User Driven Two-Dimensional Computer-Generated Ornamentation

Dustin Anderson and Zoë Wood

Abstract. Hand drawn ornamentation, such as floral or geometric patterns, is a tedious and time consuming task that requires skill and training in ornamental design principles and aesthetics. Ornamental drawings both historically and presently play critical roles in all things from art to architecture, and when computers handle the repetition and overall structure of ornament, considerable savings in time and money can result. Due to the importance of keeping an artist in the loop, we present an application, designed and implemented utilizing a user-driven global planning strategy, to help guide the generation of two-dimensional ornament. The system allows for the creation of beautiful, organic ornamental 2D art which follows a user-defined curve. We present the application and the algorithmic approaches used.

1 Introduction

Hand-drawn ornamentation, like that drawn in Figure 1, is a tedious and time consuming task that requires much skill and training in ornamental design principles and aesthetics. Ornamental drawings both historically and presently play critical roles in all things from architecture to art, and allowing computers to handle the repetition and tedium of ornamental generation allows for considerable time savings. Building on concepts from *Computer-Generated Floral Ornament* [2], we have created an application which allows users to generate 2D ornament that more strongly adheres to the ornamental design principles than in previous works. Due to the importance of keeping an artist in the loop, we present an application, designed and implemented utilizing a user-driven global planning strategy.

Ornamentalists use five principal techniques in conveying a perception of order: *repetition*, *balance*, *conformation to geometric constraints*, *growth*, and *conventionalization* [3,4,5]. In brief these principals are:

1. *Repetition:* Even a simple geometric mark, when repeated through translation, rotation, or scaling, can serve as the basis of an ornament.
2. *Balance:* The principle of balance requires that asymmetrical visual masses be made of equal "weight" [2].
3. *Conformation to Geometric Constraints:* A careful *fitting to boundaries* is a hallmark of ornament [6]. In addition, for structural integrity, *tangential junction* provides a powerful sense of physical support to an ornament.

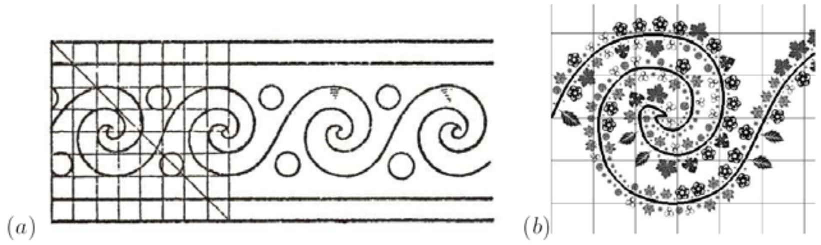

Fig. 1. (a) A physiographic wave ornament taken from [1]. (b) One of the wave segments created using our system.

4. *Growth:* Growth is a means of transporting design into new regions and continuing patterns. Especially for floral ornament, growth is an essential aspect of creating organic looking ornament. Additionally, *intention* provides another avenue for artistic control, expressing growth with external influences taken into consideration, such as growth toward pre-placed flowers or guidance along a central vine.
5. *Conventionalization:* In ornament, conventionalization is the development of abstractions of natural form. When artists develop a conventionalization, they extract only the essential aspects of form and the result often is stylized and modified to be more aesthetic.

Of these principals, our application adheres to: *repetion*, *balance*, *growth*, and *geometric constraints*. Our system allows users to select when *repetition* will be used with radius-to-texture mappings, and *balancing* is a completely automated process. Our system's interactivity support the principal of *growth*, by allowing the user to guide an ornament's growth through *intention*. Our application allows the user to place a main curve and special user-placed polygons called no-draw regions where ornament may not exist, which are used to guide the overall structure of an ornament. Our system *conforms to these geometric constraints* and since our system carefully generates ornament elements along a user-defined curve, all generated ornament structures follow the principle of *tangential junction*. Tangential junction gives the overall ornament a sense of physical "strength" insofar as it seems to "hang together," unlike the ornament generated by the system in [2] which intentionally grows ornament with the goal of filling space. These features, coupled with utilizing an interactive user-define curve and no-draw region placement as a *global planning strategy* for ornament structure, allows ornamentalists to create beautiful and organic-looking ornamental 2D art with our system.

2 Related Work

Many areas of computer graphics are related to computer generated ornamentation with the most relevant work being done by Wong *et al.*[2]. Other early work that contributed to the field include: generating the 17 symmetry patterns within a plane [7], generating periodic tilings and patterns [8], synthesis of frieze patterns [9], and generation of flora using computers [10]. In addition, Beach and Stone introduced the idea of procedurally generating a simple repeating border pattern that is warped to follow the path of a spline in their paper on graphical style sheets [11], an idea that was expanded on by Hsu and Lee, who introduced the notion of *"skeletal strokes"* to warp vector clip art along a path [12,13]. Other areas of related work include L-systems for computer-generated growth [14], fractals and dynamical systems [6], computer generated Celtic design [15], and generative parametric design of gothic window tracery [16].

In the work by Wong *et al.* [2], a modern approach to generating floral ornament is presented, and the types of ornamentation are classified. The output from the system is called *"adaptive clip art"*. The implementation of the algorithm by Wong *et al.* first places the ornamental elements algorithmically using *proxies* to the actual geometry. A *growth model* handles the placement of the proxies, where new "growth" of the ornament is accomplished by applying rules from existing motifs into portions of the panel that are not yet populated. Artists are responsible for creating the actual geometry for each proxy, but the final placement of ornament element proxies is determined by the algorithm.

A significant contribution from the work by Wong *et al.* is that the system does not create ornaments using traditional botanical growth models such as L-systems[14]. Instead the growth model represents the *artist's process* in creating aesthetic stylized plant designs, and is not meant to mirror the growth of actual flora. Kaplan [6] points out that although much effort is given to describe the principles of ornamental design in the work by Wong *et al.*, the implementation of the system only loosely adheres to them. This technique appropriately deals with small areas, which are able to be ornamented in an aesthetic fashion. Larger areas, however, such as those in an architectural setting, would most likely fail to be aesthetically pleasing due to the lack of any sort of *global planning strategies* that would guide the growth of ornaments.

3 Overview and Algorithms

Our system is an application for use in the creation of two-dimensional ornamental drawings. In general, the system allows users to input the control points for a curve which defines the general underlying structure of an ornament. The curve is loaded into a buffer and then proxies are seeded along it according to user-defined controls. Proxy sizes are determined by user controls and geometric constraints. Once seeded, varying textures are mapped onto the primitive proxy geometry and displayed to the user. Texture variations are controlled by user's selected mapping of proxy size ranges to specific textures. At this point, the user can decide to balance the ornament or not. Furthermore, the user is allowed to define polygonal regions where ornament may not exist, further promoting the user's artistic control over the *global planning* of the ornament.

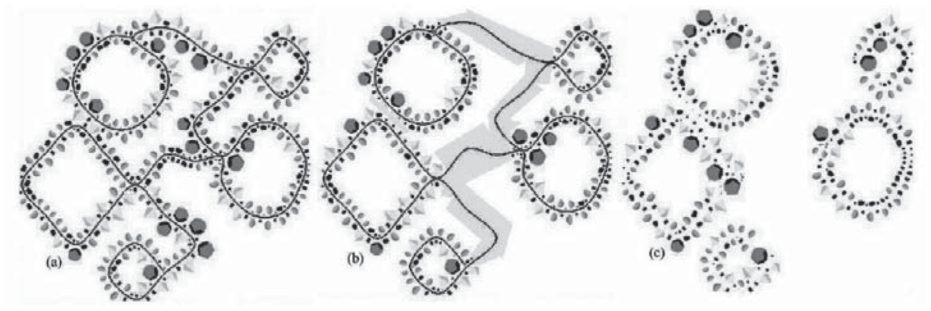

Fig. 2. The creation of seemingly multiple ornamentents from a single curve with radius-balanced group sizes 1:1. (a) The original ornament. (b) The ornament with no-draw regions active. (c) The final ornament (curve and no-draw regions hidden).

3.1 Goals

Our goal was to create a system that allowed for the direct, accurate, and interactive creation of two-dimensional ornamentation using global planning. Specifically we wanted users to be able to:

1. Create a fairly complex structural curve intuitively using the mouse
2. View the underlying structure and components of their ornament as it is created
3. Generate ornament elements that seem to "grow" *from* the user-defined structural curve
4. Compose a personalized ornament intuitively that adheres to the principles of ornamental design
5. Fine-tune a computer-generated ornament if desired, but also be able to create ornaments quickly without having to modify hundreds of controls

The following sections describe our algorithm in more detail and demonstrate how our algorithm achieves these goals.

3.2 Curve Representation

In order to achieve these goals, the system was designed with the user in mind and works in real time. Because *global planning* was the main methodology for creating a user-driven ornament, curve placement is essential. Curve points frequently sampled and connected with short lines were chosen over longer, straighter, and sharper line segments in order to achieve a more organic aesthetic. The underlying curve representation is a Catmull-Rom representation. Catmull-Rom curves allow for two directional changes at any given control point, which is crucial for giving the user the freedom to construct curves with varying size segments and varying curvatures. Our system allows for the placement of up to fifty control points via mouse input, satisfying the first goal of being able to *create a fairly complex structural curve intuitively using the mouse*. Our system

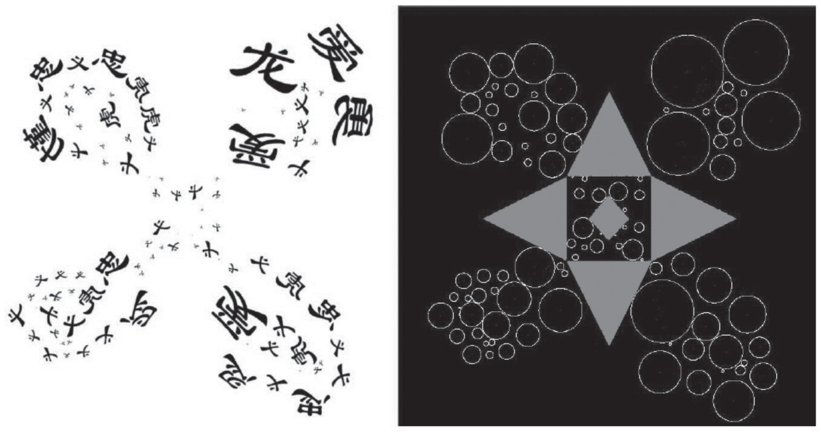

Fig. 3. A radius-balanced ornament with group sizes 1:1 with oriented texture elements avoiding no-draw regions. The no-draw regions can be seen in the buffer window (grey), and the curve is hidden.

uses 4th degree equations, which are preferred over higher degree equations for mathematical simplicity. Other curve representations could be used, however, we were pleased with the results using the Catmull-Rom curves. For a longer discussion of the curve representation please see [17]. Figure 2 shows an example of complex ornamentation using the curve placement provided by our application.

3.3 Viewing Underlying Ornament Structure

In order to address our second goal of being able to *view the underlying structure and components of an ornament as it is created*, the application includes two

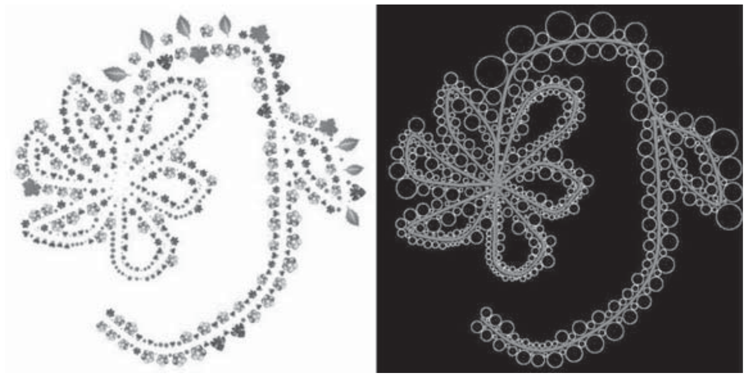

Fig. 4. A radius-balanced ornament with group sizes 1:1 and oriented *floral* texture elements. The interpolated curve is drawn as white.

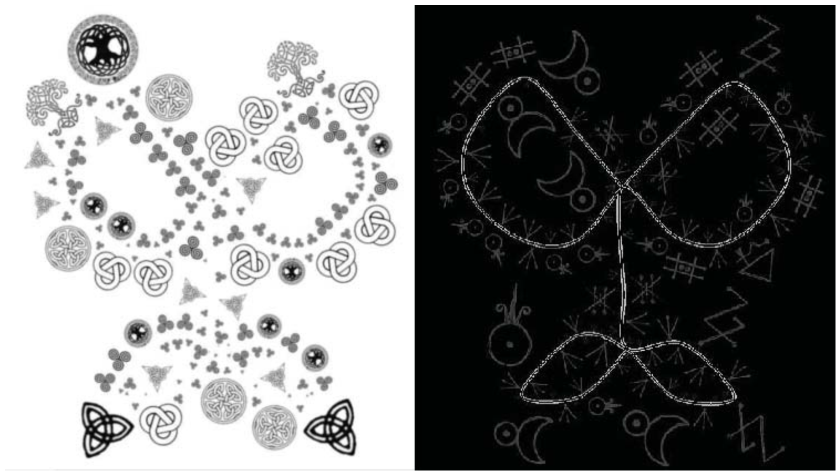

Fig. 5. Another example of ornamentation created using our application

windows viewing the ornatmention: a buffer window and an interactive window. Once the user defines the control points of the curve, the curve is drawn into the interactive window. The interactive window is the area of our application where the user enters input into the system via the mouse by placing control points and defining no-draw regions. The buffer window is where the underlying components of a user's ornament are shown in real-time as the ornament is modified. The user can choose to view the element proxies, control points, and/or the curve normals in the interactive window by turning on visibility through the application options. See Figures 3 and 6 for examples of the buffer window view and the interactive window. The interactive window is a reflection of the components in the buffer window, where proxies are mapped with textures and displayed as ornamental elements on screen.

Before proxies have been calculated and placed around the curve, the curve is scanned into a two-dimensional array called the *image buffer*. Each pixel that matches the user-defined curve color and/or outline color is considered a buffer hit, and its value in the buffer is set to a constant value representative of existing geometry. All other pixels are loaded into the image buffer as EMPTY. The mapping of the curve into the image buffer is a critical preparatory step for the seeding algorithm which calculates the placement of proxies that both do not overlap the curve, and best fill up the space.

3.4 Seeding Algorithms

Generating ornament *along* the user-placed curve creates an ornament with a strong sense of tangential junction. This satisfies the third goal of being able to create ornament where elements will "grow" from the user-defined structural curve, as seen in Figure 4. The algorithm executes as follows:

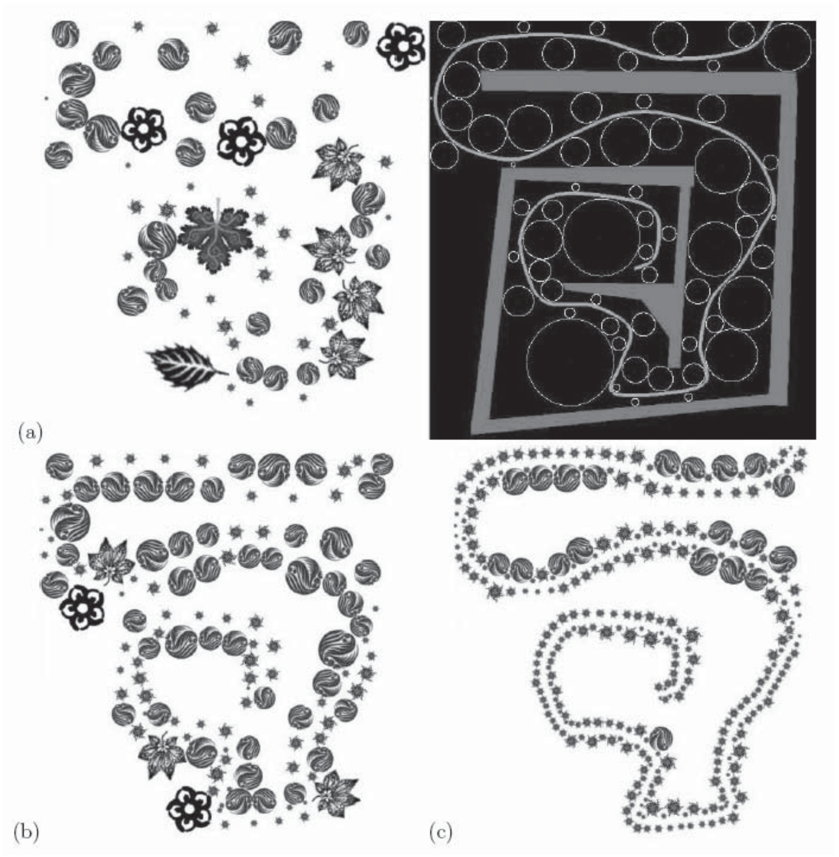

Fig. 6. In this progression, proxies begin to collect around the structural curve as sampling distance is decreased. **(a)** An ornament with the structural curve and no-draw regions hidden in the interactive window, but shown in the adjacent buffer window. **(b)** The sampling distance along the curve in (a) is decreased from 10 to 4. **(c)** The curve in (b) is then linearly interpolated.

For each sampling point along the curve corresponding to the user-defined sampling distance, a normal is computed. This calculated normal points to the correct side (left or right) of the curve, determined by the group sizing controls the user has set. A new proxy center is then generated at the user-defined largest radius size away from the curve along the normal. At this point, intersections between the new proxy and the curve, any other proxy, and no-draw regions are tested for by indexing into the image buffer. If intersection occurs, the new proxy's radius is decreased by one pixel, and the center of the proxy is moved along the normal to keep the proxy as close to the curve as possible. The process of intersection testing, decreasing radius size, and moving proxies continues until no intersections occur. Once placement is final, the proxy is saved into the image buffer, and the corresponding element in the interactive window is texture mapped according to the user-defined radius-to-texture mappings.

3.5 The Balancing Algorithms and Error Checking

As defined earlier, *balancing* of an ornament requires that asymetrical visual masses be made of equal "weight." In our system, balancing can only occur when elements are placed along the curve, where the curve splits the drawing area into *left-space* and *right-space*. Element placement along the curve, however, is able to be balanced by adjusting the "weight" of every element on one side of the curve with the elements on the other, either by balancing all proxy radii *or* by balancing the areas within each proxy.

Various methods of balancing elements are possible, including balancing by radius, balancing by area, and balancing by texture map density. In the current implementation of the system we explored balancing by radius and area. In general, balancing is done by calculating the sum of all proxy weights (for example: radii) on the left of the curve, the sum of all proxy weights on the right of the curve, and decreasing proxy weights accordingly to make the larger sum equal to the smaller sum. As long as balancing is possible, this algorithm is invoked to balance the current ornament. Figure 4 depicts a simple ornament, balanced by radius. Note that balancing may not be possible in certain circumstances where group sizes are far apart. When two weight totals can never become equal due to the minimum and maximum weight constraints (say radius growth size due to geometric contraints), a warning message is provided to the user.

4 Results and Conclusions

Using the work presented by Wong *et al.* in [2] as both a reference and a springboard for implementation ideas, our contributions give users a means of *globally planning* ornaments interactively in real-time. Our system satisfies the goals from Section 3.1. Through our efforts, an interactive computer application that allows users to produce beautiful, organic ornamental images now exists. The system allows users to select textual elements to decorate a user-defined curve, providing a means of *globally planning* an ornament's overall structure. We have shown several images created with the system, and more images created with the application can be seen in [17].

In our application, users have controls to modify:

- how the curve is drawn
- the placement, sampling distance, and sizes of proxies
- which components of the ornament are visible
- radius-to-texture mapping ranges
- if preset styles and/or color inversion are used
- the overall balancing of an ornament and element grouping sizes
- no-draw regions and their visibility

These controls allow users to completely personalize an ornament.
Here, we explain our contributions in more depth, and compare our work with the work in [2] where appropriate. Specifically, our work:

- Provides an interactive method for designing two-dimensional ornament including curve placement and texture selection and their mappings. Our system receives input through the front-end GUI, allowing users to exert artistic control over their ornament. Additionally, the ornament created with our system need not be limited by any given "theme" such as "floral" or "geometric" because of the radius-to-texture mappings that can be applied on-the-fly by users. The work done by [2] did not allow for real-time interaction with the ornamentation process.
- Presents a method to generate ornament based on an underlying curve. Inputs in [2] were predefined and were not real-time, ornament filled an arbitrary panel, and was not able to globally be directed or influenced by external sources. We have purposely kept the growth algorithms straight-forward and unobtrusive so that users can have mechanisms for directly and accurately laying down their global planning strategies.
- Helps users generate ornament that automatically adheres more closely to ornamental design principles. The system of [2] produces ornament that only loosely follows these principles. *Repetition* is controlled by radius-to-texture mappings, but is not fully controllable. *Balancing* an ornament is an automated process and is fully controllable, as is *growth* along the user-defined curve.The principle of *tangential junction* is also upheld during ornament creation and the user can *globally plan* their ornament through *intention.*
- Supplies pre-defined sets of textures and color mappings that define ornament "styles,". Although [2] presents several "styles" of ornament in their work, libraries of these styles were not accessible by users, and proxy geometry could not be changed on-the-fly. In our system, however, any RGB formatted texture can be loaded at any time. Furthermore, this capability does not restrict the ornament generated by our system to be floral in nature, as is the case in [2]. See Figure 3 for a non-floral example.

Overall, our system serves to augment the process of ornamentation by computationally managing ornament design structure while giving ornamentalists an interactive, real-time, direct, and accurate means to experiment without fear of wasting resources. With our application, users can create beautiful and personalized organic-looking ornament effectively and efficiently.

4.1 Future Work

Since two-dimensional ornamentation can be found on everything from fliers to the human body, the potential uses of our application are boundless. One of the key ways our application could be expanded is through improvements to the interface and interaction. A gesture-based means for creating strokes would allow users a very intuitive means of creating organic ornamentation. In addition, other future improvements include genetic algorithms for generation, 3D ornamentation, multiple curves, and no-draw regions as imported geometry, could improve the existing application. Lastly, only a small group of user's have had the opportunity to give us feedback on our system. Users reported that the system is "fun and easy" to use, and the controls are simple enough that users were easily able to design a personalized ornament within a few minutes, however, a full blown user study would further improve our application.

References

1. Meyer, F.S.: Handbook of Ornament. Dover Publications, New York (1957)
2. Wong, M.T., Zongker, D.E., Salesin, D.H.: Computer-generated floral ornament. In: SIGGRAPH 1998, pp. 423–434 (1998)
3. Day, L.F.: Nature and Ornament V1: Nature the Raw Material of Design 1909. Kessinger Publishing Company, New York (2007)
4. Ward, J.: The Principles of Ornament. Scribner, New York (1896)
5. Jones, O.: The Grammar of Ornament. DK ADULT, London (1910)
6. Kaplan, C.S.: Computer Graphics and Geometric Ornamental Design. PhD thesis, University of Washington (2002)
7. Alexander, H.: The computer/plotter and the 17 ornamental design types. In: SIGGRAPH 1975, pp. 160–167 (1975)
8. Grünbaum, B., Shephard, G.C.: Tilings and Patterns. W. H. Freeman & Co., New York (1986)
9. Glassner, A.: Frieze groups. IEEE Comp. Graphics and Applications, 78–83 (1996)
10. Smith, A.R.: Plants, fractals, and formal languages. In: SIGGRAPH 1984 (1984)
11. Beach, R., Stone, M.: Graphical style towards high quality illustrations. SIGGRAPH Comput. Graph. 17, 127–135 (1983)
12. Hsu, S.C., Lee, I.H.H., Wiseman, N.E.: Skeletal strokes. In: UIST 1993 (1993)
13. Hsu, S.C., Lee, I.H.H.: Drawing and animation using skeletal strokes. In: SIGGRAPH 1994, pp. 109–118 (1994)
14. Prusinkiewicz, P., Lindenmayer, A.: The algorithmic beauty of plants. Springer, New York (1990)
15. Kaplan, M., Cohen, E.: Computer generated celtic design. In: EGRW 2003, pp. 9–19. Eurographics Association (2003)
16. Kaplan, M., Cohen, E.: Generative parametric design of gothic window tracery. In: Shape Modeling International 2004, pp. 350–353 (2004)
17. Anderson, D.: Two-dimensinal computer-generated ornamentation using a user-driven global planning strategy. Technical Report CPSLO-CSC-08-02, California Polytechnic State University (2008)