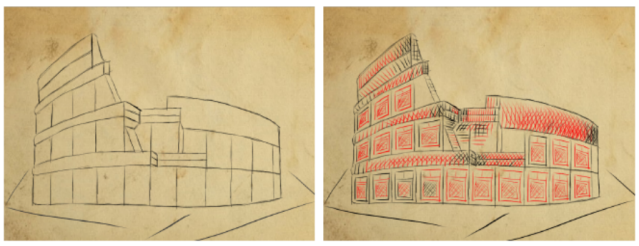



**Figure 15:** XING *et al. [XCW14]s' technique propagates user sketches (right, black strokes) to fill an outline automatically (right, red strokes).*

translates to the model in a straightforward manner. Thus, using flow within a vector field to design is a suitable control mechanism, especially for designs that aims to align their elements to the space.

### 8.4. Curves and Brushing

Curves and hand-drawn paths give an artist direct control over the final result, putting the control directly onto the canvas. Curves are needed for tasks such as creating a decorative frame or structuring a space. Some techniques consider the whole curve before computing the ornament and optimize the filling of the curve based on certain design goals. This can be understood as a form of global planning.

Curves and sketch-like methods offer well-communicated, transparent navigation. The discussed techniques are mostly interactive, and artists are familiar with their functionality from the real world and how they work directly on the canvas. The ease and directness of usage also constitute a foundation for possible immersive flow of work. Painting-like methods can support smoother navigation by integrating brush settings and increasing the quantity of controls.

On the downside, curves and brushing could lead to manual and tedious creation requirements for patterns and unstructured results. To balance this KAZI et al. [KIZD12] and XING et al. [XCW14] (Figure 15) incorporate procedural creation principles into a data-driven process. HSU et al. [HWYZ20] offer through a brushing system intuitive and transparent navigation and an artist can create any individual design goal for element arrangements. The approach of LI et al. [LBM*20] combines visual curve creation, code, and data in a unique manner by showing textual and numeric information about the underlying procedural algorithm on the canvas. The technique might increases a transparent and navigable workflow once an artist has learned the workflow. This novel approach calls for further investigation and verification.

In summary, with a brush artists are free to draw almost anything, depending only on their drawing capabilities. For creative pattern generation, however, structure, pattern hierarchies, and distribution techniques also need to be supported.

### 8.5. Element Placement

Placing single elements onto the canvas maximizes artist control and is on its own a trivial data-driven control principle. However, in combination with procedural modeling, this mechanism becomes interesting. Separately placed elements that do not follow any rules should be integrated and processed to remain part of the underlying global scene structure, as demonstrated by GIESEKE et al. [GALF17]. Element placement control mechanisms are closely related to data-driven sketch-based techniques and represent a promising approach for integrating procedural modeling functionalities into a data-driven process. GUERRERO et al. [GBLM16] present an overall transparently navigable and stimulating control mechanism (Figure 16). The carefully designed workflow fosters artist stimulation by offering novel but suitable design variations.

## 9. Outlook

This analysis shows various possibilities for future work within the specific contexts of creative pattern generation. Here we highlight directions that we have not mentioned yet but that have great potential for creative control mechanisms.

Collaboration is a valuable future line of investigation for enabling creative work. More aspects of common tools are either fully browser-based, or are in some way connected to the cloud-based

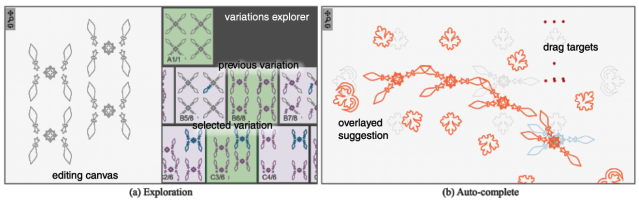



**Figure 16:** GUERRERO *et al. [GBLM16] combine the exploration of pattern variations with a manual editing of single elements, e.g., by drag and drop.*

storage of assets, settings and results; therefore, they function online and are easily shared. Collaboration is closely connected to the previously discussed issue of navigation histories. This is not only relevant for individual work processes but also for more general production pipelines in a commercial context. In this regard, the sharing and collaborative work on iterations, which involves multiple persons referencing different versions, is essential. Work has been done [TGY*09; SSTP15; OWL*18] but further investigations of collaboration for creative control are called for.

The procedural content generation (PCG) community for games is pushing the general integration of artificial intelligence (AI) into a procedural creation process. A new discipline of *mixed-initiative creative interfaces* is rising as, e.g., an ACM CHI workshop under the same name in 2017 [DHF*17]. The workshop summary states as its goal to "put human and computer in a tight interactive loop where each suggests, produces, evaluates, modifies, and selects creative outputs in response to the other." To achieve this, AI enables computer agency, and novel interfaces enable collaboration between computers and human users. On the topic of agency and automation, HEER [Hee19] discusses three case studies and poses several open questions for further developments.

Semantic attributes present a highly intuitive navigation technique, which so far has been successfully applied in the context of shape modifications, for example by YUMER et al. [YCHK15]. For procedural textures, these attributes can be based on the analysis and description of texture in regard to human perception, which has a long research tradition, starting with textons from JULESZ [Jul81]. Since then, this line of research has continued, and texture descriptions with perceptual [LDC*15] and semantic [MNN13; CMK*14] attributes have been investigated. DONG et al. [DWLS17] and LIU et al. [LGD*18] employed such features for the navigation of a procedural texture space and for the generation of suitable textures by given features. However, the results of such studies are still limited and of varying quality—the authors themselves [LGD*18] call their results experimental. Nonetheless, these works present an interesting approach that is worth further investigation, especially in combination with machine learning algorithms. Because many pattern designs are structured and follow an internal logic, it seems feasible to come up with a collection of suitable attributes. For example an ornamental design space is much smaller in comparison to all "textures in the wild" [CMK*14], and ornamentation could constitute a valuable context for further investigations into the incorporation of semantic attributes into a creative creation process.

As discussed in Section 6, control techniques are closely interrelated with the representation of the underlying models. Further automation for the creation of complex models also poses interesting challenges, such as abstraction [NSX*11], symmetry computation [CO11] and design space variations.

## 10. Conclusion

The overall challenge addressed in this survey is to show how to support artists in their work with meaningful control mechanisms. The investigation of controllability is put into the context of two-dimensional creative pattern designs. Procedural models and the computation of designs offer novel approaches to create content and provide benefits over traditional manual manufacturing. However, providing control mechanisms that are intuitive to use and allow individual designs is an ongoing research challenge. For more complete and meaningful solutions aspects of both data-driven and procedural techniques are needed and must be merged into a unified whole.

The reviewed techniques could complement each other and we hope to have furthered the direction of bringing different approaches together and to have carefully analyzed and emphasized an artist-centered perspective. This is the basis for developing innovative tools that further the ability of artists to create and to creatively express themselves.

## Image References

1. Manuscripts and Archives Division, The New York Public Library. 1450 - 1475. Historiated initial and another coat of arms. http://digitalcollections.nypl.org/items/510d47da-e47a-a3d9-e040-e00a18064a99
2. 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
3. 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. https://digitalcollections.nypl.org/items/510d47db-bc92-a3d9-e040-e00a18064a99
4. Spencer Collection, The New York Public Library. 1910. Front doubleur. http://digitalcollections.nypl.org/items/8a6be0f9-3d78-b15e-e040-e00a180602c7
5. Agnieszka Murphy. 2018. Polish folk art. 123RF, https://de.123rf.com/lizenzfreie-bilder/29119380.html?&sti=nmw3eri7lnbl7fxnhil&mediapopup=29119380
6. The Miriam and Ira D. Wallach Division of Art, Prints and Photographs, The New York Public Library. 1877. Arabesques : mosquÃl'e cathÃl'drale de Qous : typan et Ãl'u- coinÃğons en faÃŕence (XVIe. siÃĺcle). https://digitalcollections.nypl.org/items/510d47d9-66dd-a3d9-e040-e00a18064a99
7. William Morris. 1876. African Marigold Printed Textile. Planet Art CD of royalty-free PD images William Morris: Selected Works. https://commons.wikimedia.org/wiki/File:Morris_African_Marigold_textile_drawing_1876.jpg
8. Colourbox. 2011. Frame with roses, Vector. https://www.colourbox.com/vector/frame-with-roses-vector-1286656

9. Colourbox. 2013. Illustration of frame in Ukrainian folk style, Vector. https://www.colourbox.com/vector/frame-vector-6826661
10. Izabela Rejke. 2011. Traditional Polish Folk Design. http://rejke.deviantart.com/art/Traditional-Polish-Folk-Design-192417774
11. Colourbox. 2013. Ornamental khokhloma oral postcard with seamless stripe, Vector. https://www.colourbox.com/vector/ornamental-khokhloma-oral-postcard-vector-8445572
12. Free Patterns Area. Laser cut wood ornament template. 2018. https://www.freepatternsarea.com/designs/geometric-decorative-islamic-art-ornament-vector-design/. CC-BY-4.0 Creative Commons License
13. Marcel's Kid Crafts. Celtic knot pattern. 2018. http://www.marcels-kid-crafts.com/celtic-knot-patterns.html. CC-BY-4.0 Creative Commons License.

## References


[ABS06] ARBRUZZO, EMILY, BRISEN, ALEXANDER, and SOLOMAN, JONATHAN D. *Decoration 306090*. Vol. 10. Princeton Architectural Press, 2006 6.

[ADBW16] ALIAGA, DANIEL G., DEMIR, İLKE, BENES, BEDRICH, and WAND, MICHAEL. "Inverse Procedural Modeling of 3D Models for Virtual Worlds". *ACM SIGGRAPH 2016 Courses*. SIGGRAPH '16. ACM, 2016, 16:1–16:316 1.

[AGYS14] ABDRASHITOV, RINAT, GUY, EMILIE, YAO, JIAXIAN, and SINGH, KARAN. "Mosaic: Sketch-based Interface for Creating Digital Decorative Mosaics". *Proceedings of the Joint Symposium on Computational Aesthetics, Non-Photorealistic Animation and Rendering, and Sketch-Based Interfaces and Modeling*. ACM, 2014, 5–10 16.

[AKA13] ALMERAJ, ZAINAB, KAPLAN, CRAIG S., and ASENTE, PAUL. "Patch-Based Geometric Texture Synthesis". *Proceedings of the Symposium on Computational Aesthetics*. New York, NY, USA: ACM, 2013, 15–19 9, 13.

[AKAL11] ALMERAJ, ZAINAB, KAPLAN, CRAIG S., ASENTE, PAUL, and LANK, EDWARD. "Towards Ground Truth in Geometric Textures". *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Non-Photorealistic Animation and Rendering*. ACM, 2011, 17–26 13.

[AMM19] ALVAREZ, LUIS, MONZÓN, NELSON, and MOREL, JEAN-MICHEL. "Interactive design of random aesthetic abstract textures by composition principles". *Leonardo* 0 (2019), 1–11 8.

[APS08] ANASTACIO, F., PRUSINKIEWICZ, P., and SOUSA, M. C. "Sketch-based Parameterization of L-systems Using Illustration-inspired Construction Lines". *Proceedings of the Eurographics Conference on Sketch-Based Interfaces and Modeling*. Eurographics Association, 2008, 119–126 15.

[AW08] ANDERSON, DUSTIN and WOOD, ZOË. "User driven two-dimensional computer-generated ornamentation". *Advances in Visual Computing*. Springer, 2008, 604–613 9, 14.

[BBT*06] BARLA, PASCAL, BRESLAV, SIMON, THOLLOT, JOËLLE, et al. "Stroke Pattern Analysis and Synthesis". *Computer Graphics Forum* 25.3 (2006), 663–671 9, 13.

[BD04] BOURQUE, ERIC and DUDEK, GREGORY. "Procedural Texture Matching and Transformation". *Computer Graphics Forum* 23.3 (2004), 461–468 6, 9, 11.

[Bod10] BODEN, MARGARET A. *Creativity and Art: Three Roads to Surprise*. Oxford University Press, 2010 4, 5.

[BŠMM11] BENES, B., ŠT'AVA, O., MĚCH, R., and MILLER, G. "Guided Procedural Modeling". *Computer Graphics Forum* 30.2 (2011), 325–334 6, 7, 9, 14, 16, 19.

[BWCS14] BENEDETTI, LUCA, WINNEMÖLLER, HOLGER, CORSINI, MASSIMILIANO, and SCOPIGNO, ROBERTO. "Painting with Bob: Assisted Creativity for Novices". *Proceedings of the ACM Symposium on User Interface Software and Technology*. ACM, 2014, 419–428 5.

[BWL18] BIAN, XIAOJUN, WEI, LI-YI, and LEFEBVRE, SYLVAIN. "Tile-Based Pattern Design with Topology Control". *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 1.1 (2018) 9, 11, 12, 19.

[BZ17] BARNES, CONNELLY and ZHANG, FANG-LUE. "A survey of the state-of-the-art in patch-based synthesis". *Computational Visual Media* 3.1 (2017), 3–20 7.

[CEW*08] CHEN, GUONING, ESCH, GREGORY, WONKA, PETER, et al. "Interactive Procedural Street Modeling". *ACM Transactions on Graphics* 27.3 (2008), 103:1–103:10 16.

[CFA16] CHEN, YILAN, FU, HONGBO, and AU, KIN CHUNG. "A Multi-level Sketch-based Interface for Decorative Pattern Exploration". *SIGGRAPH Asia Technical Briefs*. ACM, 2016, 26:1–26:4 16.

[CL14] CHERRY, ERIN and LATULIPE, CELINE. "Quantifying the Creativity Support of Digital Tools Through the Creativity Support Index". *ACM Transactions on Computer-Human Interaction* 21.4 (2014), 21:1–21:25 4, 5.

[CMK*14] CIMPOI, M., MAJI, S., KOKKINOS, I., et al. "Describing Textures in the Wild". *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014 21.

[CNX*08] CHEN, XUEJIN, NEUBERT, BORIS, XU, YING-QING, et al. "Sketch-based Tree Modeling Using Markov Random Field". *ACM Transactions on Graphics* 27.5 (2008), 109:1–109:9 15.

[CO11] CULLEN, B. and O'SULLIVAN, C. "Symmetry Hybrids". *Proceedings of the International Symposium on Computational Aesthetics in Graphics, Visualization, and Imaging*. ACM, 2011, 33–38 21.

[Cro93] CROMWELL, PETER R. "Celtic knotwork: Mathematical art". *The Mathematical Intelligencer* 15.1 (1993), 36–47 8.

[CSC12] CHEN, YU-SHENG, SHIE, JIE, and CHEN, LIEU-HEN. "A NPR System for Generating Floral Patterns based on L-System". *Bulletin of Networking, Computing, Systems, and Software* 1.1 (2012) 8, 9, 15, 16.

[CXL19] CHEN, MINGHAI, XU, FAN, and LU, LIN. "Manufacturable pattern collage along a boundary". *Computational Visual Media* 5 (2019) 13.

[CYZL14] CHI, MING-TE, YAO, CHIH-YUAN, ZHANG, EUGENE, and LEE, TONG-YEE. "Optical illusion shape texturing using repeated asymmetric patterns". *The Visual Computer* 30.6 (2014), 809–819 8.

[CZX*16] CHEN, WEIKAI, ZHANG, XIAOLONG, XIN, SHIQING, et al. "Synthesis of filigrees for digital fabrication". *ACM Transactions on Graphics* 35.4 (2016), 98 9, 12.

[DHF*17] DETERDING, SEBASTIAN, HOOK, JONATHAN, FIEBRINK, REBECCA, et al. "Mixed-Initiative Creative Interfaces". *Proceedings of the CHI Conference Extended Abstracts on Human Factors in Computing Systems*. ACM, 2017, 628–635 4, 21.

[DKMI13] DIVERDI, S., KRISHNASWAMY, A., MĚCH, R., and ITO, D. "Painting with Polygons: A Procedural Watercolor Engine". *IEEE Transactions on Visualization and Computer Graphics* 19.5 (2013), 723–735 15.

[DLW81] DUNHAM, DOUGLAS, LINDGREN, JOHN, and WITTE, DAVID. "Creating Repeating Hyperbolic Patterns". *SIGGRAPH Computer Graphics* 15.3 (1981), 215–223 8.

[DS13] DOYLE, RICHARD BRIAN and SEMWAL, SUDHANSHU K. "Computational Celtic Canvas for Zoomorphs and Knotworks". *Proceedings of the 21st International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*. 2013 8, 17.

[DSJ19] DAVISON, TIMOTHY, SAMAVATI, FARAMARZ, and JACOB, CHRISTIAN. "Interactive example-palettes for discrete element texture synthesis". *Computers & Graphics* 78 (2019), 23–36 15.

[DWLS17] DONG, JUNYU, WANG, LINA, LIU, JUN, and SUN, XIN. "A Procedural Texture Generation Framework Based on Semantic Descriptions". (2017). arXiv:1704.04141 [cs.CV] 21.

[EL20] ETIENNE, JIMMY and LEFEBVRE, SYLVAIN. "Procedural Band Patterns". *Symposium on Interactive 3D Graphics and Games*. I3D '20. ACM, 2020 13.

[EMP*02] EBERT, DAVID S., MUSGRAVE, F. KENTON, PEACHEY, DARWYN, et al. *Texturing and Modeling: A Procedural Approach*. 3rd. Morgan Kaufmann Publishers Inc., 2002 6, 7.

[ESP08] ETEMAD, KATAYOON, SAMAVATI, FARAMARZ F., and PRUSINKIEWICZ, PRZEMYSLAW. "Animating Persian Floral Patterns". *Proceedings of the Eurographics Conference on Computational Aesthetics in Graphics, Visualization and Imaging*. Eurographics Association, 2008, 25–32 8.

[EVC*15] EMILIEN, ARNAUD, VIMONT, ULYSSE, CANI, MARIE-PAULE, et al. "WorldBrush: Interactive Example-based Synthesis of Procedural Virtual Worlds". *ACM Transactions on Graphics* 34.4 (2015), 106:1–106:11 15.

[FMD18] FRICH, JONAS, MOSE BISKJAER, MICHAEL, and DALSGAARD, PETER. "Twenty Years of Creativity Research in Human-Computer Interaction: Current State and Future Directions". *Proceedings of the 2018 Designing Interactive Systems Conference*. ACM, 2018, 1235–1257 4.

[FMR*19] FRICH, JONAS, MACDONALD VERMEULEN, LINDSAY, REMY, CHRISTIAN, et al. "Mapping the Landscape of Creativity Support Tools in HCI". *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. ACM, 2019, 1–18 4.

[GAD*20] GUEHL, PASCAL, ALLÈGRE, REMI, DISCHLER, JEAN-MICHEL, et al. "Semi-Procedural Textures Using Point Process Texture Basis Functions". *Computer Graphics Forum* (2020) 9, 11, 19.

[GALF17] GIESEKE, LENA, ASENTE, PAUL, LU, JINGWAN, and FUCHS, MARTIN. "Organized Order in Ornamentation". *Proceedings of the Symposium on Computational Aesthetics*. ACM, 2017, 4:1–4:9 1, 2, 6, 9, 12, 14, 16, 17, 19, 20.

[GBLM16] GUERRERO, PAUL, BERNSTEIN, GILBERT, LI, WILMOT, and MITRA, NILOY J. "PATEX: Exploring Pattern Variations". *ACM Transactions on Graphics* 35.4 (2016), 48:1–48:13 9, 17, 20, 21.

[GD10] GILET, G. and DISCHLER, J-M. "An Image-Based Approach for Stochastic Volumetric and Procedural Details". *Computer Graphics Forum* 29.4 (2010), 1411–1419 6, 9, 10.

[Gda17] GDAWIEC, KRZYSZTOF. "Procedural Generation of Aesthetic Patterns from Dynamics and Iteration Processes". *International Journal of Applied Mathematics and Computer Science* 27.4 (2017), 827–837 8.

[GDG12a] GILET, G., DISCHLER, J-M., and GHAZANFARPOUR, D. "Multiple kernels noise for improved procedural texturing". *The Visual Computer* 28.6 (2012), 679–689 9–11.

[GDG12b] GILET, G., DISCHLER, J-M., and GHAZANFARPOUR, D. "Multi-scale Assemblage for Procedural Texturing". *Computer Graphics Forum* 31.7 (2012), 2117–2126 9, 10.

[GGP*19] GALIN, ERIC, GUÉRIN, ERIC, PEYTAVIE, ADRIEN, et al. "A Review of Digital Terrain Modeling". *Computer Graphics Forum* 38.2 (2019), 553–577 1.

[GJB*20] GUO, JIANWEI, JIANG, HAIYONG, BENES, BEDRICH, et al. "Inverse Procedural Modeling of Branching Structures by Inferring L-Systems". *ACM Transactions on Graphics* 39.5 (2020) 6, 9, 16.

[GKHF14] GIESEKE, L., KOCH, S., HAHN, J.-U., and FUCHS, M. "Interactive Parameter Retrieval for Two-Tone Procedural Textures". *Computer Graphics Forum* 33.4 (2014), 71–79 9–11.

[GLLD12] GALERNE, BRUNO, LAGAE, ARES, LEFEBVRE, SYLVAIN, and DRETTAKIS, GEORGE. "Gabor Noise by Example". *ACM Transactions on Graphics* 31.4 (2012), 73:1–73:9 9, 10.

[GLM17] GALERNE, B., LECLAIRE, A., and MOISAN, L. "Texton Noise". *Computer Graphics Forum* (2017) 10.

[GSDC17] GUINGO, GEOFFREY, SAUVAGE, BASILE, DISCHLER, JEAN-MICHEL, and CANI, MARIE-PAULE. "Bi-Layer textures: a Model for Synthesis and Deformation of Composite Textures". *Computer Graphics Forum* 36.4 (2017), 111–122 9, 10.

[GSK12] GULATI, VISHAL, SINGH, KULWANT, and KATYAL, PUNEET. "A CAD Paradigm for Generating Woodworking Motifs". 47 (2012), 38–40 8.

[GSV*14] GILET, GUILLAUME, SAUVAGE, BASILE, VANHOEY, KENNETH, et al. "Local Random-phase Noise for Procedural Texturing". *ACM Transactions on Graphics* 33.6 (2014), 195:1–195:11 9, 10.

[GTS10] GULATI, VISHAL, TANDON, PUNEET, and SINGH, HARI. "A CAD Paradigm to Produce Zillij Style of Geometrical Patterns for Wooden Carvings". *International Journal of Computer Applications* 3.3 (2010), 28–35 8.

[HB95] HEEGER, DAVID J. and BERGEN, JAMES R. "Pyramid-based Texture Analysis/Synthesis". *Proceedings of the SIGGRAPH Conference on Computer Graphics and Interactive Techniques*. ACM, 1995, 229–238 7.

[HDR19] HU, YIWEI, DORSEY, JULIE, and RUSHMEIER, HOLLY. "A Novel Framework for Inverse Procedural Texture Modeling". *ACM Transactions on Graphics* 38.6 (2019) 9, 11.

[Hee19] HEER, JEFFREY. "Agency plus automation: Designing artificial intelligence into interactive systems". *Proceedings of the National Academy of Sciences* 116.6 (2019), 1844–1850 21.

[HF04] HAVEMANN, S. and FELLNER, D. "Generative parametric design of Gothic window tracery". *Proceedings Shape Modeling Applications*. 2004, 350–353 8.

[HLT*09] HURTUT, T., LANDES, P.-E., THOLLOT, J., et al. "Appearance-guided Synthesis of Element Arrangements by Example". *Proceedings of the International Symposium on Non-Photorealistic Animation and Rendering*. ACM, 2009, 51–60 9, 13.

[HMVI13] HENDRIKX, MARK, MEIJER, SEBASTIAAN, VAN DER VELDEN, JOERI, and IOSUP, ALEXANDRU. "Procedural Content Generation for Games: A Survey". *ACM Transactions on Multimedia Computing, Communications, and Applications* 9.1 (2013), 1:1–1:22 1, 7.

[HS12] HAMEKASI, NADER and SAMAVATI, FARAMARZ. "Designing Persian Floral Patterns using Circle Packing". *Proceedings of the International Conference on Computer Graphics Theory and Applications*. 2012, 135–142 8.

[HWYZ20] HSU, CHEN-YUAN, WEI, LI-YI, YOU, LIHUA, and ZHANG, JIAN JUN. "Autocomplete Element Fields". *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. ACM, 2020, 1–13 9, 14, 15, 19, 20.

[HXF*19] HU, ZHONGYUAN, XIE, HAORAN, FUKUSATO, TSUKASA, et al. "Sketch2VF: Sketch-based flow design with conditional generative adversarial network". *Computer Animation and Virtual Worlds* 30.3-4 (2019), e1889 17.

[Iga10] IGARASHI, YUKI. "DECO: A Designing Editor for Line Stone Decoration". *ACM SIGGRAPH Posters*. ACM, 2010, 34:1–34:1 16.

[IMIM08] IJIRI, TAKASHI, MĚCH, RADOMÍR, IGARASHI, TAKEO, and MILLER, GAVIN. "An Example-based Procedural System for Element Arrangement". *Computer Graphics Forum* 27.2 (2008), 429–436 9, 13, 16.

[Ise16] ISENBERG, TOBIAS. "Interactive NPAR: What type of tools should we create?": *Proceedings of the International Symposium on Non-Photorealistic Animation and Rendering*. 2016, 89–96 4.

[JBMR18] JACOBS, JENNIFER, BRANDT, JOEL R., MĚCH, RADOMÍR, and RESNICK, MITCHEL. "Dynamic Brushes: Extending Manual Drawing Practices with Artist-Centric Programming Tools". *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*. ACM, 2018, D316:1–D316:4 6, 9, 15.

[Jul81] JULESZ, BÉLA. "Textons, the elements of texture perception, and their interactions". *Nature* 290 (1981), 91–97 21.

[Kap02] KAPLAN, CRAIG STEVEN. "Computer Graphics and Geometric Ornamental Design". University of Washington. PhD thesis. 2002 8.

[KB18] KHAMJANE, AZIZ and BENSLIMANE, RACHID. "Generating Islamic Quasi-Periodic Patterns: A New Method". *ACM Journal on Computing and Cultural Heritage* 11.3 (2018), 13:1–13:18 8.

[KC03] KAPLAN, MATTHEW and COHEN, ELAINE. "Computer Generated Celtic Design". *Proceedings of the Eurographics Workshop on Rendering*. Eurographics Association, 2003, 9–19 8, 17.

[KH17] KANG, HYEONGYEOP and HAN, JUNGHYUN. "Feature-preserving Procedural Texture". *The Visual Computer* 33.6-8 (2017), 761–768 9, 10.

[KIZD12] KAZI, RUBAIAT HABIB, IGARASHI, TAKEO, ZHAO, SHENGDONG, and DAVIS, RICHARD. "Vignette: Interactive Texture Design and Manipulation with Freeform Gestures for Pen-and-ink Illustration". *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2012, 1727–1736 9, 16, 20.

[KS04] KAPLAN, CRAIG S. and SALESIN, DAVID H. "Islamic Star Patterns in Absolute Geometry". *ACM Transactions on Graphics* 23.2 (2004), 97–119 8.

[LBM*20] LI, JINGYI, BRANDT, JOEL, MECH, RADOMÍR, et al. "Supporting Visual Artists in Programming through Direct Inspection and Control of Program Execution". *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. ACM, 2020, 1–12 15, 20.

[LBMH19] LI, YIFEI, BREEN, DAVID E., MCCANN, JAMES, and HODGINS, JESSICA. "Algorithmic Quilting Pattern Generation for Pieced Quilts". *Proceedings of Graphics Interface 2019*. Canadian Information Processing Society, 2019 9, 11.

[LBW*14] LU, JINGWAN, BARNES, CONNELLY, WAN, CONNIE, et al. "DecoBrush: Drawing Structured Decorative Patterns by Example". *ACM Transactions on Graphics* 33.4 (2014), 90:1–90:9 9, 15, 16.

[LBZ*11] LI, YUANYUAN, BAO, FAN, ZHANG, EUGENE, et al. "Geometry Synthesis on Surfaces Using Field-Guided Shape Grammars". *IEEE Transactions on Visualization and Computer Graphics* 17.2 (2011), 231–243 9, 12.

[LDC*15] LIU, JUN, DONG, JUNYU, CAI, XIAOXU, et al. "Visual Perception of Procedural Textures: Identifying Perceptual Dimensions and Predicting Generation Models". (2015). PLoS ONE 10(6): e0130335 21.

[LFA*15] LUKÁČ, MICHAL, FIŠER, JAKUB, ASENTE, PAUL, et al. "Brushables: Example-based Edge-aware Directional Texture Painting". *Computer Graphics Forum* 34.7 (2015), 257–268 15.

[LGD*18] LIU, JUN, GAN, YANHAI, DONG, JUNYU, et al. "Perception-driven procedural texture generation from examples". *Neurocomputing* 291 (2018), 21–34 21.

[LGH13] LANDES, PIERRE-EDOUARD, GALERNE, BRUNO, and HURTUT, THOMAS. "A Shape-Aware Model for Discrete Texture Synthesis". *Proceedings of the Eurographics Symposium on Rendering*. Eurographics Association, 2013, 67–76 9, 14.

[LHVT17] LOI, HUGO, HURTUT, THOMAS, VERGNE, ROMAIN, and THOLLOT, JOELLE. "Programmable 2D Arrangements for Element Texture Design". *ACM Transactions on Graphics* 36.3 (2017), 27:1–27:17 6, 9, 12.

[LHW*06] LIN, WEN-CHIEH, HAYS, J., WU, CHENYU, et al. "Quantitative Evaluation of Near Regular Texture Synthesis Algorithms". *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2006, 427–434 10.

[LLC*10] LAGAE, A., LEFEBVRE, S., COOK, R., et al. "State of the Art in Procedural Noise Functions". *Proceedings of the Conference of the European Association for Computer Graphics*. The Eurographics Association, 2010 7, 8.

[LLD12a] LASRAM, ANASS, LEFEBVRE, SYLVAIN, and DAMEZ, CYRILLE. "Procedural Texture Preview". *Computer Graphics Forum* 31.2 (2012), 413–420 6.

[LLD12b] LASRAM, ANASS, LEFEBVRE, SYLVAIN, and DAMEZ, CYRILLE. "Scented Sliders for Procedural Textures". *Proceedings of the Conference of the European Association for Computer Graphics*. The Eurographics Association, 2012 6.

[LP00] LEFEBVRE, LAURENT and POULIN, PIERRE. "Analysis and Synthesis of Structural Textures". *Proceedings of the Graphics Interface Conference*. 2000, 77–86 9, 10.

[LVLD10] LAGAE, ARES, VANGORP, PETER, LENAERTS, TOON, and DUTRÉ, PHILIP. "Procedural Isotropic Stochastic Textures by Example". *Computers and Graphics* 34.4 (2010), 312–321 6, 10.

[MBS*11] MAHARIK, RON, BESSMELTSEV, MIKHAIL, SHEFFER, ALLA, et al. "Digital Micrography". *ACM Transactions on Graphics* 30.4 (2011), 100:1–100:12 16.

[MDLW15] MARTÍNEZ, JONÀS, DUMAS, JÉRÉMIE, LEFEBVRE, SYLVAIN, and WEI, LI-YI. "Structure and Appearance Optimization for Controllable Shape Design". *ACM Transactions on Graphics* 34.6 (2015) 11.

[MM10] MERRELL, PAUL and MANOCHA, DINESH. "Example-based Curve Synthesis". *Computers and Graphics* 34.4 (2010), 304–311 9, 15, 16.

[MM12] MĚCH, RADOMÍR and MILLER, GAVIN. "The Deco framework for interactive procedural modeling". *Journal of Computer Graphics Techniques* 1.1 (2012), 43–99. (Visited on 12/03/2015) 7, 9, 15, 17.

[MNN13] MATTHEWS, T., NIXON, M. S., and NIRANJAN, M. "Enriching Texture Analysis with Semantic Data". *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2013, 1248–1255 21.

[MOT99] MOUGHTIN, CLIFF, OC, TANER, and TIESDELL, STEVEN. *Urban Design: Ornament and Decoration*. Architectural Press, 1999 6.

[MWLT13] MA, CHONGYANG, WEI, LI-YI, LEFEBVRE, SYLVAIN, and TONG, XIN. "Dynamic Element Textures". *ACM Transactions on Graphics* 32.4 (2013), 90:1–90:10 13.

[MWT11] MA, CHONGYANG, WEI, LI-YI, and TONG, XIN. "Discrete Element Textures". *ACM Transactions on Graphics* 30.4 (2011), 62:1–62:10 9, 13.

[NSX*11] NAN, LIANGLIANG, SHARF, ANDREI, XIE, KE, et al. "Conjoining Gestalt Rules for Abstraction of Architectural Drawings". *ACM Transactions on Graphics* 30.6 (2011), 185:1–185:10 21.

[Ost98] OSTROMOUKHOV, VICTOR. "Mathematical tools for computer-generated ornamental patterns". Springer, 1998, 193–223 8.

[OWL*18] O'LEARY, JASPER, WINNEMÖLLER, HOLGER, LI, WILMOT, et al. "Charrette: Supporting In-Person Discussions Around Iterations in User Interface Design". *Proceedings of the CHI Conference on Human Factors in Computing Systems*. ACM, 2018, 535:1–535:11 21.

[Oxf17] OXFORD ENGLISH DICTIONARY ONLINE. http://www.oed.com. Accessed August 20, 2017. 2017 2.

[OZH15] OUYANG, P., ZHAO, W., and HUANG, X. "Beautiful Math, Part 5: Colorful Archimedean Tilings from Dynamical Systems". *IEEE Computer Graphics and Applications* 35.6 (2015), 90–96 8.

[Per85] PERLIN, KEN. "An Image Synthesizer". Vol. 19. 3. ACM, 1985, 287–296 2, 7.

[PGDG16] PAVIE, NICOLAS, GILET, GUILLAUME, DISCHLER, JEAN-MICHEL, and GHAZANFARPOUR, DJAMCHID. "Procedural texture synthesis by locally controlled spot noise". *Proceedings of International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*. 2016 9, 10.

[PH19] PEIHAN TU, DANI LISCHINSKI and HUANG, HUI. "Point Pattern Synthesis via Irregular Convolution". *Computer Graphics Forum (Proceedings of SGP 2019)* 38.5 (2019), 109–122 13.

[PHL*09] PALUBICKI, WOJCIECH, HOREL, KIPP, LONGAY, STEVEN, et al. "Self-organizing Tree Models for Image Synthesis". *ACM Transactions on Graphics* 28.3 (2009), 58:1–58:10 15.

[PLA*16] PHAN, H. Q., LU, J., ASENTE, P., et al. "Patternista: Learning Element Style Compatibility and Spatial Composition for Ring-based Layout Decoration". *Proceedings of the Joint Symposium on Computational Aesthetics and Sketch Based Interfaces and Modeling and Non-Photorealistic Animation and Rendering*. Eurographics Association, 2016, 79–88 14.

[Pru90] PRUSINKIEWICZ, PRZEMYSLAW. *The algorithmic beauty of plants*. Springer, 1990 7.

[PS06] PEDERSEN, HANS and SINGH, KARAN. "Organic Labyrinths and Mazes". *Proceedings of the 4th International Symposium on Non-Photorealistic Animation and Rendering*. ACM, 2006, 79–86 8.

[RMF*20] REMY, CHRISTIAN, MACDONALD VERMEULEN, LINDSAY, FRICH, JONAS, et al. "Evaluating Creativity Support Tools in HCI Research". *Proceedings of the 2020 ACM Designing Interactive Systems Conference*. ACM, 2020, 457–476 4.

[RMGH15] RITCHIE, DANIEL, MILDENHALL, BEN, GOODMAN, NOAH D., and HANRAHAN, PAT. "Controlling Procedural Modeling Programs with Stochastically-ordered Sequential Monte Carlo". *ACM Transactions on Graphics* 34.4 (2015), 105:1–105:11 7, 9, 12.

[RTHG16] RITCHIE, DANIEL, THOMAS, ANNA, HANRAHAN, PAT, and GOODMAN, NOAH D. "Neurally-guided Procedural Models: Amortized Inference for Procedural Graphics Programs Using Neural Networks". *Proceedings of the International Conference on Neural Information Processing Systems*. Curran Associates Inc., 2016, 622–630 12.

[Sal02] SALESIN, DAVID H. "Non-Photorealistic Animation and Rendering: 7 Grand Challenges". *Keynote talk at Second International Symposium on Non-Photorealistic Animation and Rendering*. 2002 4.

[ŠBM*10] ŠT'AVA, O., BENES, B., MĚCH, R., et al. "Inverse Procedural Modeling by Automatic Generation of L-systems". *Computer Graphics Forum* 29.2 (2010), 665–674 1, 7, 9, 13.

[Shn07] SHNEIDERMAN, BEN. "Creativity Support Tools: Accelerating Discovery and Innovation". *Communications of the ACM* 50.12 (2007), 20–32 4, 5.

[SKA18] SAPUTRA, REZA ADHITYA, KAPLAN, CRAIG S., and ASENTE, PAUL. "RepulsionPak: Deformation-Driven Element Packing with Repulsion Forces". *Proceedings of the 44th Graphics Interface Conference*. Canadian Human-Computer Communications Society, 2018, 10–17 9, 14, 19.

[SKAM17] SAPUTRA, REZA ADHITYA, KAPLAN, CRAIG S., ASENTE, PAUL, and MĚCH, RADOMÍR. "FLOWPAK: Flow-based Ornamental Element Packing". *Proceedings of the Graphics Interface Conference*. Canadian Human-Computer Communications Society, 2017, 8–15 6, 9, 14, 16, 17, 19, 20.

[SLD17] SHUGRINA, MARIA, LU, JINGWAN, and DIVERDI, STEPHEN. "Playful Palette: An Interactive Parametric Color Mixer for Artists". *ACM Transactions on Graphics* 36.4 (2017), 61:1–61:10 4.

[SP16] SANTONI, CHRISTIAN and PELLACINI, FABIO. "gTangle: A Grammar for the Procedural Generation of Tangle Patterns". *ACM Transactions on Graphics* 35.6 (2016), 182:1–182:11 6, 9, 12, 14, 19.

[SSG*17] SUMMERVILLE, ADAM, SNODGRASS, SAM, GUZDIAL, MATTHEW, et al. "Procedural Content Generation via Machine Learning". (2017). arXiv:1702.00539 [cs.AI] 7.

[SSTP15] SALVATI, GABRIELE, SANTONI, CHRISTIAN, TIBALDO, VALENTINA, and PELLACINI, FABIO. "MeshHisto: Collaborative Modeling by Sharing and Retargeting Editing Histories". *ACM Transactions on Graphics* 34.6 (2015), 205:1–205:10 21.

[STBB14] SMELIK, RUBEN M., TUTENEL, TIM, BIDARRA, RAFAEL, and BENES, BEDRICH. "A Survey on Procedural Modeling for Virtual Worlds". *Computer Graphics Forum* (2014) 1, 6, 7.

[Tak16] TAKAYAMA, JOE. "Medallions". *SIGGRAPH Asia 2016 Art Gallery*. ACM, 2016, 15:1–15:1 8.

[TGY*09] TALTON, JERRY O., GIBSON, DANIEL, YANG, LINGFENG, et al. "Exploratory Modeling with Collaborative Design Spaces". *ACM Transactions on Graphics* 28.5 (2009), 1–10 16, 21.

[TLL*11] TALTON, JERRY O., LOU, YU, LESSER, STEVE, et al. "Metropolis procedural modeling". *ACM Transactions on Graphics* 30.2 (2011), 1–14 7, 9, 12.

[TMNY04] TERRY, MICHAEL, MYNATT, ELIZABETH D., NAKAKOJI, KUMIYO, and YAMAMOTO, YASUHIRO. "Variation in Element and Action: Supporting Simultaneous Development of Alternative Solutions". *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2004, 711–718 4.

[TWO16] TODI, KASHYAP, WEIR, DARYL, and OULASVIRTA, ANTTI. "Sketchplore: Sketch and Explore with a Layout Optimiser". *Proceedings of the ACM Conference on Designing Interactive Systems*. ACM, 2016, 543–555 16.

[TWY*20] TU, PEIHAN, WEI, LI-YI, YATANI, KOJI, et al. "Continuous Curve Textures". *ACM Transactions on Graphics* 39.6 (2020) 9, 11, 12, 19.

[TYK*12] TALTON, JERRY, YANG, LINGFENG, KUMAR, RANJITHA, et al. "Learning Design Patterns with Bayesian Grammar Induction". *Proceedings of the ACM Symposium on User Interface Software and Technology*. ACM, 2012, 63–74 9, 13.

[TYSB11] TOGELIUS, J., YANNAKAKIS, G. N., STANLEY, K. O., and BROWNE, C. "Search-Based Procedural Content Generation: A Taxonomy and Survey". *IEEE Transactions on Computational Intelligence and AI in Games* 3.3 (2011), 172–186 1.

[War96] WARD, JAMES. *The principles of ornament*. Chapman and Hall, 1896 6.

[Wei06] WEISBERG, R.W. *Creativity: Understanding Innovation in Problem Solving, Science, Invention, and the Arts*. Wiley, 2006 4.

[Whi10] WHITEHEAD, JIM. "Toward Proccedural Decorative Ornamentation in Games". *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*. ACM, 2010, 9:1–9:4 8.

[WLKT09] WEI, LI-YI, LEFEBVRE, SYLVAIN, KWATRA, VIVEK, and TURK, GREG. "State of the art in example-based texture synthesis". *Proceedings of the Conference of the European Association for Computer Graphics*. The Eurographics Association, 2009, 93–117 7.

[WZS98] WONG, MICHAEL T., ZONGKER, DOUGLAS E., and SALESIN, DAVID H. "Computer-generated Floral Ornament". *Proceedings of the Conference on Computer Graphics and Interactive Techniques*. ACM, 1998, 423–434 6–9, 12, 14, 15.

[XCW14] XING, JUN, CHEN, HSIANG-TING, and WEI, LI-YI. "Autocomplete Painting Repetitions". *ACM Transactions on Graphics* 33.6 (2014), 172:1–172:11 9, 16, 20.

[XHC*18] XIA, HAIJUN, HENRY RICHE, NATHALIE, CHEVALIER, FANNY, et al. "DataInk: Direct and Creative Data-Oriented Drawing". *Proceedings of the CHI Conference on Human Factors in Computing Systems*. ACM, 2018, 223:1–223:13 16.

[XKG*16] XING, JUN, KAZI, RUBAIAT HABIB, GROSSMAN, TOVI, et al. "Energy-Brushes: Interactive Tools for Illustrating Stylized Elemental Dynamics". *Proceedings of the Symposium on User Interface Software and Technology*. ACM, 2016, 755–766 15, 16.

[XM09] XU, LING and MOULD, DAVID. "Magnetic Curves: Curvature-Controlled Aesthetic Curves Using Magnetic Fields". *Proceedings of the Eurographics conference on Computational Aesthetics in Graphics, Visualization and Imaging*. The Eurographics Association, 2009 9, 15, 16.

[XM15] XU, LING and MOULD, DAVID. "Procedural Tree Modeling with Guiding Vectors". *Computer Graphics Forum* 34.7 (2015), 47–56 16.

[XWSY15] XING, JUN, WEI, LI-YI, SHIRATORI, TAKAAKI, and YATANI, KOJI. "Autocomplete Hand-Drawn Animations". *ACM Transactions on Graphics* 34.6 (2015) 16.

[YCHK15] YUMER, MEHMET ERSIN, CHAUDHURI, SIDDHARTHA, HODGINS, JESSICA K., and KARA, LEVENT BURAK. "Semantic Shape Editing Using Deformation Handles". *ACM Transactions on Graphics* 34.4 (2015), 86:1–86:12 21.

[YM09] YEH, YI-TING and MĚCH, RADOMÍR. "Detecting Symmetries and Curvilinear Arrangements in Vector Art". *Computer Graphics Forum* 28.2 (2009), 707–716. (Visited on 06/10/2015) 9, 17.

[ZCT16] ZEHNDER, JONAS, COROS, STELIAN, and THOMASZEWSKI, BERNHARD. "Designing Structurally-sound Ornamental Curve Networks". *ACM Transactions on Graphics* 35.4 (2016), 99:1–99:10 8, 9, 11, 12, 19.

[ZJL14] ZHOU, SHIZHE, JIANG, CHANGYUN, and LEFEBVRE, SYLVAIN. "Topology-constrained Synthesis of Vector Patterns". *ACM Transactions on Graphics* 33.6 (2014), 215:1–215:11 9, 16.