TWO-DIMENSIONAL COMPUTER-GENERATED ORNAMENTATION USING A USER-DRIVEN GLOBAL PLANNING STRATEGY

A Thesis

Presented to

the Faculty of California Polytechnic State University

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Computer Science

by

Dustin Robert Anderson

AUTHORIZATION FOR REPRODUCTION OF MASTER'S THESIS

I reserve the reproduction rights of this thesis for a period of seven years from the date of submission. I waive reproduction rights after the time span has expired.

_____

Signature

_____

Date

APPROVAL PAGE

TITLE: Two-Dimensional Computer-Generated Ornamentation Using a User-Driven Global Planning Strategy

AUTHOR: Dustin Robert Anderson

DATE SUBMITTED: June 2007

Dr. Zoë Wood
Advisor or Committee Chair _____
Signature

Dr. Franz Kurfess
Committee Member _____
Signature

Dr. Chris Buckalew
Committee Member _____
Signature

Abstract

Two-Dimensional Computer-Generated Ornamentation Using a User-Driven Global Planning Strategy
Dustin Robert Anderson

Hand drawn ornamentation, such as floral or geometric patterns, 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 art to architecture; however, little work has been done in exploring their algorithmic and interactive generation. The field of computer graphics offers many algorithmic possibilities for assisting an artist in creating two-dimensional ornamental art. When computers handle the repetition and overall structure of ornament, considerable savings in time and money can result. Today, the few existing computer algorithms used to generate 2D ornament have over-generalized and over-simplified the process of ornamentation, resulting in the substitution of limited amounts of generic and static "clip art" for once personalized artistic innovations.

Two possible approaches to computational ornamentation exist: interactive tools give artists instant feedback on their work, while non-interactive programs can carry out complex and sometimes lengthy computations to produce mathematically precise ornamental compositions. Due to the importance of keeping an artist in the loop for the production of ornamentation, 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, the algorithmic approaches used, and the potential uses of this application.

Key terms: ornamentation, computer graphics, art, algorithmic generation, interactive application

Acknowledgments

A special thank you goes out to to Zoë Wood who has supported me when I most needed it, and who not only sparked my passion in computer graphics, but helped it flourish. For her constant encouragement, her insights, and her optimism — I cannot thank you enough.

And—although I do not know a single one of them—thank you to Wong *et al.* cited in [48] from ten years ago for braving new territory and providing the intellectual springboard for my own work.

I would like to thank those who have believed in me throughout the years, and who have always inspired me to do more than I thought I could. Thank you to my parents, grandparents, and sister, for always believing in me.

For those who have taught me imagination and passion will always win the day.

> "There are no such things as limits to growth, because there are no limits to the human capacity for intelligence, imagination, and wonder."
>
> –*Ronald Reagan*

Contents

List of Figures **viii**

1 Motivation **1**
1.1 Problem . . . 1
1.2 Project Overview . . . 3

2 Background **6**
2.1 Ornamentation in History . . . 6
2.1.1 Early Ornamentation . . . 6
2.1.2 Ornamentation in Architecture . . . 8
2.1.3 Ornamentation Today . . . 11

3 Definitions **13**
3.1 Principles of Ornamental Design . . . 15
3.1.1 Repetition . . . 15
3.1.2 Balance . . . 16
3.1.3 Conformation to Geometric Constraints . . . 16
3.1.4 Growth . . . 17
3.1.5 Conventionalization . . . 18
3.2 System Terminology . . . 21

4 Related Work **22**
4.1 Groundwork for Computer Generated Ornamentation (1970+) . . . 23
4.2 L-Systems, Computer-Generated Growth (1968) . . . 24
4.3 Fractals and Dynamical Systems (∼1980) . . . 25
4.4 Computer Generated Floral Ornament (∼1998) . . . 27
4.5 Computer Generated Celtic Design (∼2003) . . . 28
4.6 Generative Parametric Design of Gothic Window Tracery (∼2004) . . . 31

5 Overview and Algorithms **33**
5.1 Goals . . . 33
5.2 Achieving Goals . . . 34
5.2.1 Curve Algorithms and the Selection Process . . . 35
5.2.2 Loading the Image Buffer . . . 40

5.2.3 Seeding Algorithms . . . 42
5.2.4 Customizable and Stylish Ornament . . . 46
5.2.5 The Balancing Algorithms and Error Checking . . . 48
5.3 Review . . . 53

6 The System Interface **54**
6.1 The Controls Window . . . 56
6.2 The Texture Customization Window . . . 60
6.3 The Interactive and Buffer Windows . . . 61
6.4 The Preset Styles Window . . . 62

7 Results and Conclusions **63**
7.1 Usability Feedback . . . 67
7.2 System Output: Ornamental Images . . . 68
7.3 Future Work . . . 73

8 Prelude **76**

Bibliography **77**

List of Figures

1.1 Hand-Drawn and Computer-Generated Ornament 1

2.1 Examples of Early Ornamentation 7
2.2 Ornamental Architecture 9
2.3 Modern Day Ornament Examples 11
2.4 The Tattoo: Self-Ornamentation 12

3.1 The Categorization of Ornament Elements 14
3.2 Tangential Junction and the Hierarchical Composition of an Ornament 17
3.3 Conventionalization of the *Glaucium Flavum* 19
3.4 M.C. Escher's "Plane Filling II" 20

4.1 Construction of the Snowflake Curve 24
4.2 The Mandelbrot Set 26
4.3 Computer-Generated Floral Ornament 29
4.4 Three Celtic Knot Examples 30
4.5 Gothic Window Tracery 31

5.1 A Fifty-Point Ornament Curve 35
5.2 A Cubic Bézier Curve 36
5.3 NURBS Curves of Varying Orders with a Catmull-Rom Curve Overlay 38
5.4 The Influence of a Control Point in a Series of Catmull-Rom Curves 41
5.5 A Scattering of Elements 43
5.6 Comparison: Oriented and Non-Oriented Elements 46
5.7 Ornament Balancing by Radius 49
5.8 Flowchart Diagram of the Radius Balancing Algorithm 51
5.9 Ornament Balancing by Area 52

6.1 A Screen Capture of Our Entire System 55
6.2 The Controls Window 56
6.3 Linear Interpolation of a User-Defined Catmull-Rom Curve 57
6.4 Interpolation and Sampling Distance Alteration for Fine-Tuning 59
6.5 The Texture Customization Window 61
6.6 The Interactive and Buffer Windows 62
6.7 The Preset Styles Window 62

7.1 **Results:** Ten Ornaments in Different Styles 68

Chapter 1

Motivation

1.1 Problem

Hand-drawn ornamentation, like that drawn in Figure 1.1a, is a tedious and time consuming task that requires much skill and training in ornamental design principles and aesthetics. In order to create an aesthetic design adhering to ornamental design principles, artists must spend time learning this craft. Regardless of the size of an ornament, this training is always required [25] in making any ornamental design decisions.

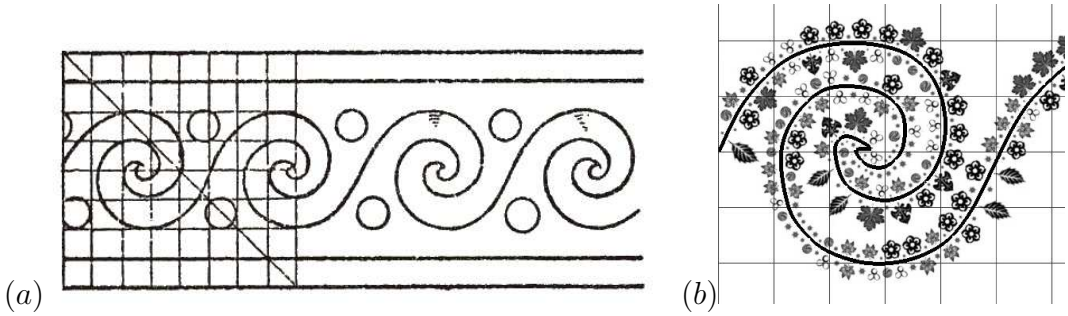


Figure 1.1: **(a)** A physiographic wave ornament taken from [29] **(b)** One of the wave segments from (a) rendered using our system

Ornamental drawings both historically and presently play critical roles in all things from architecture to art; however, little work has been done in exploring their algorithmic and interactive generation. The field of computer graphics offers many algorithmic possibilities for assisting an artist in creating two-dimensional ornamental art, but the few existing computer algorithms used to generate 2D ornament have over-generalized and over-simplified the process of ornamentation, resulting in the substitution of limited amounts of generic and static "clip art" for once personalized artistic innovations. Clip art is by no means dynamic, and even though millions of *static* images have been created for use on computers, these images are often hard to find, sometimes costly to acquire, and are rarely *exactly* what users are looking for. To solve this problem, a form of *adaptive clip art* [48] is required that allows users to dynamically control how ornament is constructed.

Providing ornamentalists with intuitive user interfaces to systems that generate *adaptive clip art* allows for more artistic freedom and experimentation without fear of wasting resources [48]. Interactive tools give artists instant feedback on their work, while even non-interactive programs can carry out complex and sometimes lengthy computations to produce mathematically precise ornamental compositions [23]. Allowing computers to handle the repetition and tedium of ornamental generation allows for considerable time savings which, depending on the artifact(s) being ornamented, may translate into financial savings as well for individuals or companies. And, although shallow training in the field of ornamentation may still be necessary, a system that is able to generate dynamic, interactive, and customizable ornament that adheres to the principles of ornamental design would help artisans by managing the structural overhead during creation, saving time and work, and promoting the artisan's ability to better allocate their time.

1.2 Project Overview

We take the approach of giving ornamentalists a means by which they can generate "adaptive clip art" that responds to user interaction with the system, a term and concept proposed by Wong *et al.* in their paper *Computer-Generated Floral Ornament* [48]. Building on concepts in this paper, this thesis and its accompanying application strives to provide ornamentalists with a useful tool to augment the process of ornament creation to solve the problems presented in Section 1.1. Specifically, our program helps users generate 2D ornament that more strongly adheres to the ornamental design principles than in previous works. Defined at length in Chapter 3, the five main principles of ornament design are:

1. Repetition
2. Balance
3. Conformation to Geometric Constraints
4. Growth
5. Conventionalization

Unlike the most related work done by [48] that generates ornament which strictly conforms to geometric constraints and is not directly interactive, the ornament generated by our system adheres strongly to the principles of *repetion*, *balance*, and *growth*, ensuring that the ornament grows within the *geometric constraints* of a preset square window size, and allows for *conventionalization* to a degree by allowing users to choose their own textures for ornament elements. Textures are externally created by users as BMP files confined to the size of 2^n, and texture mapped using the closest OpenGL-generated mipmap.

Additionally, our system's interactivity allows the user to guide an ornament's growth through *intention*, providing yet another avenue for artistic control. In-

tention helps drive the overall ornament design by allowing for external influences to influence structural properties. Here, a main curve the user places and special user-placed polygons called no-draw regions where ornament may not exist are used to guide the overall structure of an ornament. The interactive curve and no-draw region placement which structurally guides the ornament provides a user-driven **global planning strategy** for ornamentation.

Furthermore, since our system generates ornament elements along a user-defined curve where each element faces the curve, all generated ornament structures more strongly follow a principle known as *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 [48] which intentionally grew ornament with the goal of filling space. The sense that the ornament "hangs together" in our system increases as sampling is more frequently performed.

Also, our system allows users to select when *repetition* will be used with radius-to-texture mappings, and *balancing* is a completely automated process. These features, coupled with utilizing a user-define curve as a global planning strategy for ornament structure, allows ornamentalists to create beautiful and organic-looking ornamental 2D art with our system. Overall, our system satisfies the goals we defined at the onset of the project, presented in Section 5.1.

Even so, the problem space of generating all possible ornamental designs is simply enormous, as ornament is used today as a way to engage audiences in all forms of written communication. In our case, the opportunity for ornament design is still quite large, even when restricting the domain of ornamentation to along a single pre-drawn user-defined curve within a square panel. We will present a method for generating ornament that strongly adheres to the principles of ornamental design

(Chapter 3), and expect that the methods used here can be extrapolated to other areas of design such as those based in strict geometric domains, and others, and will have uses across more than just the discipline of computer science. We hope not to replace ornamentalists at all, but to allow the algorithmic creation of ornament to enrich the ornamentation process.

Chapter 2

Background

2.1 Ornamentation in History

From the moment humans could invent objects for themselves, they began adorning their creations with patterns and textures, "stamping" their works with the sigil of humanity. This ornamentat is among the oldest form of human expression, and is an ancient human endeavor. As humans evolved, so did their ornamental talents, and ornamentation throughout history is directly tied to a particular snapshot in time. Ornamentation (or lack thereof), is indicative of a particular place, time, culture, and attitude [23].

2.1.1 Early Ornamentation

Ornamental practices were already well developed by the Neolithic Age (Figure 2.1a) [3], and could be seen in everything from the eating vessels to the primitive clothing of the time. Later, virtually all of the commissioned writing of the Middle Ages was illuminated with ornament, and especially those manuscripts of the 13th century

(Figure 2.1b) stand out as being some of the most beautiful literary artifacts ever produced [48].

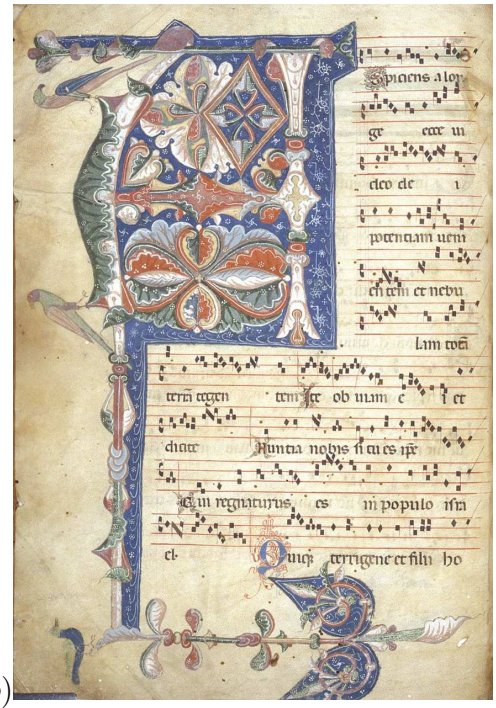
(*a*)  (*b*) 

Figure 2.1: **(a)** A jar from the Chinese Machiayao culture, ornamented with brown swirls and dots from the neolithic period, circa 3500 B.C. [8] **(b)** An illuminated manuscript from the 13th century depicting the letter 'A' with two parrots and another bird, created in Piacenza, Italy [33]. ©British Library Board. All Rights Reserved (Egerton 2977 f. 1v).

Another early example of ornamentation comes from the Celtic tribes of Europe, where the only written records of their civilization are the texts left by classical authors, the first of which appear circa 500 B.C. [12]. The Celts created elaborate knotwork and art, examples of which are shown in Figure 4.4, and today their intricate designs are the focus of the mathematical realm of knot theory [25]. In essence, Celtic ornamentation is an abstract non-imitative form of artwork, consisting of entangled threads which maintain a strict over-under alternating pattern between

every thread crossing. The discrete interlacing structure of Celtic knotwork is known as *plaitwork*, which appears carved on tombstones, etched into personal items, and in most forms of Celtic art [34]. Historical collections of some of the most famous knotwork can be found in the ornamented manuscripts of the British Isles such as the Book of Kells - the most famous illuminated bible of the Middle Ages [30] - and the Lindisfarne Gospels. Celtic knotwork is strictly geometric in nature, and so is often devoid of human and animal forms. Even so, when these forms do appear, they are intertwined *with* the plaitwork, where Celtic "ribbons" become animal limbs and hair. In this case, especially, interlacing is used as an aesthetic design [23].

Along with Celtic manuscripts, some of the earliest printed books were often ornamented by skilled artists, but around the turn of the 16th century, those skills were used less and less. By 1530, almost all ornamentation had disappeared [31].

Today, documents can be produced more easily and in greater quantities for the masses using computer technology and more advanced printing methodologies, but these methods of production have transformed the craft. Ornamentation has become over-generalized and over-simplified through technological modernization, and is not the craft of a skilled artisan as it once was. Static "clip art" has removed the need for skilled artisans, and replaced their innovations with pre-generated imagery. Standard word processing tools support ornamentation only through "clip art," and only to a small degree, if at all [23].

2.1.2 Ornamentation in Architecture

In architecture, ornament has historically played a critical and famous role. Especially for the Greeks and Romans, buildings often represented more than just their functional role to a society. Many were adorned with intricate decorative artwork

that was not only aesthetic, but served as mediums for telling stories or praising deities [43]. The Doric, Ionic, and Corinthian style pillars served different roles, each distinctly differentiable from the next. Today, however, most modern buildings, despite the help of sophisticated CAD tools, are largely devoid of such beautiful decorations [48]. Examples of ornamental architure are depicted in Figure 2.2.

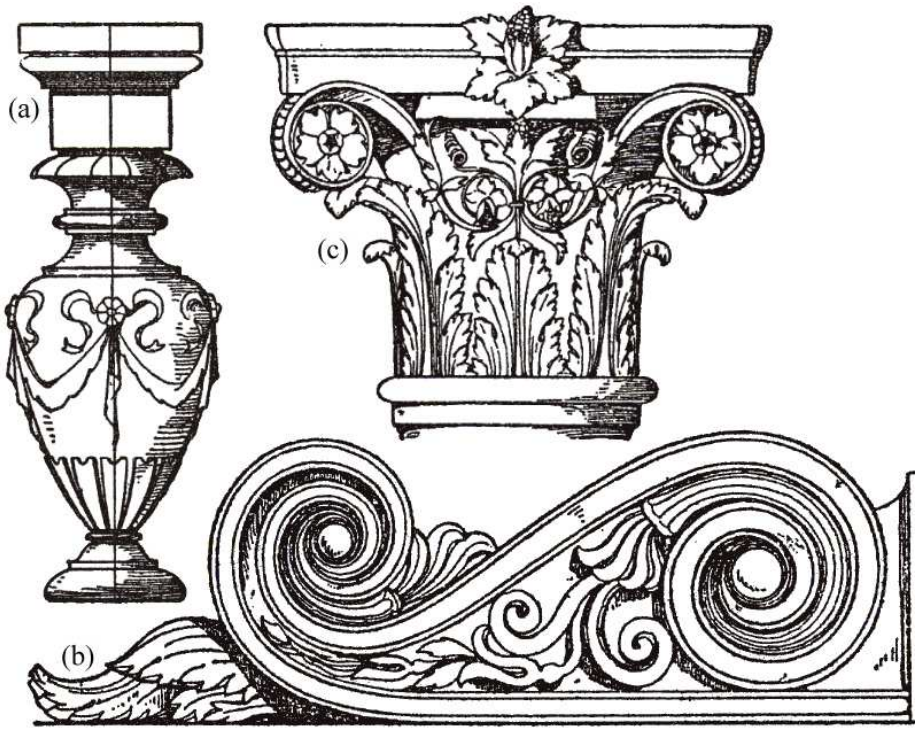


Figure 2.2: Examples of ornament found in architecture taken from [29]. **(a)** A large standing vase, etched with floral ornament **(b)** A wall ornament **(c)** A pillar's capital, ornamented with floral ornament

The Gothic eras, spanning from 1066 to around 1530 A.D., saw many architectural and engineering achievements as well, where gigantic and elaborate stone structures were erected, such as the well-known Notre Dame Cathedral in Paris, France. This, and the Cathedral of Seville located in Spain, the largest religious cathedral in the country and the third largest (by square footage) in the entire

world, boasts of intricate Gothic stone window tracery and ornamentation [45]. According to Martindale, architecture during this period was the most important and original art form during this time [28], where Gothic masons could build much larger and taller buildings than their Romanesque counterparts using flying buttresses and vaulted ceilings. Through the combination of only a few simple geometric patterns, Gothic architecture, and especially the window tracery of this era, exhibits quite complex geometric shape configurations in its ornamentation. Examples of real-world and computer generated window tracery are depicted in Figure 4.5.

However, history is not full of only those who appreciate the intricate details of ornamentation, and has seen individuals who scoff at the essence and origins of ornament, rejecting it in an attempt to cultivate a more forward-thinking, more "modern" way of life. *Horror vacui* — literally translated to "fear of the vacuum" — is the term used to characterize the human desire to adorn every blank wall and give every surface of a building decoration and texture [23]. Those who are opposed to this notion of *horror vacui* (or, the more positive outlook of *amor infiniti*, a term proposed by Gombrich [17]) are labeled, by Gombrich, as the "cult of restraint." The most recent revival of this cult came in the form of the modernist movement in architecture, the leaders of which were born in the 1880s. Its pioneers were architects like Mies van der Rohe, Le Corbusier, Gropius, and the Italian Futurists. These individuals rebelled against an overuse of ornament, and reveled in the beauty of technology and machines that promised to change the world for the better. To the modernists, ornament was tied to an archaic and crude way of life, and by rejecting it, would allow for the introduction of the new ideals of the twentieth century [21]. Architecture of this modern period has a distinctly spare, austere style with blank walls and sharp right angles, devoid of any ornamentation. [23].

2.1.3 Ornamentation Today

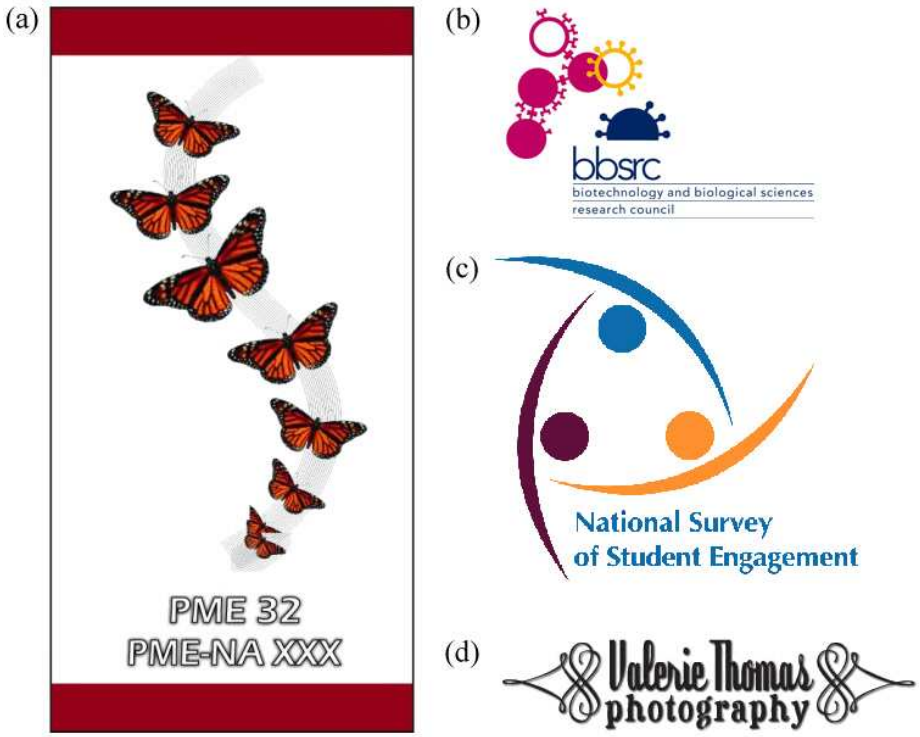


Figure 2.3: Examples of modern-day ornament in company logos. **(a)** The logo for the International Group for the Psychology of Mathematics Education (PME) **(b)** The Biotechnology and Biological Sciences Research Council logo **(c)** The National Survey of Student Engagement logo **(d)** A logo for Valerie Thomas' independent photography studio

Today, ornamentation is found everywhere — from subtle adornments on architectural structures, to commonplace tattoo parlors where individuals decorate *themselves* with ornament. It is estimated that one in seven individuals bears some sort of tattoo in North America, which accounts for over 45 million people in North America alone [46], but ornamentation can be found everywhere around the globe.

Figure 2.3 depicts examples of modern-day ornamentat from company and group logos.

Ornamentation is definitely not a dying art, and is used in everything from self-expression to business marketing. Ornamentation can be found in all style, from having Celtic origins, to following Asian stylistic drawings, to exhibiting a more tribal flare. Building décor, advertisements, web pages, and more, use ornamentation not only to engage audiences, but to distinguish themselves from all the rest. Not only do companies use ornament as a way to stand out in a crowd, but people do, too, decorating their bodies with permanent tattoos such as those in Figure 2.4.

(a)

(b)

Figure 2.4: **(a)** A physiographic wave form of ornamentation **(b)** A modern-day tattoo with strikingly similar properties to the traditional physiographic ornament in (a)

Chapter 3

Definitions

For our purposes, we will use the definitions proposed by Wong *et al.* where the term *ornament* serves to mean the aesthetic enrichment of the surfaces of man-made objects in ways not directly contributing to their functional utility [48]. In order to provide a sense of the richness and depth of the problems involved in creating ornament, we will briefly describe some of the principles that underlay its design. According to Kaplan, "ornament, like art, is hard to pin down, always evading definition on the wings of human ingenuity." As such, it is important to note that the literature often describes the structure and common features of ornament, yet no complete definition has yet to be provided that is universally agreed upon [23].

According to Wong *et al.*, the *elements* of ornamental design can be broken down into three broad categories [48], adapted from [29]:

1. ***geometrical elements***, such as lines, polygons, ovals, and the like (Figure 3.1a);
2. ***natural forms***, which can be further classified as
    - animal/human forms (Figure 3.1b),
    - plants (Figure 3.1c),

- physiographic features (Figure 3.1d); and

3. ***artificial objects***, such as shields, ribbons, and torches (Figure 3.1e)

Secondly, for our purposes, we will similarly divide the *applications* of ornament into four main contexts:

1. to ***bands***, which have finite thickness in one dimension and are infinitely repeating in the other;
2. to ***half-open borders***, which are tightly constrained along one or more edges, but open in other directions;
3. to ***panels***, which are arbitrary bounded regions of the plane; and
4. to the ***open plane***, in which the ornament typically becomes a repeating pattern or "wallpaper." [48]

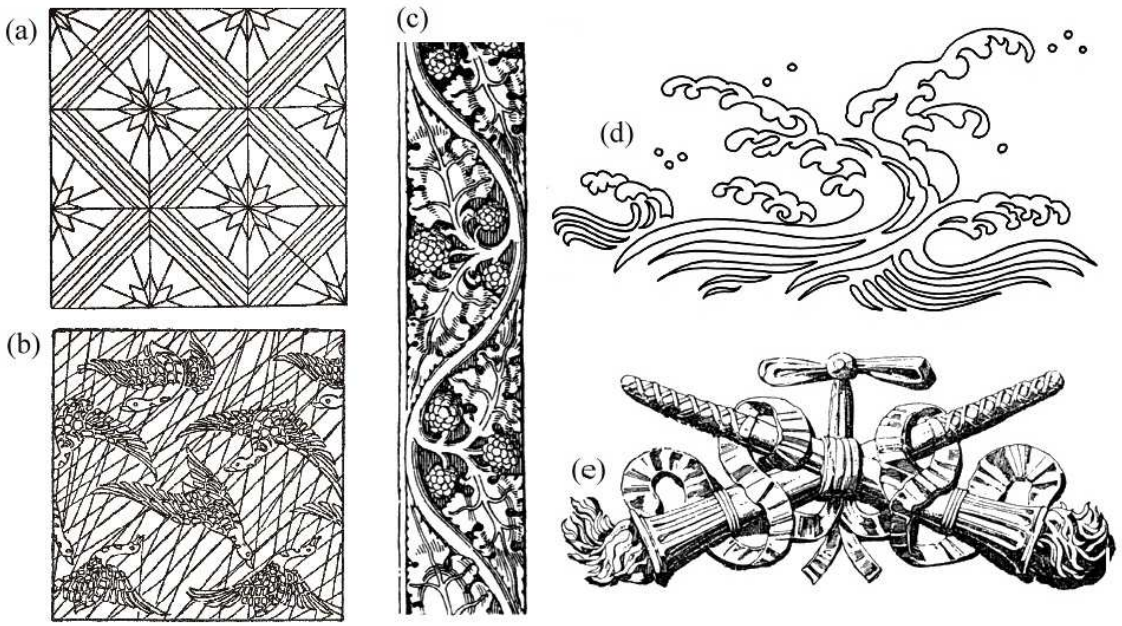


Figure 3.1: Examples of the elements of ornament [29, 48]. **(a)** Geometrical elements **(b-d)** Natural forms **(e)** Artifical objects

3.1 Principles of Ornamental Design

Wong *et al.* also explains that the one outstanding underlying principle of ornamentation is the conveyance of a sense of order or design [48, 17]. Ornamentalists use five principal techniques in conveying a perception of order: *repetition*, *balance*, *conformation to geometric constraints*, *growth*, and *conventionalization*. [11, 47, 22].

3.1.1 Repetition

Even a simple geometric mark, when repeated, can serve as the basis of an ornament. When forms are repeated, they may be repeated exactly, through translation (shifting), rotation, and may even be scaled. The possible types of repetition are produced when:

Simple translation: a form is copied and moved to a new location

Glide reflection: a form is reflected about an axis, then translated to a new location

Bilateral symmetry: a form is reflected about some axis

Radiation: a form is copied outwards from a central source

Bilateral symmetric radiation: a pattern containing rotational symmetries contains a source of radiation that is positioned off-center from the design elements it controls

Analogous: similar, rhythmic controlling lines are used to place and constrain different floral or figurative elements

Alternation: patterns are created by following successive changes from one form to another and back again, any number of times

Scaled: the same form is copied, varying only in size, usually combined with a translation and/or rotation

Organic variation: variation is introduced within a class of forms to add organic dynamism to their composition, usually through color alternation or scaled repetition

Additionally, ornament often can be found arranged in any of the seven Frieze patterns [40], giving it an underlying spatial geometric aesthetic. Overall, repetition is one of the — if not ***the*** — most fundamental ordering principle that ornamentalists adhere to when exercising their craft [48].

3.1.2 Balance

The principle of balance requires that asymmetrical visual masses be made of equal "weight." The principle of balanced masses, combined with the primal motivation for ornamentation, *horror vacui*, yields the principle of *uniform density*. This principle dictates that ornament should uniformly fill its allotted space. In some ornaments, elements with similar masses are distributed non-uniformly in space. In this case, the imbalance created with unequal positioning of forms can be offset by different elements of a smaller or large scale. This type of ordering leads to a balance within and among levels of hierarchies of visual mass [48].

3.1.3 Conformation to Geometric Constraints

A careful *fitting to boundaries* is a hallmark of ornament from many cultures [23]. In floral ornamentation, for example, the period of a meandering vine has to be adjusted not only to fit properly between the top and bottom edges of a panel, but also must provide feasible positions for secondary shoots to invade other portions of the ornamented region. Sometimes, even, the geometry of the design elements are deformed to better fill space [48].

For structural integrity purposes, for ornament that must "hang together," **tangential junction** provides a powerful sense of physical support for a production, an example of which is shown in Figure 3.2a. The placement of geometric points of

maximum and/or minimum concavity or convexity, or skeletal layouts of regions to be filled, also adds to the complexity of an ornament. Frequently, the creation of an ornament requires several recursions of subdivision and filling of space, leading to a many-tiered hierarchical composition in the final design, which is shown in Figure 3.2b [48].

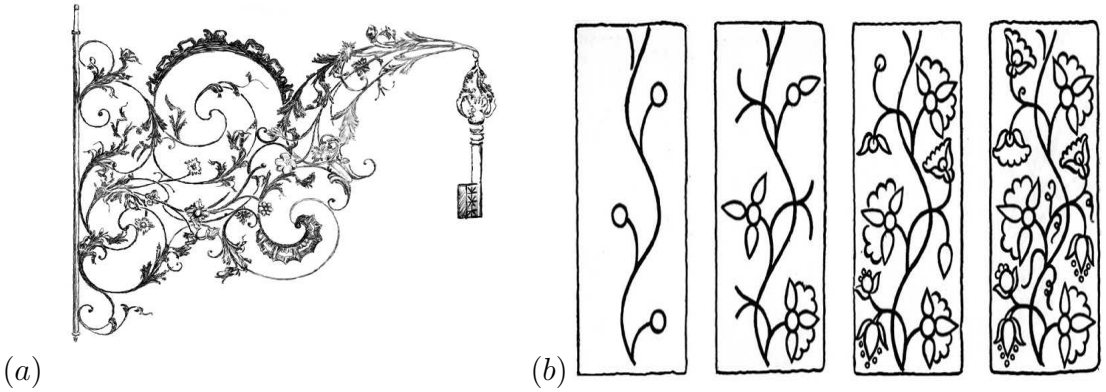


Figure 3.2: **(a)** Tangential junction gives this ornament a strong sense of physical self-support **(b)** The sequence of steps taken by a 19th-century textile designer from India in laying out a woodblock print, showing the hierarchical composition of the final ornament

3.1.4 Growth

For our purposes, we will use the definition given by Wong*et al.* for floral ornament: any ornamental design process involving plant-like growth models, such as branching structures, or plant-like elements, such as vines, leaves, or flowers [48].

Especially for floral ornament, growth is an excellent source for continuous patterns that abide by *horror vacui*, allowing for a credible means of transporting design into new regions. In this case, large spaces can be filled with the larger trunks of trees or broad leaves, and surrounding spaces can be filled with small spiral branches, additional vines and floral elements, and more. Non-rigid repetition of forms derived

from natural looking growth can also breathe life into a design [48].

Additionally, **intention** provides another avenue for artistic control. Intention is not just the process of growth in the absence of external influences, but a way of expressing growth with such influences taken into consideration. In essence, intention helps drive the overall ornament design by allowing for external influences to shape it. Examples include growth toward pre-placed flowers, guidance along a central vine, and the cooperative formation of symmetric structures, sometimes even from non-analogous locations in an overall branching structure [48].

3.1.5 Conventionalization

In ornament, conventionalization is the development of abstractions of natural form, very much unlike the more standard use of the word, which tends to imply a lack of invention. On the contrary, conventionalization in ornamentation is a highly creative process [48].

When artists develop a conventionalization, they extract only the essential aspects of form and do not allow idiosyncrasies of any specific instance of the form to persist. Instead, the conventionalized form mimics the form of reality in essence, but often is stylized and modified to be more aesthetic. According to Kaplan, since abstraction relies on a deep understanding of the object being abstracted, an automated process would seem to require real machine intelligence, which currently is not feasible [23]. As such, human beings play a critical role in the production of any sort of ornamentation, especially those that are not strictly geometric in nature.

Figure 3.3 is an example of a conventionalization, showing a side-by-side comparison of a study of the horned poppy drawn from nature, and a conventional representation based on the same study. The wave of most leaves gets amplified

and regularized in its conventionalization, while the form of the seed pods has been stylized to fill space [48].

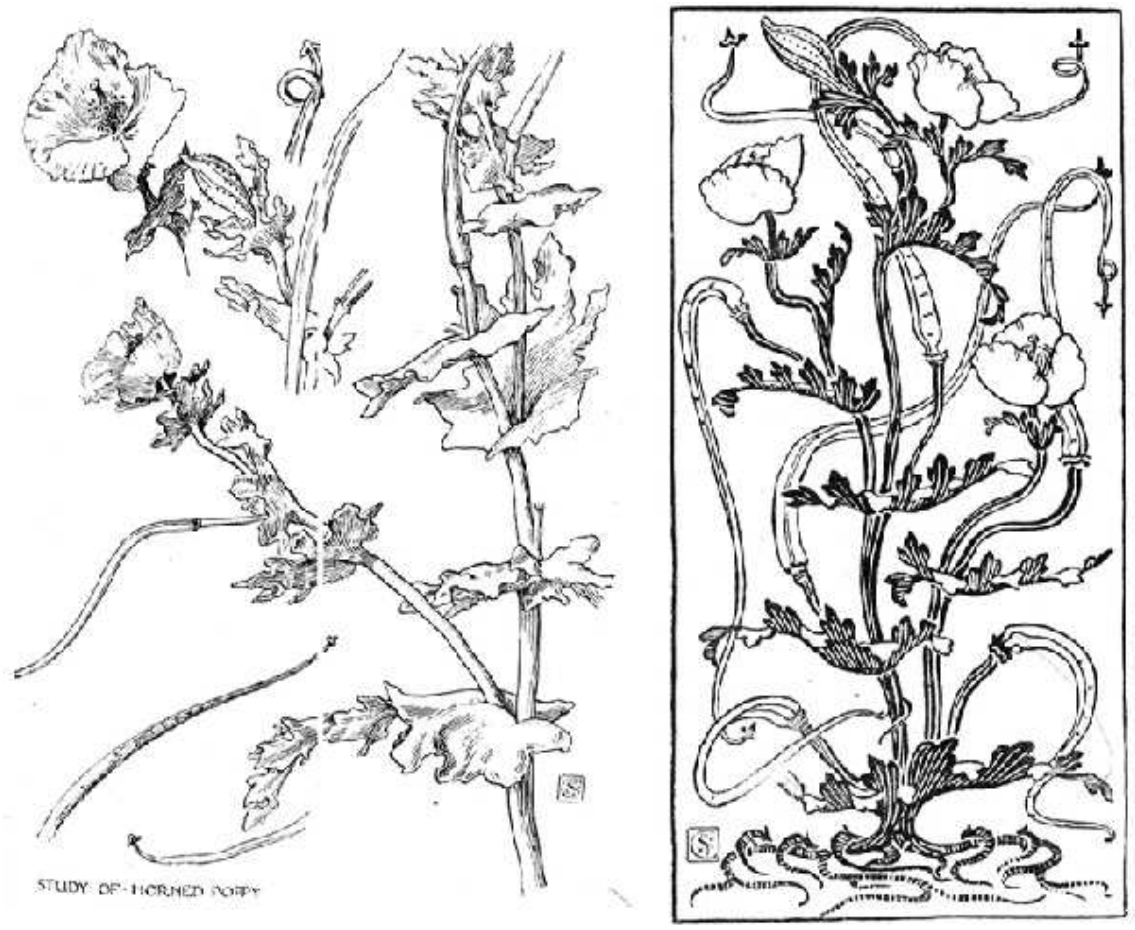


Figure 3.3: Conventionalization of the *Glaucium Flavum* taken from [48]. LEFT: The horned poppy as drawn from naturalistic observation. RIGHT: A conventionalization of the same horned poppy.

As another example, the artist M.C. Escher relies heavily on the process of conventionalization in his artwork [23]. Escher's tilings conventionalize animal forms especially, where even representations that are highly iconic and stylized still manage to be suggestive of such creatures. Escher created over sixty [7] of these tilings, and work has been done in the field of computer science toward algorithmically cre-

ating this specific type of imagery [24]. In some cases, conventionalization gives way to outright invention: shapes are decorated with suggestive eyes and appendages, but are not meant to depict any real animal [23]. Escher uses this kind of stylization to fill all possible canvas space—*amor infiniti*—while creating engaging imagery, an example of which is depicted in Figure 3.4.

Figure 3.4: M.C. Escher's "Plane Filling II" (c) 2008 The M.C. Escher Company - the Netherlands. All rights reserved. Used by permission. www.mcescher.com

3.2 System Terminology

With the conceptual principles of ornamental design just presented, there are still additional terms that are important to understand and specific to our system. The following is a list of this terminology:

buffer window: the area of our system's GUI that displays the underlying components of the ornament being created

control point: a two-dimensional point placed by the user by left-clicking with the mouse, defining a point along the main structural curve

element: a proxy that is texture mapped based on the radius-to-texture mappings provided by the user, displayed only in the interactive window

interactive window: the interactive area of our system's GUI where users can provide input to the system via the mouse

no-draw region: a region placed by the user using the right-mouse button where ornament may not exist

normal: a vector that is perpendicular to a given line or surface

proxy: a simple geometric circle or radius r, generated using the parametric equation of a circle, displayed only in the buffer window

radius-to-texture mappings: the upper and lower bounds of radius size, where radii within a given range will be mapped with a corresponding texture

seed: a two-dimensional point in space which serves as the center of a proxy

seeding: the process of placing seeds in the window, around which proxies will be generated

Chapter 4

Related Work

The computer has given artists a medium for expression which is not permanent, and never unchangeable. Computers allow for virtually unlimited artistic exploration, where artists can practice their craft without ever making unrecoverable errors, while never wasting resources [23]. While interactive tools can *help* artists create masterpieces, computer technology currently cannot carry out the creative process itself. Interactive tools give artists instant feedback on their work, while non-interactive programs can carry out complex (and sometimes lengthy) computations to produce mathematically precise compositions. Even with computers, though, the essence of creativity must come from a human being, because creativity (at least, currently) cannot be generated algorithmically.

Despite the fact that ornament has historically played a critical role in all things from architecture to art, little work has been done specifically in exploring the algorithmic generation of such adornments using computers. Even so, the following sections serve to recount the most relevant of the contributions from the literature that seem related to the two-dimensional generation of ornament.

4.1 Groundwork for Computer Generated Ornamentation (1970+)

At the 2nd annual SIGGRAPH conference in 1975, Howard Alexander presented his work in the Fortran programming language for generating the 17 symmetry patterns within a plane [2]. His contribution was followed by Grünbaum and Shephard, who used a more sophisticated computer program to generate periodic tilings and patterns [18]. Both works dealt only with generating geometric configurations on open planes, and it was not until Glassner's synthesis of frieze patterns when these patterns were created in bounded regions called *bands* [16].

The generation of flora using computers has been an area that has seen a significant amount of research and work, some of which is discussed here. Smith used parallel rewriting grammars in his work to model plant growth, called *graftals* [42]. These grammars were used to generate branching structures, which could then be visually enhanced through a post-processing step. This two step process was adopted and adapted later by Wong *et al.* for their work in *Computer-Generated Floral Ornament* [48]. Siromoney and Siromoney used modified graph grammars to generate specific kinds of patterns called *kolam* patterns [41], but their work only distantly relates to ornamentation as we have described it.

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 [5], 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 [20, 19]. Again, Wong *et al.* built upon this idea of skeletal strokes to create a mechanism for automatically arranging them within a given plane to create floral ornament [48].

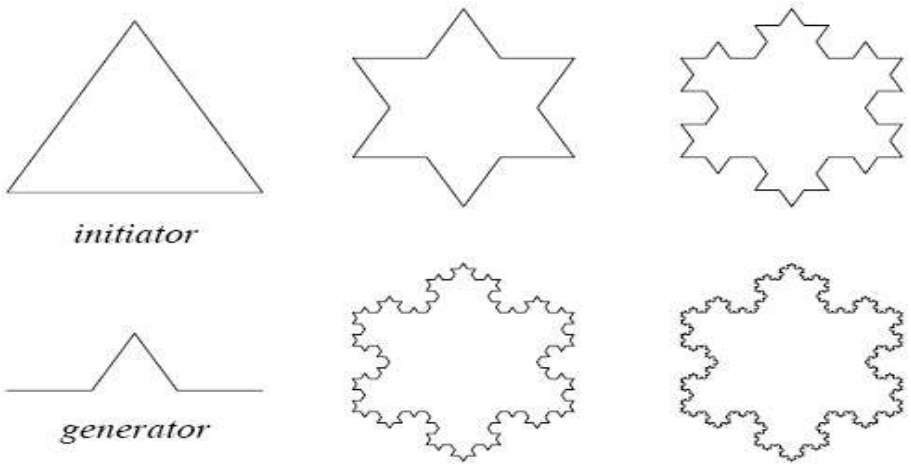


Figure 4.1: The Snowflake Curve is created using an *initiator* and a *generator*. The placement of the initiator creates the form onto which each straight-line segment is recursively replaced with a copy of the generator [38].

4.2 L-Systems, Computer-Generated Growth (1968)

Lindenmayer systems (often called *L-systems*) are formal grammars most famously used to model the growth processes of plants, but have also found uses in modeling the morphology of a variety of organisms [38]. The main power of L-systems comes from their central concept of *rewriting*. Generally speaking, the use of rewriting allows complex objects to be defined successively by replacing parts of an initial object using a set of *rewriting rules* or *productions*. The classic example of an image defined in terms of these rules is the *snowflake curve* proposed in 1905 by Niels Fabian Helge von Koch, depicted in Figure 4.1. In L-systems, such productions are applied in parallel. Parallel production application has an essential impact on the formal properties of rewriting systems, and in L-systems the parallel application of rewriting is intended to capture the process of cell division in multicellular organisms, where many divisions may occur at the same time [38].

L-Systems were developed in 1968 by the Hungarian theoretical biologist and botanist from the University of Utrecht, Aristid Lindenmayer. Originally, L-systems did not include enough detail to allow for the comprehensive modeling of complex plants, and they focused strictly on *topology* (neighborhood relationships between cells or larger plant modules). Their geometric capabilities were beyond the scope of the computational theory [35].

Within a few years, however, several geometric interpretations of L-systems were proposed with the hope of turning them into a versatile tool for modeling all sorts and complexities of flora. They have been used by a large number of researchers and experimenters for everything from creating animations of plant development [37] to the interactive arrangement of foliage models [35], to ecological simulations [36]. An important body of research has been devoted to various more complex graph-rewriting systems [38].

A significant amount of work has also been done using *L-systems* for growth to produce natural looking plants, especially by Prusinkiewicz *et al.* [38] and their Virtual Laboratory [39]. The synthetic structures based on L-systems and eventually *open L-systems* adapt natural-looking growth of flora to space, but are in no way designed to grow these plants in adherence to the principles of two-dimensional ornamental design [48].

4.3 Fractals and Dynamical Systems (∼1980)

Although capable of rendering some impressive ornamentation from the past, computers have also made possible a style of ornamentation that could not have been conceived or produced by human hands alone. Ornaments that require precise mathematical computations and the uncanny need for repetition have come to fruition

only through modern technological advances.

According to Kaplan, *fractals* are probably ***the*** form of ornamentation that is most associated with computers. Fractals often contain a staggering numbers of repetitive elements yet have little symmetry, and still manage to convey a highly structured sense of order [23]. The Mandelbrot set, pictured in Figure 4.2, serves as a foundational example of this claim. In the Mandelbrot set, reflection is about a single axis, yet self-similarity is apparent at every point and at every scale within the set. In fact, many computer scientists today continue to research interesting ways to render the Mandelbrot set and fractals like it [13]. Furthermore, Benoît Mandelbrot gives examples of the recursive branching structure of trees and flowers, analyzing their Hausdorff-Besicovitch dimension and writes inconclusively that "trees may be called fractals in part" [13].

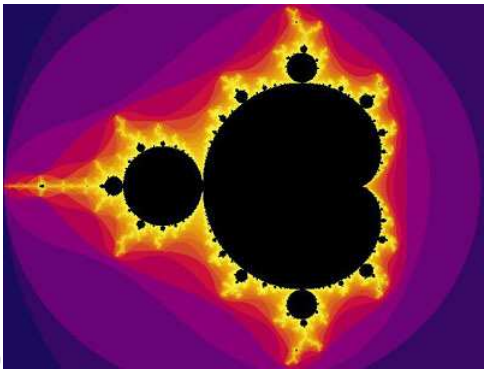
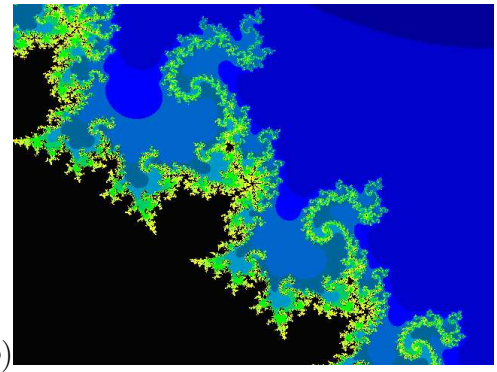
(*a*)  (*b*) 

Figure 4.2: **(a)** The full Mandelbrot set, where coloring is applied based on number of iterations needed to show divergence [10] **(b)** An enlarged region of the Mandelbrot set, taken from *http://www.jimloy.com/fractals/mandel.htm*

Chaos is closely related to fractal geometry, in spite of its ordered appearance [23]. Field and Golubitsky have created numerous ornamental designs by plotting the attractors of dynamical systems, focusing especially on those attractors that have finite or wallpaper-like symmetry [13]. Fractals such as these truly bring orna-

mentation into the digital age.

4.4 Computer Generated Floral Ornament (∼1998)

In the work by Wong *et al.*, a modern approach to generating floral ornament is presented [48], and the types of ornamentation are classified. A "field guide" [23] is also created for each type, and the final implemented system is capable of generating ornaments over finite planar regions called *panels*. The output from the system is called *"adaptive clip art,"* which not only is meant to be aesthetic, but encapsulates the rules for generating the ornamental patterns produced [48]. An example ornament generated from this system is shown in Figure 4.3.

The implementation of the algorithm that generates this floral ornament by Wong *et al.* first places the ornamental elements algorithmically using *proxies* to the actual geometry. Once a layout is finalized, the proxies are then populated with their full geometry. During this two step process, 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 [48]. The scene is seeded, and the algorithm is left to "grow" the ornament based upon a given set of rules, which are meant to capture the *essence* of the ornament [48]. Artists are responsible for creating the actual geometry for each proxy that the algorithm may use, but the final placement of ornament element proxies remains intact, regardless of the actual geometry selected. Although several computer generated floral ornaments are depicted in *Computer-Generated Floral Ornament*, how the geometry was chosen for each ornamental component is never discussed.

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 or open L-systems, which are commonly used to "grow" realistic looking plant life [38]. The rationale for this decision is based on the notion that floral *ornament* is exactly ***not*** like its real-world counterpart. Here, the growth model represents the *artist's process* in creating aesthetic stylized plant designs, and is not meat to mirror the growth of actual flora at all. While floral ornaments may involve leaves, flowers, vines, and so forth, the conventionalized versions of these elements are most often connected and arranged in ways that nature would never produce [48]. Because of this, L-systems do not provide the kind of "human touch" that is needed in the ornamentation process, and so are rejected as candidates for the algorithmic foundations in order to stay truer to the *essence* of ornament creation.

Kaplan points out that although much effort is given to description of the principles of ornamental design in the work by Wong *et a.*, the implementation of the system only loosely adheres to them [23]. Small areas, then, are appropriately dealt with using this technique, and 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 otherwise be helpful in guiding the growth of ornaments [23].

4.5 Computer Generated Celtic Design (~2003)

Geometric in nature, Celtic ornament was never intended to imitate real-life forms. Essentially, the abstract ornaments of the Celts consist of entangled threads which maintain a strict over-under alternating pattern between every thread crossing. However, the knowledge of the historical methods of designing this style of ornamentation (without computers to create them) has been lost [23].

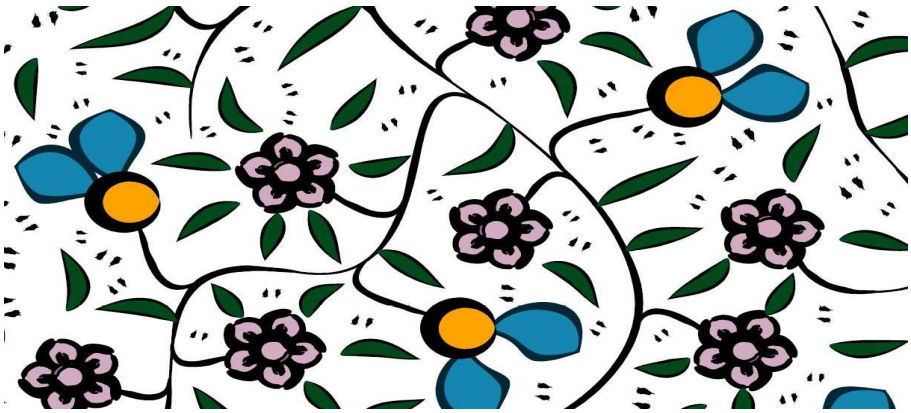


Figure 4.3: A computer-generated floral ornament, taken from [48]

Trying to unravel this mystery, George Bain presents a method of forming Celtic knotwork based on breaking crossings in plaitwork and then systematically rejoining the broken "ribbons" [4]. His work is one step beyond the early theories of John Romilly Allen, who makes suggestions on the production methods of Celtic knotwork and its eight elementary knot types [14]. Cromwell builds further on Allen's theories and, somewhat similar to Bain's work, uses an arrangement of two dual rectangular grids for ornament generation. He examines one-dimensional frieze patterns in Celtic artwork, also concluding that its structure relates to how broken crossings in the plaitwork can be arranged [9].

And, although definitely not the approach used by the original Celtic artisans, Browne uses a tile-based algorithm to fit knotwork into arbitrary outlined forms to generate his Celtic ornaments. His algorithm fills regions of a form with "tiles" that are as close as possible to squares and equilateral triangles, all bearing pre-assigned motifs which "link up" in ways that produce a seamless Celtic knotwork design [6]. When programmed to fill the forms of alphabet letters, the final ornament bears a strong resemblance to the illuminated letters of ancient Celtic manuscripts. Even so,

this method relies heavily on computer-driven computations, where even a fraction of the final design never could have been manually created by a single artisan.

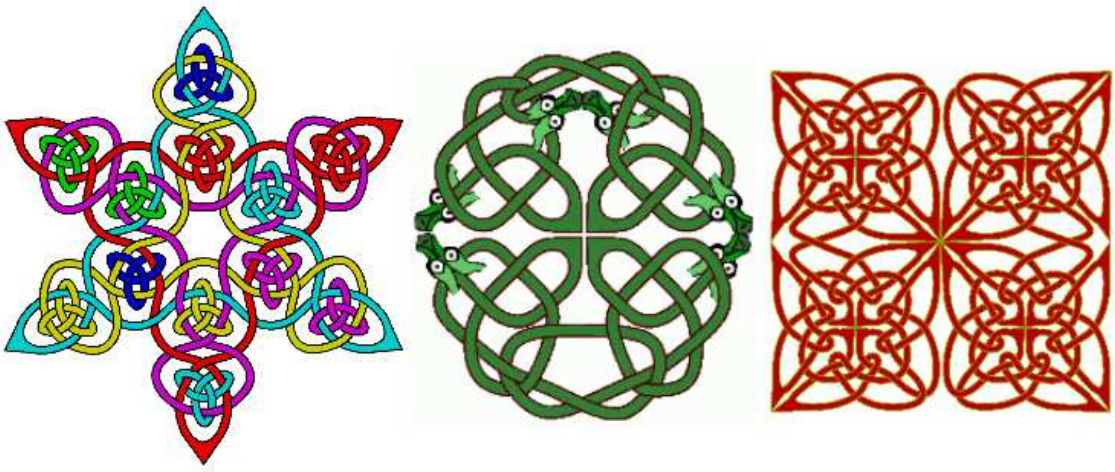


Figure 4.4: Three Celtic knot examples taken from [25]. The center image shows how the algorithm can integrate external images into the knotwork.

More recently, Kaplan and Cohen implemented a system that allows for a more complete generation of Celtic knotwork, producing output that is more "correct" than some of its predecessors [25]. Additionally, new smoothing techniques are also applied in this work to help stylize the threads to look more "natural." All possible Celtic knots are able to be created using the underlying planar graph structures of knot representation, an improvement over prior software based on grids that could only generate a subset of Celtic knots. This work also presents the first method for computers that allows for external images to be interwoven and connected into the knotwork, as displayed in the center image of Figure 4.4. Kaplan and Cohen further extend their efforts to three-dimensions by applying their algorithms to two-dimensional manifold meshes [25]. Cloth simulation and rendering seem natural extensions for future work of this system, which has yet to be explored.

More recently, Zongker implemented an interactive tool to breathe digital life

to many of these methods [49]. Other popular treatments of Celtic knots on the internet are given by Mercat [15] and Abbott [1].

4.6 Generative Parametric Design of Gothic Window Tracery (~2004)

Gothic architecture, built from 1066 to around 1530, exhibits complex geometrically shaped ornament, especially in its window tracery. Although intricate, this ornamentation generally is created by combinations of only a limited set of simpler geometry and commonplace operations such as intersection, offsetting, and extrusion [26]. As such, Gothic architecture, especially, is an open yet challenging domain for parametric modeling.

Figure 4.5: Ornamentation found in Gothic window tracery. Comparisons are made between real-world images and their computer generated counterparts.

Kaplan and Cohen use the Generative Modeling Language (GML), a "very simple" stack-based programming language, to create polygonal mesh representations of some of this Gothic window tracery. Although the underlying programming language may be described as "simple," the ornamented meshes produced are quite

complex, and are compared to their real-life counterparts in Figure 4.5 [26].

The work here focuses on a procedural approach and modularization so that, like the physical architecture itself, complex ornamentation can be created through the combination of more elementary constructions [26]. The modularization of such a system allows for the grouping of certain ornamental features to create a *style* for the window tracery, and a type of synergy is created for the overall window aesthetic.

Chapter 5

Overview and Algorithms

Our system is a tool for use in the creation of two-dimensional ornamental drawings. The user interacts with the system using both the right and left mouse buttons and a GUI to create the ornament of their choosing. 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 memory into a buffer and then proxies are seeded along it according to user-defined controls. Once seeded, textures are mapped onto the primitive proxy geometry and displayed to the user. 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.

5.1 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

In order to achieve these goals, the system was designed with the user in mind. The final application was always meant to accept external input and provide meaningful output to users of the system at all times, and was written as such.

5.2 Achieving Goals

Because **global planning** was the main methodology for creating a user-driven ornament, curve placement was essential and was the first functionality fully implemented. 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* overall aesthetic. The underlying curve representation selection process is discussed in greater detail in Section 5.2.1, where a Catmull-Rom representation was finally settled upon after experimenting with and rejecting bézier and NURBS curves. 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.*

Figure 5.1: An ornament with a structural curve composed of fifty control points. Each control point is depicted as an orange circle atop the curve. All proxies in this element are placed on the left side of the curve only.

5.2.1 Curve Algorithms and the Selection Process

The selection of curve representation was one of the first challenges that had to be overcome during system implementation. A few curve representations were considered during development:

1. Bézier and Quadratic Curves
2. NURBS Curves
3. Cubic Parabolic and Catmull-Rom Curves

Bézier Curves

User placement of control points requires that two changes of direction around a given control point are possible. As such, the entire family of quadratic curves was ruled out, including the possibility of using quadratic bézier curves. Quadratic

curves allow, at most, one directional change at any given control point. Additionally, hooking quadratic curves together is an exceptionally tricky process, because the derivatives of the curves being linked must be the same. In essence, this limits how long curve segments can be, which is a limitation we did not want to place on the user.

Upon even further consideration, bézier curves also do not provide the strict space control that other families of curves allow, and so were ruled out early. More specifically, one goal of the system was to provide the user with a *precise* means to globally plan an ornament design. Bézier curves, although defined by control points, do not move directly through these control points. Even *cubic* bézier curves (which also do not move directly through control points) do not allow for direct and accurate global planning.

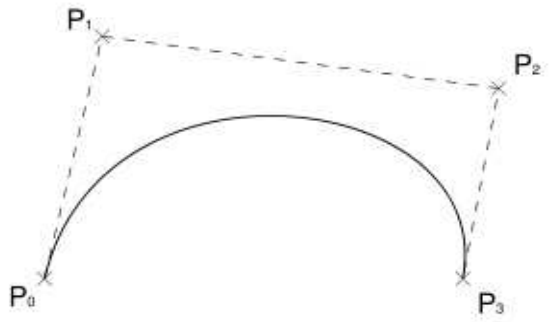


Figure 5.2: Cubic bézier curves do not move *through* their defining control points, failing to provide users with a direct and accurate global planning method.

NURBS Curves

A non-uniform rational B-spline (NURBS) is a mathematical model commonly used in computer graphics for generating and representing curves and surfaces. Especially in computer graphics, NURBS surfaces are often generated when modeling three-

dimensional geometry, and each of the generated *patches* are fitted or "stitched" together in such a way that the boundaries between patches are invisible. This process is mathematically expressed by the concept of geometric continuity.

NURBS curves were implemented upon initial development of the system, but were later found to be insufficient. Since the algorithms used for element placement (discussed in greater detail in Section 5.2.3) require that sample points be accessible, and the NURBS implementation made this challenging, we replaced the NURBS with a cubic parabolic curve interpolation strategy using Catmull-Rom curves.

Additionally, akin to the logic behind rejecting bézier curves as the main curve representation, the NURBS implementation did not provide the user with a direct and accurate means of globally planning ornaments. Even when *pinned uniform* NURBS [32] were implemented using an appropriate knot vector, where the start of the curve and the end of the curve are *at* a control point, any implementation of a NURBS curve that promised to transition smoothly from one curve segment to the next would not move *through* all other user-defined control points. Figure 5.3 depicts several pinned uniform NURBS curves of varying orders, and each is compared with the final implementation of Catmull-Rom curves currently utilized by the system.

Cubic Parabolic Curves: The Catmull-Rom Curve

With bézier curves and NURBS curves rejected, Catmull-Rom curves were then considered. Specifically, this type of curve allows 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. Catmull-Rom curves were chosen, and serve as the method of curve representation in the current system.

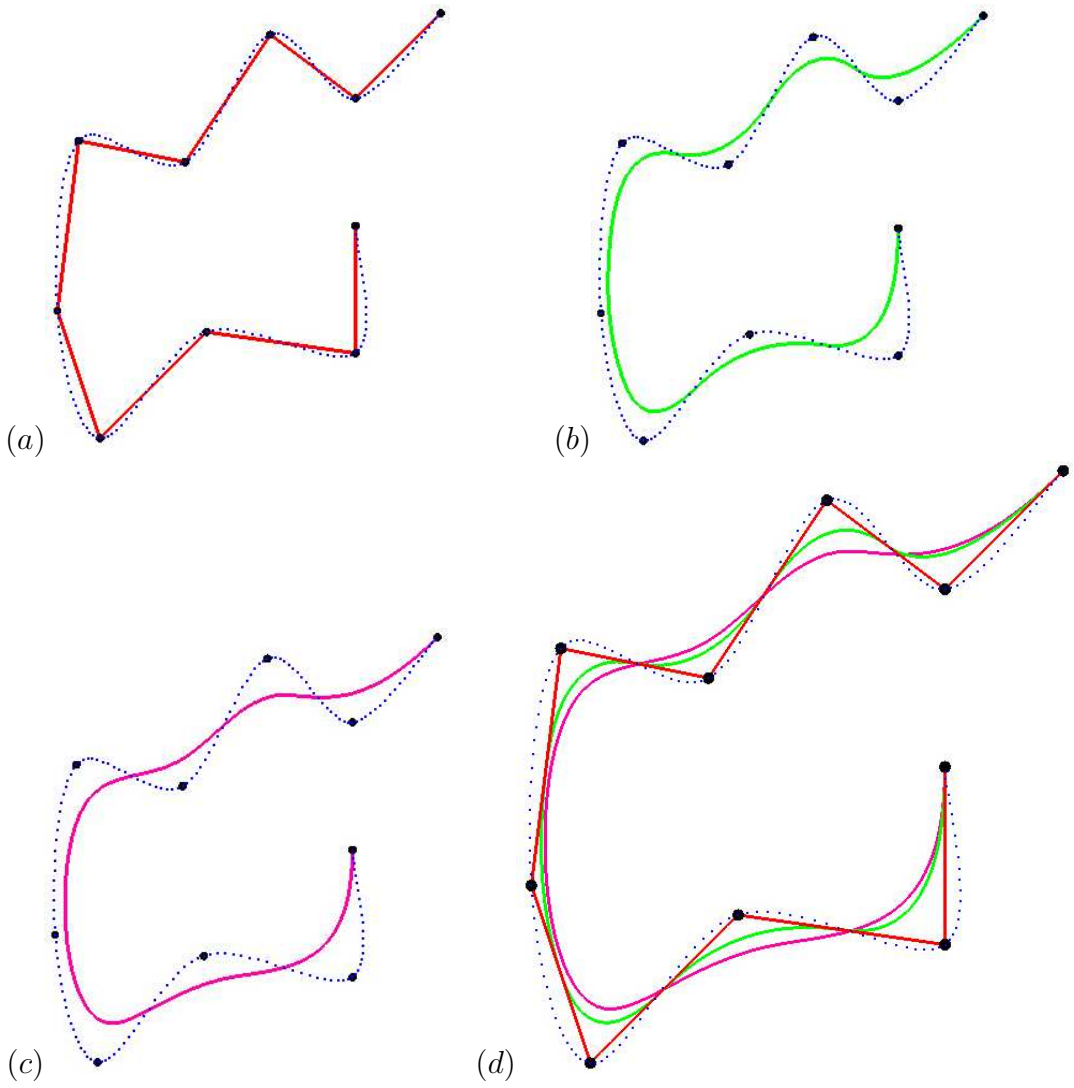


Figure 5.3: In these images, pinned uniform NURBS curves have been defined, and are depicted with the Catmull-Rom curve implementation (shown as blue points). A higher order NURBS results in a smoother curve that is farther from the control points. All curves share the same control points, but only the Catmull-Rom curve moves *through* each control point smoothly. **(a)** A very ridged GLUT NURBS curve of order 2 [linear] **(b)** A smoother GLUT NURBS curve of order 3 [quadratic] **(c)** An even smoother GLUT NURBS curve of order 4 [cubic] **(d)** The Catmull-Rom curve implementation overlaid atop all other NURBS curves of orders 2, 3, and 4.

The number of linked curves is one less than the number of control points, and each curve has its own equation. The current implementation of the system handles up to 50 control points, requiring that up to 49 equations for each dimension be solved as the curve is placed. Since our system deals with two-dimensional points in space, 98 equations must be computed and solved when all 50 control points have been placed to generate the final curve that is displayed to the user.

Catmull-Rom splines are calculated such that the tangent at each point p_i is calculated using the previous and next points on the spline, $\tau(p_{i+1} - p_{i-1})$. The geometry matrix for a single Catmull-Rom spline is as follows [44]:

$$p(s) = \left| \begin{array}{cccc} 1 & u & u^2 & u^3 \end{array} \right| \left| \begin{array}{cccc} 0 & 1 & 0 & 0 \\ -\tau & 0 & \tau & 0 \\ 2\tau & \tau - 3 & 3 - 2\tau & -\tau \\ -\tau & 2 - \tau & \tau - 2 & \tau \end{array} \right| \left| \begin{array}{c} p_{i-2} \\ p_{i-1} \\ p_i \\ p_{i+1} \end{array} \right|$$

The parameter τ is known as *tension* and it affects how sharply the curve bends at the interpolated control points within each curve segment. For our purposes, we keep the tension factors consistent at $\frac{1}{2}$. Each Catmull-Rom segment can be defined as:

$$p(s) = c_0 + c_1 u + c_2 u^2 + c_3 u^3$$

or, more compactly

$$\sum_{k=0}^{3} c_k u^k$$

where u is a parameterized value [0,1] of the control point currently being sampled, calculated by $\frac{(sample - start)}{(end - start)}$. Combined, the equation to solve for each (x,y) pair along a given Catmull-Rom curve becomes:

$$\mathbf{X} = (P0_x * (-.5u^3 + u^2 - .5u)) + (P1_x * (1.5u^3 - 2.5u^2 + 1)) +$$
$$(P2_x * (-1.5u^3 + 2u^2 + .5u)) + (P3_x * (.5u^3 - .5u^2))$$
$$\mathbf{Y} = (P0_y * (-.5u^3 + u^2 - .5u)) + (P1_y * (1.5u^3 - 2.5u^2 + 1)) +$$
$$(P2_y * (-1.5u^3 + 2u^2 + .5u)) + (P3_y * (.5u^3 - .5u^2))$$

Generally, 4th degree equations are preferred over higher degree equations for mathematical simplicity. Although we could have generated higher-order curves to generate an equation for the changing curve as the user adds control points, linking together multiple cubic Catmull-Rom curves piecewise results in the same curve smoothness, without the increasingly difficult mathematical overhead. Additionally, cubic parabolic curves provide better control over space, unlike their bézier brethren. The user is also able to modify small regions of the curve at a time, instead of having the entire curve be recalculated when any control point is moved, as would be the case with béziers. By definition, Catmull-Rom splines have first derivative continuity, local control, and interpolation, and do not lie within the convex hull of their control points [44]. The local control property of Catmull-Rom curves was ideal for providing users with a more accurate means for global planning, depicted in Figure 5.4.

5.2.2 Loading the Image Buffer

The existence of the buffer window satisfies the second goal of being able to *view the underlying structure and components of an ornament as it is created.* The buffer window, described in Section 6.3, is the area of our application 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

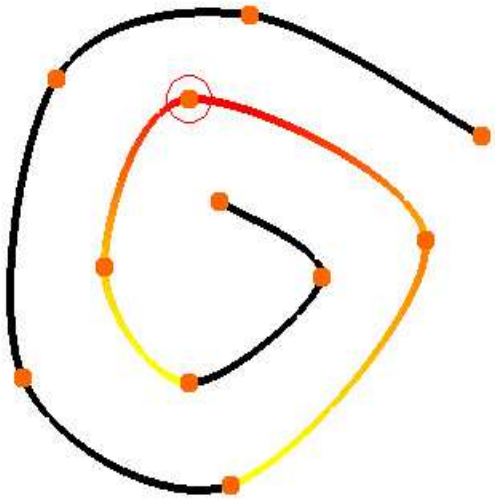


Figure 5.4: A single control point in a series of Catmull-Rom curves only **locally** affects the curve segments between its two succeeding and preceding control points in our implementation. Here, the influence of moving the selected control point is depicted with the strongest influence corresponding to red shading, and the weakest influence indicated by yellow shading.

normals in the interactive window by turning on visibility through the controls window (Section 6.1) at any time during ornament creation.

Once the user defines the control points of the curve, the curve is drawn into the interactive window. The interactive window, described in Section 6.3, 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 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, it is scanned into a two-dimensional array called the **image buffer** using the OpenGL *glReadPixels()* command. Each pixel that matches the user-defined curve color and/or

outline color is considered a curve "hit," and its value in the buffer is set to a constant value representative of curve geometry. Those pixels that are not the curve color 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 algorithms described in 5.2.3 which calculate the placement of proxies that both do not overlap the curve, and best fill up the space. The image buffer is the only means used of testing points against curve geometry. Once the image buffer is loaded with each (x,y) curve coordinate, the array is mapped appropriately to match the world space in the interactive window, and is written back out to the buffer window using a process which involves the OpenGL *glDrawPixels()* command so that the user can view the underlying components of their ornament, satisfying one of the main goals presented in Section 5.1) of the system. Once the image buffer is loaded accurately, proxy placement ("seeding") can then commence.

5.2.3 Seeding Algorithms

Selecting seed points randomly in the window—"scattering" the elements—is one method of ornament generation to fill the window, whereas the other method of ornament generation *along* the user-placed curve creates an ornament with a stronger sense of tangential junction, providing an stronger overall sense that the ornament "hangs together" better.

Seed Scattering

The user can define the maximum number of elements that will be scattered from the controls window (Section 6.1), producing an ornament similar in essence to that of Figure 5.5. Once the user has placed the desired curve, that curve is loaded into

the image buffer, and seed points are randomly generated in pixel coordinates. Each pair of (x,y) seeding values is tested for curve intersection against the (x,y) locations in the buffer. If intersection occurs, a new seed point is generated and tested for curve intersection. Each proxy has fifty chances to be re-generated into a valid (x,y) coordinate before it is skipped completely.

If curve intersection does not occur, the seed is placed into the image buffer and a new proxy is created with a radius equal to the minimum radius value the user has defined. The proxy being added is tested against other proxies already in the buffer, the curve, and all no-draw regions, and if no intersections are detected, its radius continues to grow until any intersection does occur.

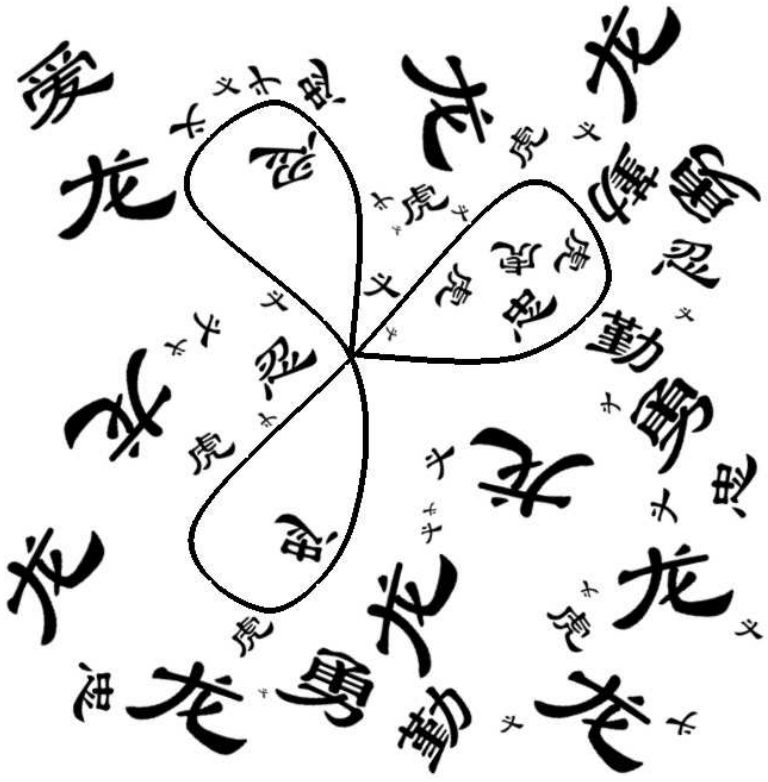


Figure 5.5: Ornament proxies are scattered within the confines of the square screen space, not intersecting any other proxy or the user-defined curve. Corresponding elements are rotated at arbitrary angles.

Although the proxy-proxy intersection test could have been (and originally was) done mathematically by testing the sum of two proxy's radii against the distance between their two centers, intersection code for both proxy-curve and proxy-proxy intersections was able to be reused for random scattering once it was written for seeding along the curve. Originally, only the proxy-curve intersection test was reliant on the buffer, and the following test (which is not reliant on the buffer) was carried out among all proxies to test for proxy-proxy intersection:

```
if (proxyA.radius + proxyB.radius > distanceBetweenAandB)
   intersect = true
```

Regardless of the intersection methodology used, once the current proxy's radius is as large as it can get, the current proxy is loaded into the buffer and its element is reflected in the interactive window, its texture randomly flipped about the X-axis and arbitrarily rotated between 0 and 360 degrees for variety. These texture transforms prevent each proxy from bearing the same statically-oriented texture, which creates a repetitive and obvious pattern throughout the ornament. In contrast, randomized flipping and rotatation of element textures gives the ornament a more organic feel. Examples of ornaments with oriented and non-oriented elements are presented in Section 7.2.

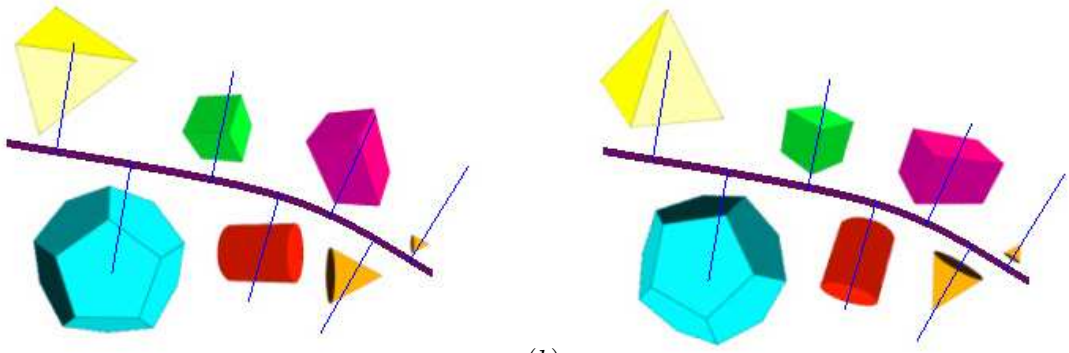
Seeding Along the Curve

The process of seeding along the curve is also reliant on the intersection tests performed with the image buffer. At the successful completion of this algorithm, proxies are places along the user-defined curve, and element textures are rotated to face the

curve, helping promote a sense of tangential junction. The algorithm executes as follows:

For each sampling point along the curve corresponding to the user-defined sampling distance, a normal is computed from the midpoint between that point and its preceding point. Determined by the group sizing controls the user has set, this calculated normal may or may not have to be inverted to point to the correct side (left or right) of the curve. A new proxy center (x,y) coordinate pair 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 accomodate this radius size change to keep the proxy close to the curve. If a proxy is not moved after its radius is decreased it shrinks around its center, sometimes being left far from the structural curve. The process of intersection testing, decreasing radius size, and moving proxies continues until no intersections occur. When this is true, 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.

After seeding is complete, the textures that are mapped onto the proxies and displayed as elements are oriented so that the bottom edge of the user-created texture is normal to the curve at its seed point. Users can view these normals by turning on their visibility in the drawing options panel in the controls window (Section 6.1). Having elements rotated to face the curve gives the overall sense that elements "grow" outwardly from the curve, and also results in giving the ornament a stronger sense of *tangential junction*, as seen in Figure 5.6. This satisfies the third



(a) (b)

Figure 5.6: **(a)** Non-oriented elements do not provide a sense that they are growing *from* the user-defined curve, and do not provide any sense of tangential junction **(b)** Elements appear to grow *from* the user-defined curve when they are oriented, helping the ornament "hang together" through tangential junction

goal of being able to create ornament where ornament elements will *"grow" from the user-defined structural curve.*

Although possible, the current implementation of the system does not combine the two processes of seeding along the curve and balancing along the curve. The process of balancing assumes that first the curve has been loaded into the image buffer, then the above algorithm is invoked for placing elements along the curve, and then balancing is performed as a separate and final step.

5.2.4 Customizable and Stylish Ornament

To satisfy the goal of allowing users to *compose a personalized ornament intuitively that adheres to the principles of ornamental design*, presets had to be coded into the system so that users could create ornament quickly without having to modify several controls during ornament creation. *Styles* provide a one-click application of a preset look and feel of an ornament. Each style is depicted in Section 7.2. Flexibility of the system, however, also required that all parameters be user-modifiable in the case

that the user wants fine-grain control over ornament creation. All of the options in the controls window (Section 6.1) are meant to help the user exert artistic control over their ornament. 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. Even without taking advantage of these controls, however, ornament can be generated quickly without ever modifying the fine-grain controls. Simply opening the application and clicking with the mouse will allow a user to create an ornament with all settings at their default values. The mouse controls, radius-to-texture mappings, and preset styles give users an intuitive method to efficiently plan out a personalized ornament in a matter of minutes. These controls also give users the ability to control the extent to which an ornament adheres to the five principles of ornament, and can be modified at the user's convenience. This satisifies the final goal of allowing users to *fine-tune a computer-generated ornament if desired, but also be able to create ornaments quickly without having to modify hundreds of controls.* Table 5.1 recounts how different controllable aspects of our system help ornament adhere to the fundamental ornamental principles: [1]

[1]Completely upholding the principle of *Conformation to Geometric Constraints* was never a goal of our system, as our focus was to grow ornament from a user-defined curve, and not necessarily just to fill up space. As such, this principle is only enforced algorithmically insofar as ornament is constrained to a square window.

Principle	Supporting System Functionality
Repetition	radius-to-texture mappings, proxy grouping sizes, no-draw regions
Balance	automatic ornament balancing, proxy grouping sizes
Growth	interactive curve placement, proxy placement and element rotation algorithms, no-draw regions
Conventionalization	radius-to-texture mappings

Table 5.1: The principles of ornamental design are upheld through system functionality

Balancing of the curve is the final step in ornament creation, if the user has decided and input that balancing is desirable system action. The following Section describes the algorithm used for balancing proxies along the curve.

5.2.5 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*. Scattered elements cannot be balanced, as they are not seeded with respect to the curve. 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.

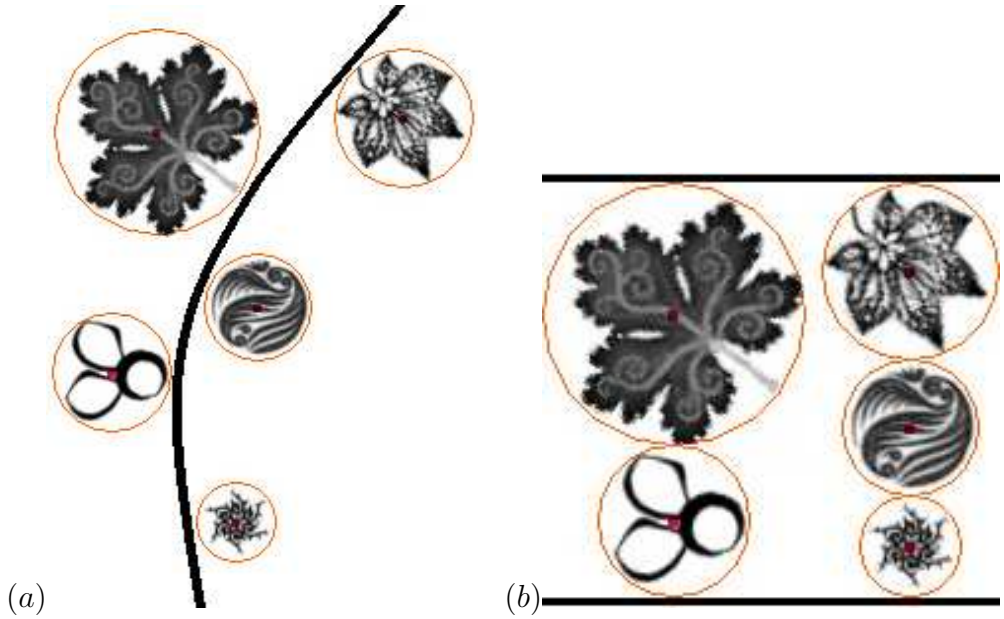


Figure 5.7: **(a)** An ornament balanced using the radius balancing algorithm, all proxies visible **(b)** A repositioning of the elements in (a) which depicts the sum of all radii on one side of the curve being equal to the sum of all radii on the other side of the curve

Balancing by Radius

The user has the ability to balance the current ornament by turning on balancing in the controls window. Balancing by radius is done by calculating the sum of all proxy radii on the left of the curve, the sum of all proxy radii on the right of the curve, and decreasing proxy radii accordingly to make the larger sum equal to the smaller sum. The sum of the radii on the right is called the *right points*, and the sum of the radii on the left is called the *left points*. As long as balancing is possible, the algorithm described in Figure 5.8 is invoked to balance the current ornament. Figure 5.7 depicts a simple ornament, balanced by radius.

A Balancing Impossiblity Scenario

Balancing may not be possible in certain circumstances where group sizes are far apart. For example, consider the following scenario where the user defines parameters as:

- The maximum radius size is set to **55**
- The minimum radius size is set to **5**
- The left grouping size is set to **1**
- The right grouping size is set to **12**
- The curve is long enough, and **13** proxies have been generated and placed

In this scenario, the point total for the left side of the curve, bearing only one proxy at a maximum radius of 55, could be 55 at maximum. The point totals for the right side of the curve, bearing 12 proxies, must always be greater than 55, when the minimum radius size is set to 5.

$$1 * 55 = \textbf{55 points} \text{ for the left side, at maximum}$$
$$5 * 12 = \textbf{60 points} \text{ for the right side, at minimum}$$

Since these two point totals can never become equal due to the minimum and maximum radii constraints, the balancing algorithm is not invoked, and a warning message stating that the curve cannot be fully balanced (but has been balanced as much as possible) is provided to the user.

Balancing by Area

The algorithm for balancing an ornament by area is very similar in nature to that of balancing by radius. Instead of summing radii, point totals are calculated by summing all *areas* within each proxy on the two sides of the curve. The algorithm

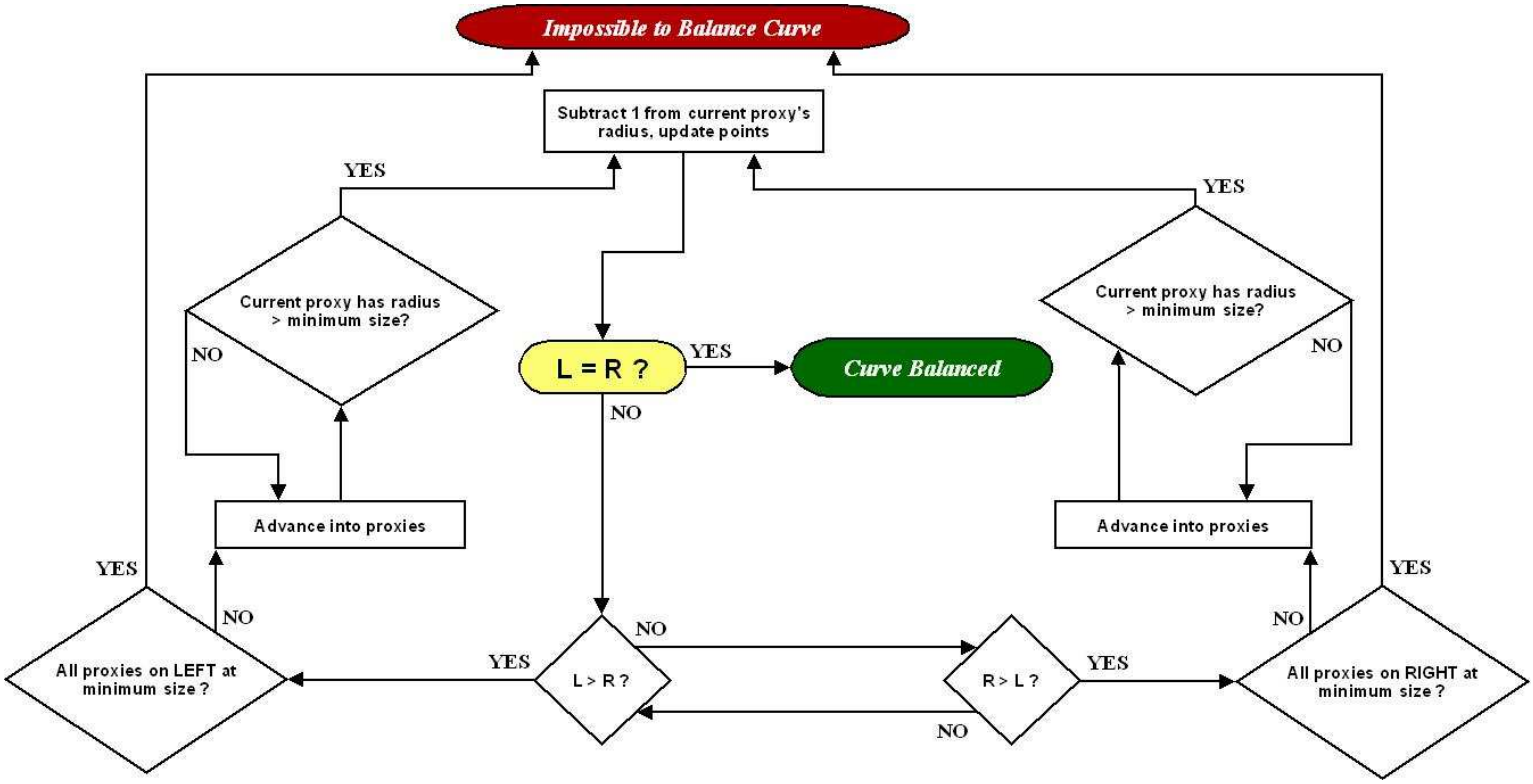
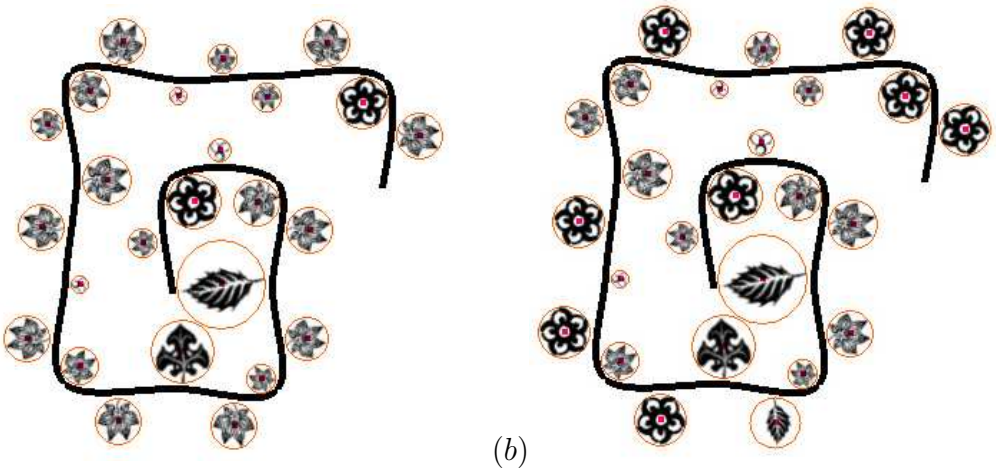


Figure 5.8: A flowchart diagram of the radius balancing algorithm

depicted in Figure 5.8 is still carried out, and radii sizes are always checked to ensure they are within user-defined ranges. Radii sizes are decreased sequentially among all proxies until the side with a larger point total becomes less than the other side. At this point, the difference between the point totals are taken to find the remaining area necessary for balancing, and the radius needed to make the totals equal is calcualted by $radius = \sqrt{\frac{area}{\pi}}$. The current proxy being examined by the algorithm is then given this radius to bring the point totals to the same sum. Additionally, balancing is not always possible when balancing by area, for reasons already discussed involving the interaction between size constraints and grouping sizes. A comparison image between balancing by radius and balacing by area is shown in figure 5.9.



(a) (b)

Figure 5.9: **(a)** An ornament balanced by radii (point totals: 182) **(b)** The same ornament, balanced by area (point totals: 10,404.9548)

Balancing by Texture Map Density

Ornaments could also be balanced by summing up the individual pixels in the textures mapped onto each proxy's element. *Texture map density* could be calculated

by a strict counting of pixels, or by assigning "weights" to different hues. A similar but more complex algorithm to that depicted in figure 5.8 would be used, using not radius size or area, but density value for point comparisons. This balancing method, not implemented in our system, would produce even more accurately balanced ornamentation.

5.3 Review

Through our efforts, we have created a system that allows for the direct, accurate, and interactive creation of two-dimensional ornamentation using **global planning**. Our system allows for organic-looking personalizable ornaments to be generated, and while the ornament structural overhead is managed algorithmically, users are given the freedom to experiement and create any ornament of their liking. Through use of the the curve and seeding algorithms, the image buffer, and customizable ornament controls, users are 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, (table 5.2.4)
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 challenges of meeting these system goals were successfully overcome in the final implementation of our system.

Chapter 6

The System Interface

The final system was implemented in C++ using OpenGL and the GLUT toolkit libraries. Although several libraries for creating a graphical user interface (GUI) were investigated and experimented with, including GLOW SDK version 1.0.4 and the Qt Windows Open Source Edition for C++ Developers version 4.4.0, GLUI version 2.35 was eventually chosen due to its complete integration with GLUT.

Our system provides an interactive method for designing two-dimensional ornament by allowing a user-defined curve to help guide the structure of ornament. Customizable texture selections and their mappings to ornament elements give ornamentalists artistic freedom when creating ornament, without the artist needing to keep track of ornament design overhead. Ornamentalists no longer need to spend hours designing an ornament before-hand, as our system allows for quick ornament creation without fear of wasting resources. Additionally, arbitrary textures can be loaded into the application, so ornament elements are not limited to narrow styles such as just floral or geometric ornament. A screenshot of the entire system is presented in Figure 6.1.

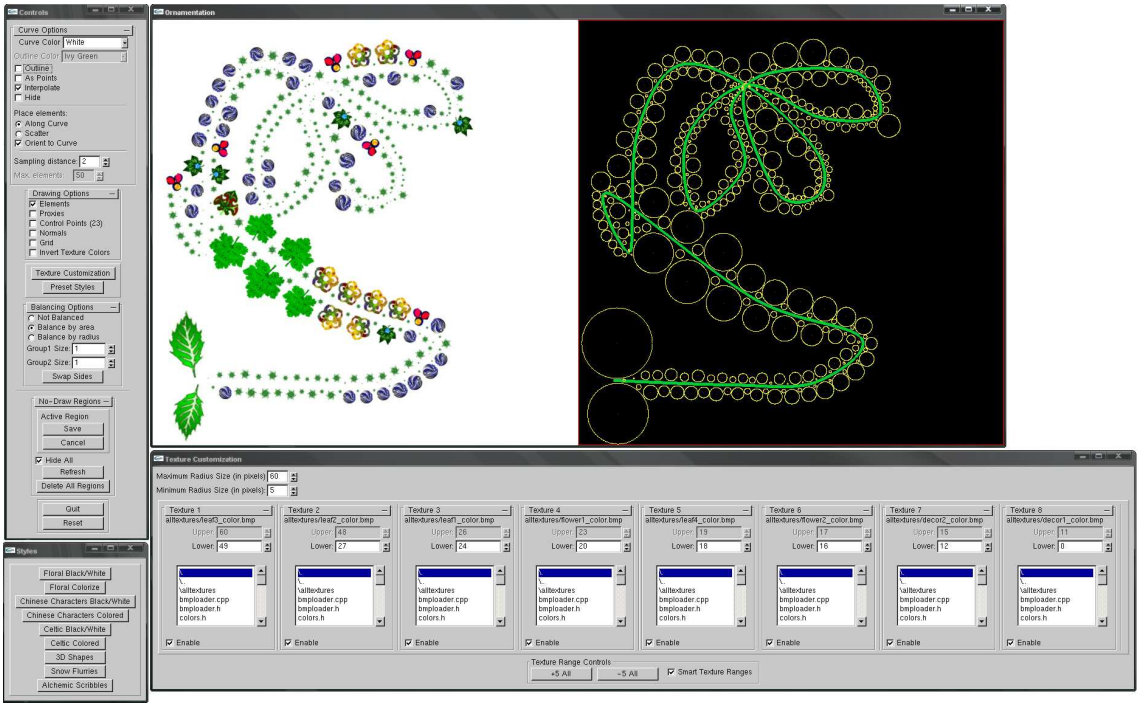


Figure 6.1: A screen capture of our entire system with all windows visible, including a sample element being created using the GUI

Our system is not intended to replace artists, but augments the ornamentation process by automatically balancing ornament, allowing user-defined proxy group sizes, and providing direct and accurate ornamental controls. Once an ornament has been created, additional control is possible using user-defined no-draw regions to even further guide the creation of an ornament.

The system is visually comprised of **five** main windows:

1. The controls window
2. The interactive window
3. The buffer window
4. The texture customization window
5. The preset styles window

6.1 The Controls Window

The **controls window** (Figure 6.2) is the area on the far left of the application which provides the user with controls to customize the ornament they are designing.

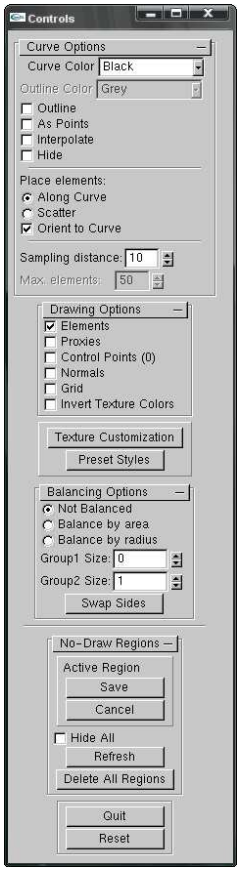


Figure 6.2: The controls window

The first category of controls are the curve options which allow users to select the colors of the curve and its outline, toggle outline visibility, show the curve as points or as a solid line, interpolate along the curve (sample more frequently when seeding along the curve), and hide the curve completely. By default, none of these options are enabled, and the curve color is black. Because the curve is a collection of connected points, when the curve is not displayed as solid, "bleeding" of the curve into proxies is sometimes possible due to curve placement, proxy placement, and/or proxy size. Interpolation along the curve is done by linearly interpolating curve points, resulting in one sampling point less than double the original number of sample points. This always results in the placement of elements closer to one another, and often has the overall effect of decreasing radius size as elements are placed closer and closer together, as can be seen in Figure 6.3. Hiding the curve turns off the curve intersection tests completely in the seeding algorithm (Section 5.2.3), and as a result, the radius of some elements tends

to increase in certain circumstances.

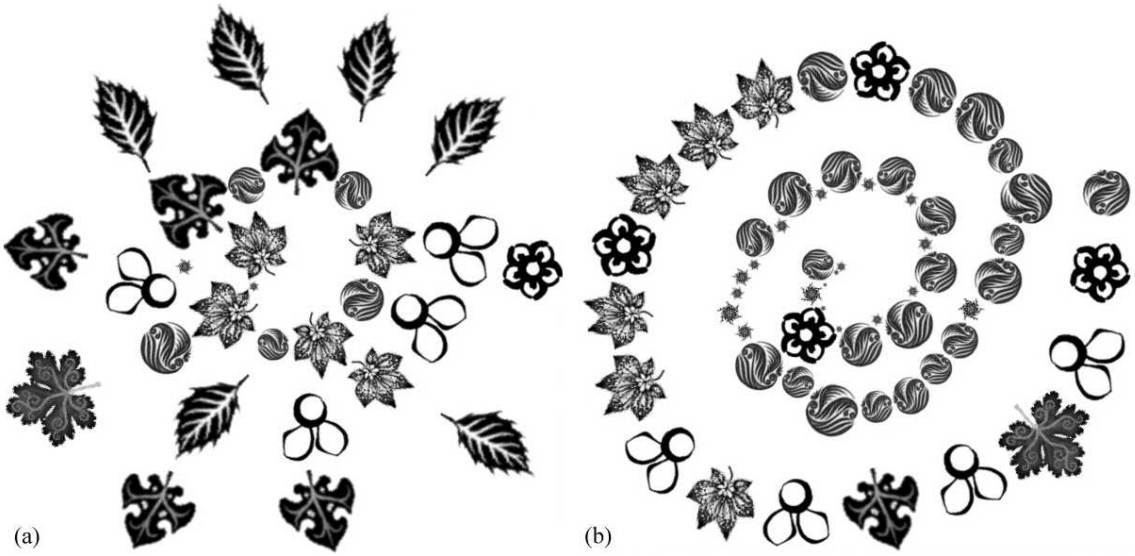


Figure 6.3: **(a)** A spiral ornament along a curve with proxies placed only on the left. The corresponding elements are depicted. **(b)** The same curve as in (a) but linearly interpolated. Linear interpolation of the curve effectively increases sampling frequency for proxy placement, and generally has the overall effect of globally decreasing radii size.

The placement of elements is also chosen in the curve options panel, where elements can be seeded along the curve, or randomly scattered within the window. If elements are placed along the curve, the sample distance at which seed points are selected can be modified, in the range of 1 to 100, inclusive, with a default sampling distance of every tenth point. Users can alter sampling distance and use curve interpolation to exert more artistic control over element placement and create stronger tangential junction, demonstrated in Figure 6.4.

If elements are scattered randomly in the window, the user can choose how many elements to scatter, also in the range of 1 to 100, inclusive. The user inputs the maximum number of elements that can appear on screen during scattering, but any

element that is randomly placed on top of or within an already existing element is not drawn, so the actual number of elements displayed may not precisely match the number the user inputs. From these controls, users can decide whether or not to orient the element textures to the curve by rotating them.

The second category of controls are the drawing options, where certain components of the ornament can be displayed or hidden. By default, only the final elements of the ornament are shown. Any components that have their corresponding check box checked are shown in the interactive window, and reflected in the buffer window appropriately. The components that can have their display toggled on and off are:

- Every element's proxy, with center point depicted
- The control points the user has placed to create the curve
- The normals from the curve along which the proxies are placed
- An overlay grid, splitting the interactive window into a 10x10 grid
- The inversion of texture colors

The third category of controls are the balancing options, where users may select the left and right grouping sizes of elements along the curve, swap sides, and choose whether to balance the curve using radii, area, or not at all. Balancing is discussed in further detail in Section 5.2.5.

The fourth category of controls deals with the no-draw regions that are able to be placed by users and are representative of window regions where ornament cannot exist. The active region, drawn in light blue as the user places its vertices with the right mouse button, can either be saved permanently for the current ornament, or can be canceled if re-placement is necessary. All no-draw regions can be deleted or hidden, and the ornament can be "Refreshed" with these regions taking effect

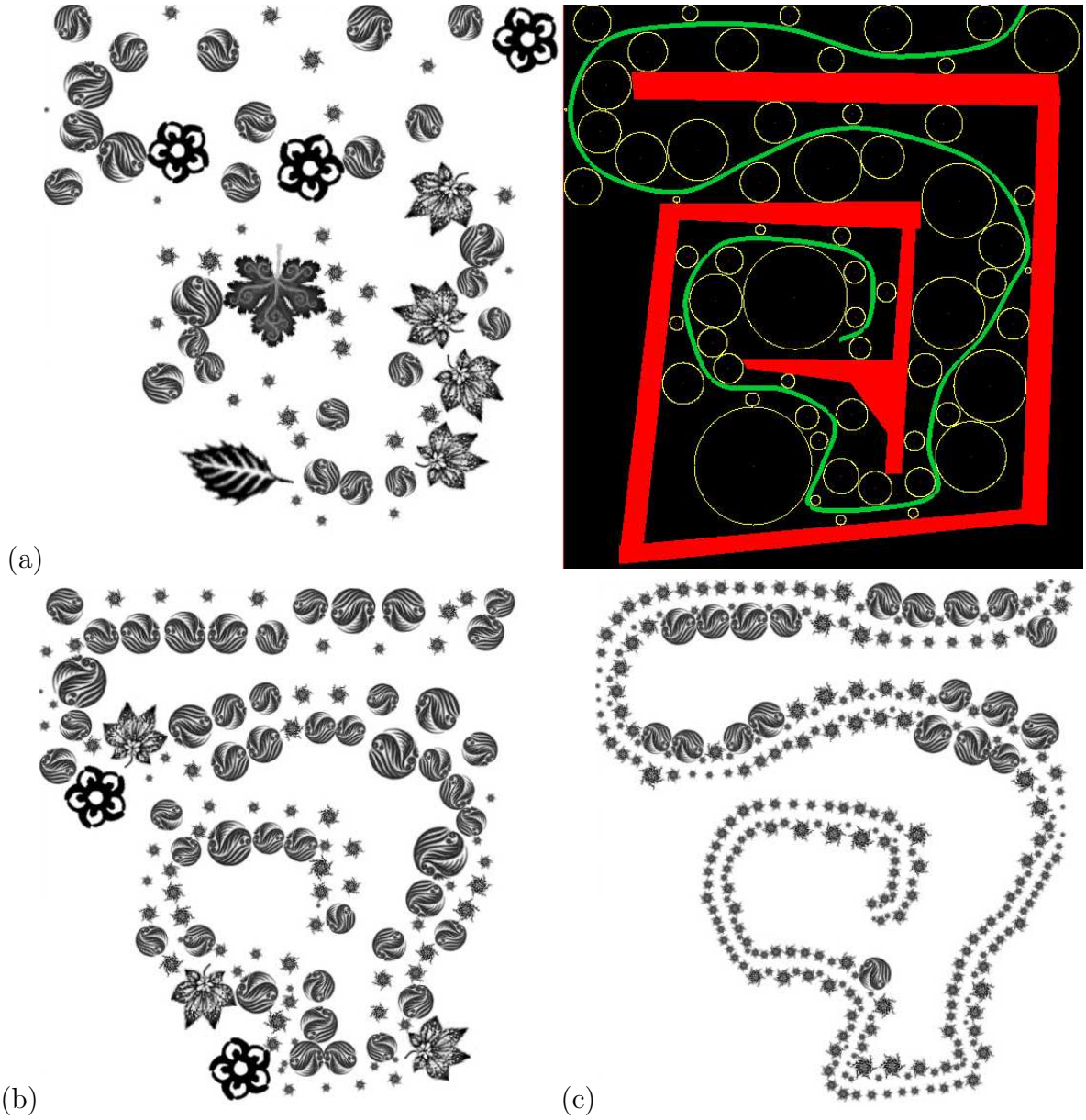


Figure 6.4: Linear interpolation and modification of the sampling distance gives users more artistic control over proxy placement and size. In this progression, proxies begin to collect around the structural curve as sampling distance is decreased. Coupled with linear interpolation along the curve, the final element has a strong sense of tangential junction. **(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.

or not. If the ornament is drawn or refreshed while no-draw regions are "Hidden," these regions do not affect the ornament.

6.2 The Texture Customization Window

The texture customization window allows users to modify the radius-to-texture mappings of each proxy. Proxies with a radius less than or equal to the "upper" range and greater than or equal to the "lower" range of a texture will be mapped in the interactive window with that texture. In this window, the maximum and minimum sizes for proxy radii and the ranges for each of the eight possible textures can be set. Each of the eight textures can be either enabled, disabled, or changed, with the requirement that any new loaded texture must exist in the *alltextures/* directory of the source code. For convenience, a +5 and -5 button is provided for large-scale modification of radius-to-texture mappings, and the ranges for textures are always made to cover the entire range of possible radius sizes by default. That is, disabling a texture does not leave those proxies blank, but instead increases the range of the next texture in line, so that all proxies are always given a texture. "Smart Texture Ranges" is on and checked by default, but by unchecking its check box, users are free to alter radius-to-texture mappings without any bounds checking. In this case, if a radius falls in more than one texture range, the texture with the lowest number is given priority, and its mapping will be applied.

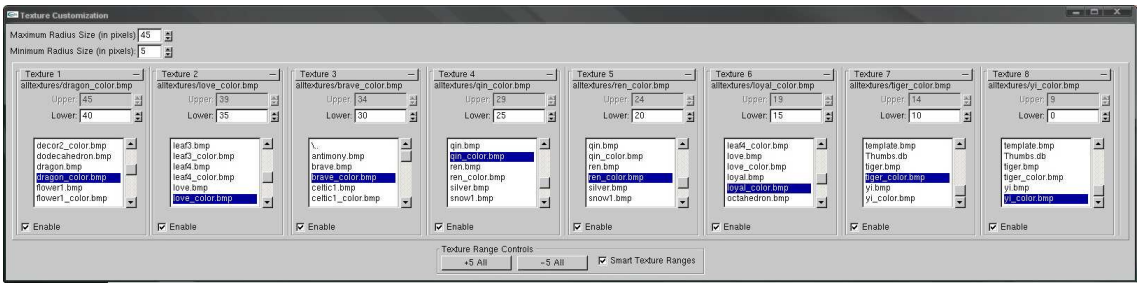


Figure 6.5: The texture customization window

6.3 The Interactive and Buffer Windows

The interactive window (Figure 6.6a) is the area where users draw their desired curve around which ornament will be placed or scattered. The curve is drawn by placing up to fifty control points in the window by clicking the left mouse button, where each control point added is connected to the last control point using the mathematics of Catmull-Rom curves.

The buffer window (Figure 6.6b) displays the underlying components of the ornament, which users may decide to show or hide in the interactive window by setting controls on the GUI. The buffer window allows the proxy-curve and proxy-proxy intersection tests to be carried out, which are discussed in further detail in Section 5.2.1. Every proxy stored in the buffer is mapped to a different texture depending on its radius, and displayed in the interactive window. The element proxies are shown with their centers in the buffer, and the window is updated every time the user interacts with the interactive window. The buffer window itself, however, is not directly interactive, and only reflects the underlying structure of the ornament being created.

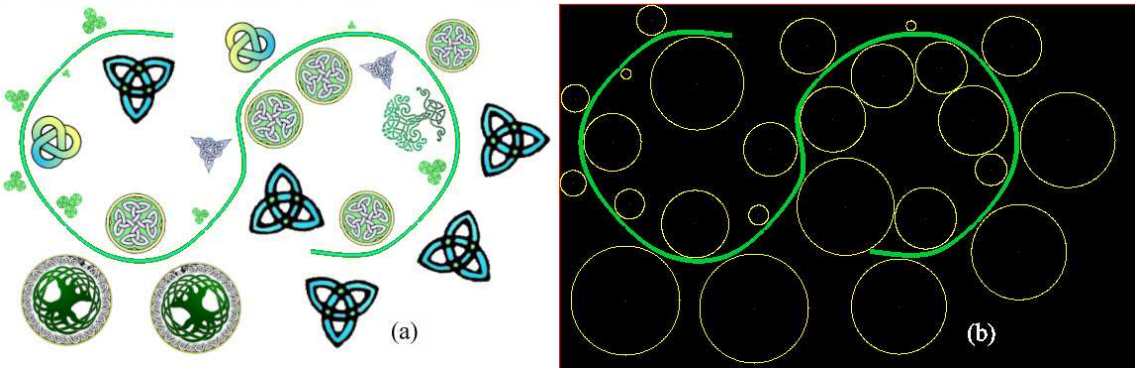


Figure 6.6: The representation of an ornament in both **(a)** the interactive window and **(b)** the buffer window

6.4 The Preset Styles Window

The styles window (Figure 6.7) contains a library of preset settings that are applied to the current ornament. Our system comes with nine built-in styles of varying motifs.

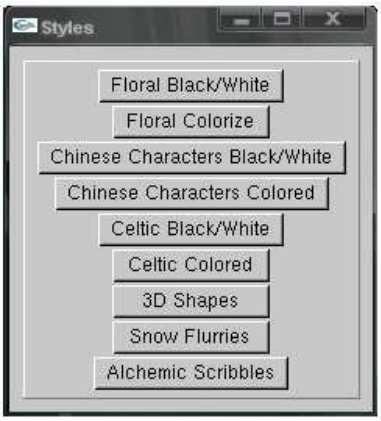


Figure 6.7: The preset styles window

Styles provide users with a one-click application of up to eight different textures, inversion of colors, main curve color, and/or curve outline color.

Chapter 7

Results and Conclusions

Despite the fact that ornament both historically and presently plays critical roles in all things from architecture to art, little work has been done exploring its algorithmic and interactive generation. In this section, our contributions to the field of two-dimensional computer-generated ornamentation are presented, and comparisons to the works discussed in Chapter 4 are made.

Using the work presented by Wong *et al.* in [48] 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 5.1 that were determined at the project's commencement, allowing users 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

Here, we explain our contributions in more depth, and compare our work with the work in [48] 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 at any time. The work done by [48] did not allow for real-time interaction with the ornamentation process. Instead, an arbitrary panel was defined and the growth algorithm was allowed to populate the panel with predefined geometry, and "seeding" of their ornament was performed off-line by a programmer through code, instead of through a GUI.

- Presents a method to generate ornament based on an underlying curve, the input of which could be transferred at a later date into physical brush strokes on electronic touch-pads/tablets, discussed in Section 7.3. Inputs in [48] 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. Both our system and [48], however, have the umbrella goal of capturing the *essence* of ornamental pattern, encoding it in a set of rules that eventually compose what Wong *et al.* terms *adaptive clip art*. Here, 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 for ornamentation.

- Allows users to automatically balance an ornament that more strongly adheres to the five principles of ornamental design. Although [48] discusses the principles of ornamental design (recounted and summarized in Chapter 3.1), their system produces ornament that only loosely follows these principles. Of the five principles presented—repetition, balance, growth, conventionalization, and conformation to geometric constraints—only the last is implemented fully. The principle of balance and conventionalization are not followed, and the principles of growth and repetition are only loosely adhered to. In contrast, our system helps users generate ornament that automatically adheres more closely to ornamental design 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. *Conventionalization* is possible depending on what textures the user wishes to map onto ornament elements. The principle of *tangential junction* is also upheld during ornament creation since all ornament proxies and elements are placed *around* and facing the curve, and we give the user a way to **globally plan** their ornament through *intention*. Completely upholding the principle of *Conformation to Geometric Constraints* was never a goal of our system, as our focus was to grow ornament from a user-defined curve, and not necessarily just to fill up space. As such, this principle is only enforced algorithmically insofar as ornament is constrained to a square window.

- Supplies pre-defined sets of textures and color mappings that define ornament "styles," an idea proposed in the work by Kaplan and Cohen in *Generative Parametric Design of Gothic Window Tracery*, discussed in Section 4.5. Although [48] 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 with no alpha channel of size 2^n 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 [48].

- Groups ornament proxies on either side of the user-defined curve to provide another level of customizability. This functionality was completely handled by the growth algorithms in [48], and was not customizable by the user.

- Introduces, for the first time, no-draw regions where ornament cannot exist. This feature is meant to eventually be developed into the handling of importing already existing images into the application, as discussed in Section 7.3.

Through our efforts, an interactive computer application that allows the user 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 each of the nine preset styles are presented in Section 7.2.

Our system also employs special user-defined no-draw regions that even further give ornamentalists control over their compositions, and coupled with the interactive placement of curve control points, fully supports a user-driven **global planning** strategy for ornamentation. With our system, users can create beautiful and personalized organic-looking ornament effectively and efficiently.

Overall, our system serves to augment the process of ornamentation by computationally managing ornament design structure while giving ornamentalists an

interactive, direct, and accurate means to experiment without fear of wasting resources. Even with improvements over previous works, however, our contributions to the field of two-dimensional computer-generated ornament within computer science has only tapped the problem space of ornamentation. The potential for our work and future related works based on our system is immeasurable. Whether ornamentation applications such as this one can help children be creative at an early age, or turn young adults onto technology as being "fun," or whether applications such as ours may save companies thousands of dollars, is still unknown. Even so, investigation into this realm of computer science that is deeply interwoven with art opens doors to new possibilities in the future where the opportunity for and adoptability of such programs we hope will grow.

7.1 Usability Feedback

We received very little, but useful, feedback from users about its intuitive design. Most users reported that the system is "fun and easy" to use, and the controls are simple enough that even younger adults (age 15) were easily able to design a personalized ornament within a few minutes. The ornamental controls of balancing and group sizing were very understandable by most users, with little to no explanation. Most users were able to intuitively deduce what each control did by experimenting with their ornament, and comparing ornaments with and without certain features enabled.

7.2 System Output: Ornamental Images

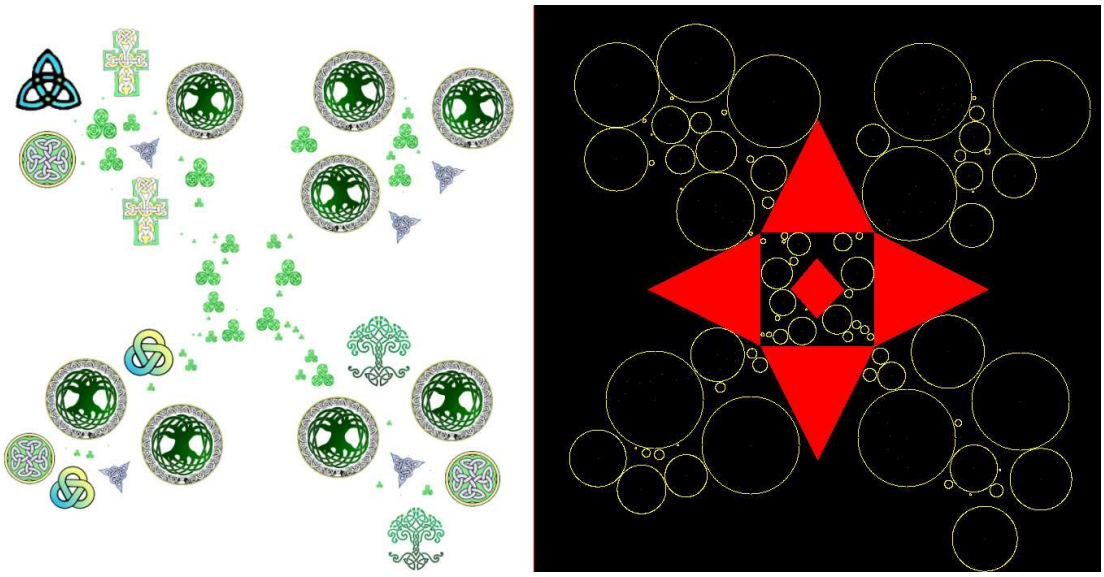
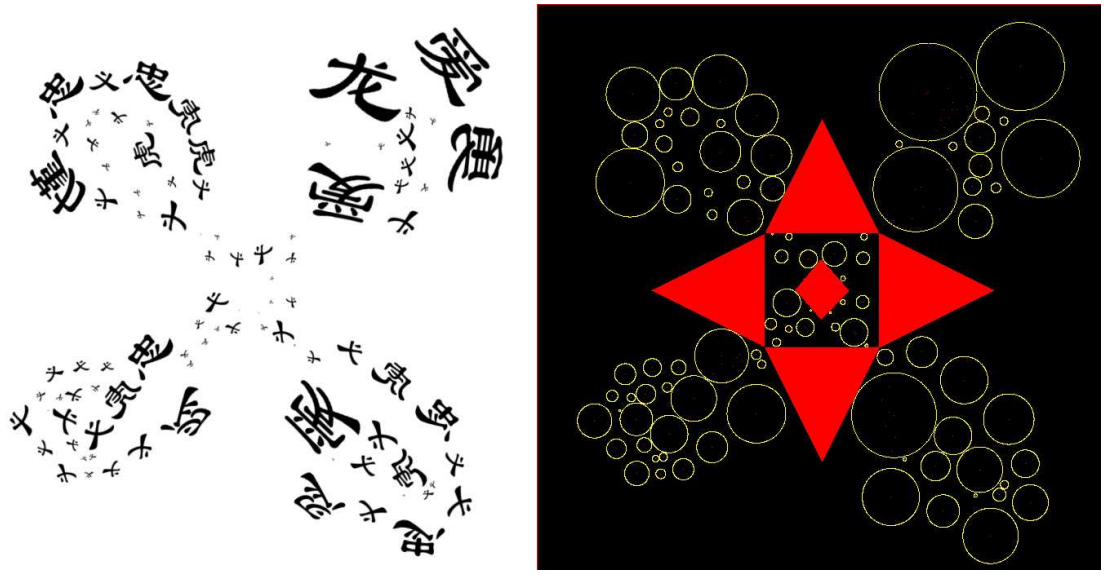
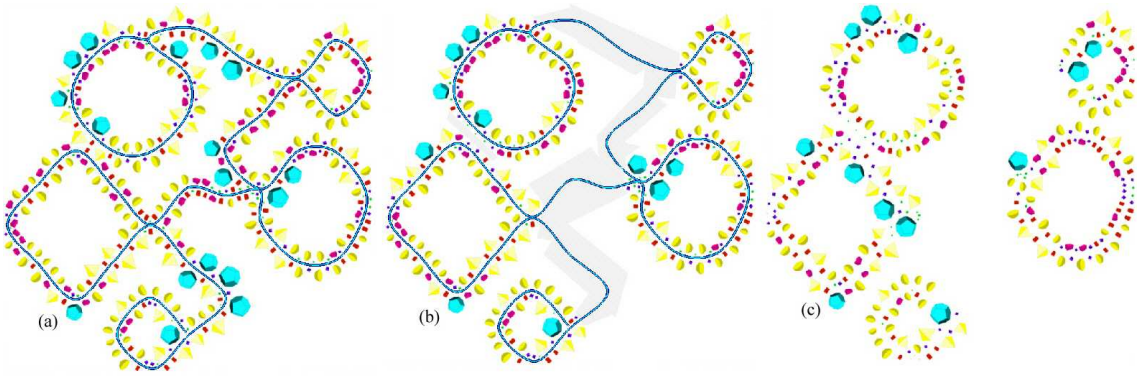


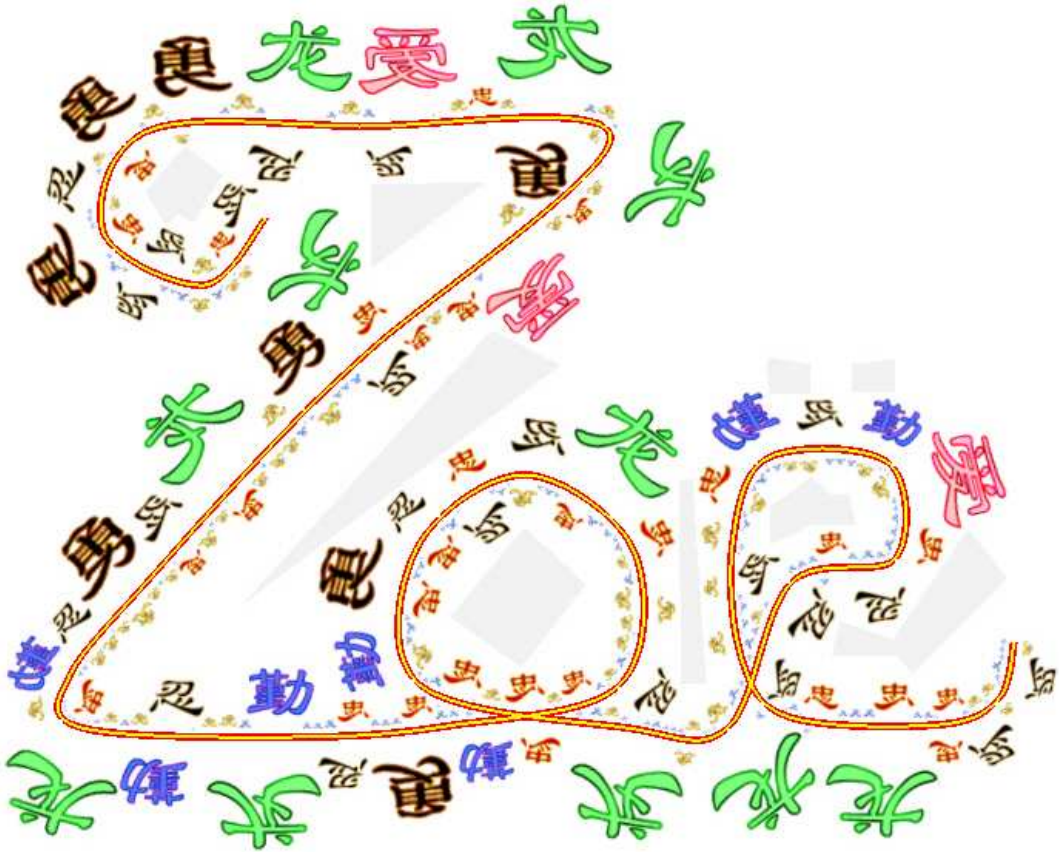
Figure 7.1: **A:** An unbalanced ornament with group sizes 2:1 with non-oriented *colored celtic* textured elements avoiding no-draw regions. The no-draw regions can be seen in the buffer window (red), and the curve is hidden.



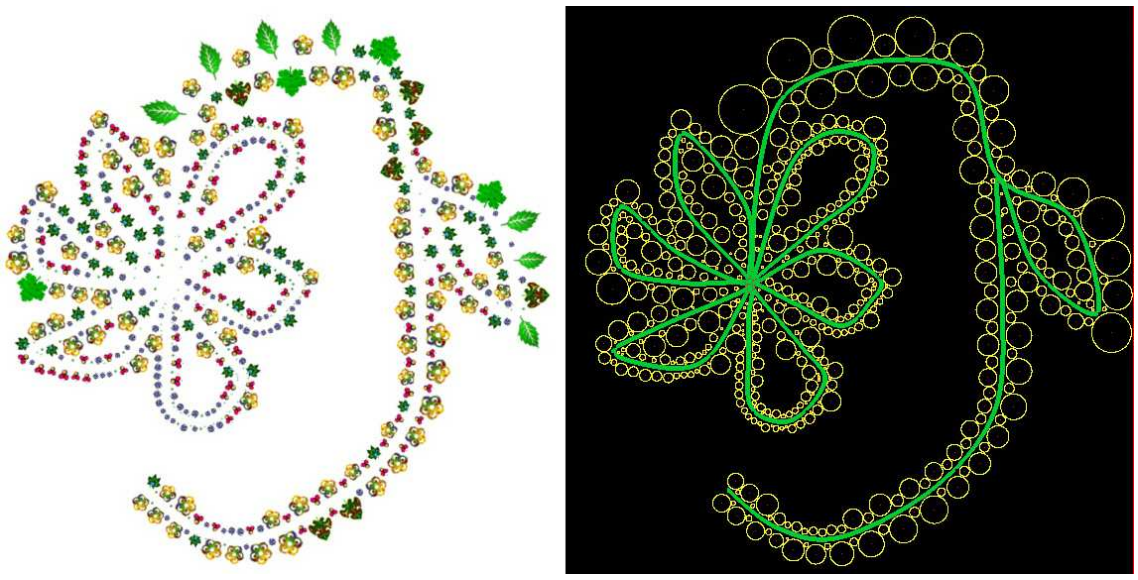
B: A radius-balanced ornament with group sizes 1:1 with oriented *black and white chinese character* texture elements avoiding no-draw regions. The no-draw regions can be seen in the buffer window (red), and the curve is hidden.



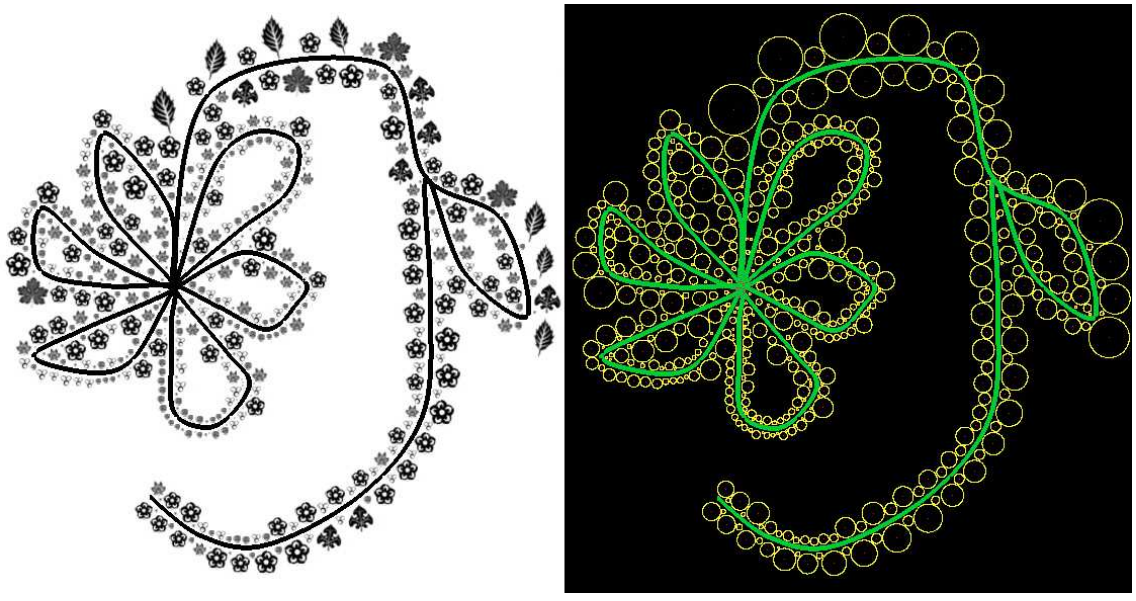
C: The creation of seemingly multiple ornamentents from a single curve using *3D Shape* texture elements with radius-balanced group sizes 1:1 **(a)** The original ornament **(b)** The ornament with no-draw regions active **(c)** The final ornament with both curve and no-draw regions hidden



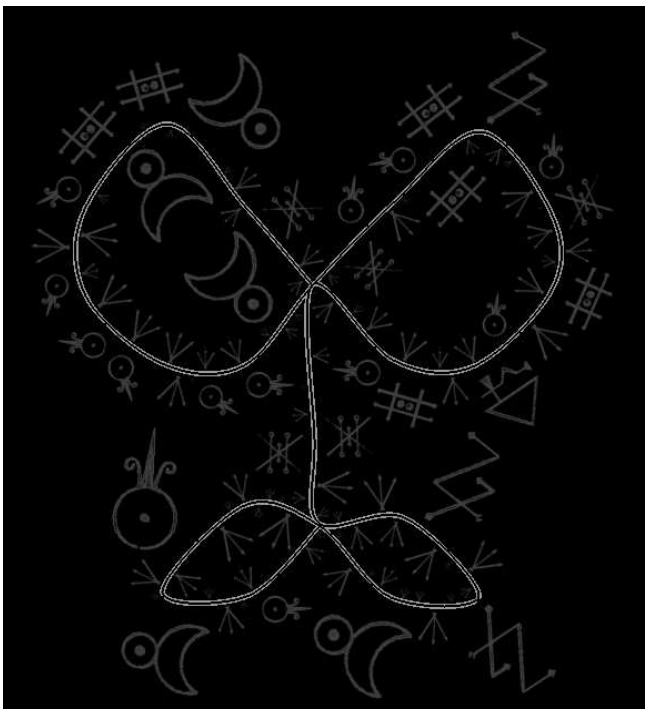
D: A rendition of the name Zoë, ornamented with oriented and unbalanced *colored chinese character* texture elements with group sizes 5:1. No-draw regions (visible) help better control proxy placement and size.



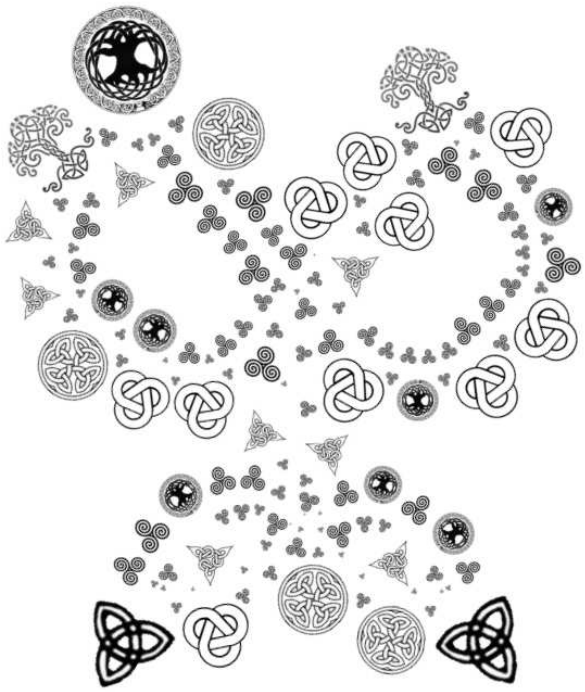
E: A radius-balanced ornament with group sizes 1:1 and oriented *colored floral* texture elements. The interpolated curve is drawn as white.



F: An unbalanced ornament with the same structure as (E) using unoriented *black and white floral* texture elements.

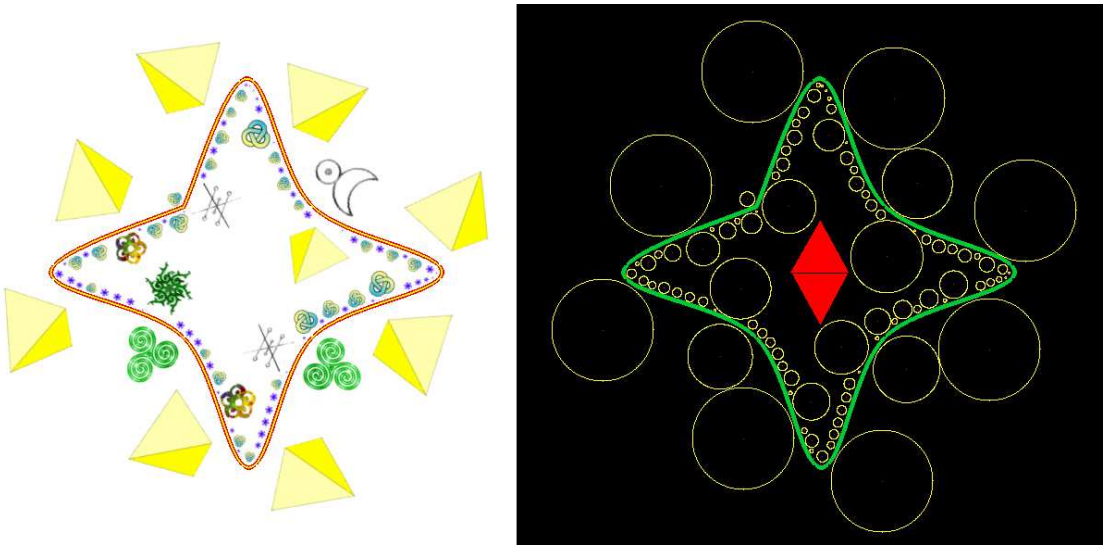


G: An unbalanced ornament with group sizes 1:2 with oriented elements textured with the *alchemic scribbles* style. The ornament colors have been inverted.



H: The same ornament as in (G) with group sizes 1:1 with oriented *black and white celtic* textured elements with the curve drawn as white.

I: The name Pam created using no-draw regions. The ornament uses *snow flurry* texture elements along a hidden curve with element groupings only on the left to decorate the text.



J: A radius-balanced ornament with 8:1 groupings using custom textures loaded from different styles. The curve is drawn in yellow with a red-orange outline. No-draw regions are hidden in the interactive window, but are visible in the buffer window.

7.3 Future Work

Since two-dimensional ornamentation can be found on everything from fliers to the human body, the potential uses of our application—and others like it hopefully to be produced in the future—are boundless. Here, we identify several areas in which our work can be expanded:

Interface and Interaction: Akin to one of the goals of [48] of generating a more "organic" looking ornament by enhancing ornamentalists' tools, a more "organic" *interface* would match the essence of the system overall. A gesture-based means for creating strokes, where each stroke would be its *own* curve about which ornament can "grow" would allow users to create "branching" ornamental structures that would better fill space, and provide additional global planning mechanisms. An expert in the field of computer graphics evaluated a very early version of our system and suggested that, ideally, a more "organic" method of curve placement could be employed, such as incorporating electronic touch-pads or art tablets that would allow users to "draw" their curves with a natural stroke-based system [27]. A more natural interface would replace the current method of placing control points using the mouse to draw curves. A live-video camera could also potentially fill this same role, where users would move a certain object under a camera which would map motion to curve placement. Ideally, the object used for curve placement would be intuitive, such as an actual paint brush or drawing implement. Fingers also might provide an intuitive interface for curve drawing and placement using video-capture. Control points in this type of system would have to be automatically generated/estimated at the completion of each stroke.

Individial Proxy/Element Alterations: Once an ornament has been algorithmically generated from user input, users could further be able to customize the ornament by individually selecting and altering any proxy and/or its corresponding element. Allowing the user to select individual proxies in the buffer window and allowing for interactive resizing would provide increased artistic precision. Allowing the user to select individual elements in the interactive window and allowing for single-element texture alteration (in addition to the radius-to-texture mappings currently implemented) would also provide more artistic freedom.

Constrain to Region: As discussed in Chapter 3, allowing the user to select the *kind* of ornament being created using a template would help promote efficiency. Users could select whether they are creating an ornament in a band, a half-open border, a panel, on the open plane, or unconstrained as free-form, instead of having to manually place no-draw regions to enforce these kinds of overall structures.

Improved Scattering: Scattering of seed points around the curve was the original method of seeding and was implemented only as a *proof-of-concept* of our system. The functionality was left in the final version of the system since it did create interesting ornaments, but improved scattering methods utilizing alternative sampling methods that seed elements more regularly but still produce organic images, would allow for ornament to better adhere the principle of *conformation to geometric constraints*.

Balancing by Texture Map Density: As mentioned in Chapter 5.2.5, balancing an ornament based upon the number of pixels which reside on either side of the curve would produce ornaments even more balanced than in the current system. Balancing by radius and area give good approximations to the actual space elements take up, but neither are as accurate as balancing by texture map density.

Genetic Algorithms for Generation: Using genetic algorithms for proxy (and possibly even curve) placement would add another organic quality to our application, but potentially may lessen the extent to which users can *directly* interact with the system. Users could define which principle(s) they most want their ornament to adhere to, and genetic algorithms could help define the proxies and/or curve to fulfill these requirements.

Area Alterations: Instead of only providing no-draw regions where ornament cannot exist, allowing regions where the *density/sparsity* of proxies can be controlled would allow element placement to be better controlled by the user. Just as several no-draw regions can be placed in the current system implementation, "density regions" could be placed in the interactive window, each with their own density controls.

Multiple Curves: In our system, all ornament must be created around a single contiguous curve. Allowing for multiple curves would gives users even more artistic freedom when creating their ornaments.

No-Draw as Imported Geometry: Our system introduces, for the first time ever, no-draw regions where ornament cannot exist. These regions are meant to mimic the feature of importing already existing images ("clip art") into the application, and generating ornament which would grow *around* them. The potential for importing images was considered during development and the code for no-draw regions was written with this functionality in mind for future implementation. Imported images need only be saved into the image buffer as NODRAW regions, and a framework for managing imported "layers" would be required.

No-Draw Workarounds: Potentially, the no-draw regions that the user can place could also affect the curve itself, and not only the proxies that grow from it. For example, if the user places one of these new workaround-regions so that it overlaps the current curve, the curve could automatically be redrawn to go *around* it. In this way, the structural curve would always be surrounded by elements, but the curve placement would automatically altered to avoid all workaround-regions. However, if the user has a global plan and a final image in mind, allowing the computer to alter the curve's path once it is placed would cause curve placement to be less accurate. Although this is an interesting area to explore in future versions of the system, a direct and accurate method of curve placement was a main goal of our system.

Styles and Themes: Although several styles of ornament are accessible in the current system, and the radius-to-texture mappings allow complete user-customizability of images to be used, adding more styles would give ornamentalists more choices without having to manually set parameters or individually load textures. The ability to *save* the current style of ornament for later use into a "library" would most likely be a useful function for users. Also, allowing an arbitrary number of textures without setting a limit of eight like the current system, would allow for more artistic creativity. Styles currently only define textures, inversion of colors, main curve color, and outline color, but this could be expanded to include other components of the ornament as well.

3D Ornamentation: The methods used in our system could be applied to three-dimensions where, for example, the seeding algorithms could be used for placement of geometry. For example, one application of this could be using our algorithms to create a "path through the woods," where three-dimensional tree geometry would be "grown" upwards from proxies, leaving a "path" through these trees that the user could navigate through. This has applications in many fields of computer science and computer graphics, including game design.

System Evaluation: Evaluation for the potential uses of this system for purposes other than its original goal of helping ornamentalists in the ornamentation process would provide further insight into other areas and disciplines that might benefit from adopting this type of system. The system would benefit from investigations into its uses in education, other areas of art, and in industry (helping to create interior design wallpapers, for example), and could be expanded in almost every direction.

Chapter 8

Prelude

The public was first exposed to the work done by Wong *et al.* cited in [48] at the 1998 SIGGRAPH conference—ten years ago. Ten years later, the work is expanded on and, we hope, revived, for new explorations. Technology changes quickly, and in just one decade several advancements over the original system have been made. However, the *spirit* of our work, at its core, dates back to the earliest days of mankind, far before cavemen were ever ornamenting their cavern walls. There will always be a need and a place for ornament in the world, and even when augmented by technology, the *essence* of the ornament dating back to 8000 B.C. should always, always be preserved.

Bibliography

[1] Abbott, S. Steve abbotts computer drawn celtic knotwork. Accessed 19 Feb 2008.

[2] Alexander, H. The computer/plotter and the 17 ornamental design types. In *SIGGRAPH '75: Proceedings of the 2nd annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1975), ACM, pp. 160–167.

[3] Archibald, C. H. *Traditional Methods of Pattern Designing*. The Clarendon Press, Oxford, 1910.

[4] Baine, G. *Celtic Art: The Methods of Construction*. Dover Publications, 1973.

[5] Beach, R., and Stone, M. Graphical style towards high quality illustrations. *SIGGRAPH Comput. Graph. 17*, 3 (1983), 127–135.

[6] Browne, C. Font decoration by automatic mesh fitting. In *EP '98/RIDT '98: Proceedings of the 7th International Conference on Electronic Publishing, Held Jointly with the 4th International Conference on Raster Imaging and Digital Typography* (London, UK, 1998), Springer-Verlag, pp. 23–43.

[7] B.V., T. M. E. C. M.c. escher: The official website. Accessed 27 Jan 2008.

[8] Cheung, A. M. Chinese antiquities. accessed 20 May 2008.

[9] Cromwell, P. R. Celtic knotwork: Mathematical art. *The Mathematical Intelligencer* (1993), 36–47.

[10] Darling, D. Mandelbrot set. Accessed 2 Feb 2008.

[11] Day, L. F. *Nature and Ornament V1: Nature the Raw Material of Design 1909*. Kessinger Publishing Company, New York, 2007.

[12] Denault, L. T. Celtic europe. Accessed 7 Feb 2008.

[13] Dispot, F. Welcome to arabeske studio! Accessed 30 March 2008.

[14] Drew. Eight basic knotwork patterns. Accessed 21 Feb 2008.

[15] entrelacs.net. Celtic knotwork: The ultimate tutorial. Accessed 17 Feb 2008.

[16] Glassner, A. Frieze groups. *IEEE Computer Graphics and Applications* (1996), 78–83.

[17] Gombrich, E. H. *The Sense of Order: A Study in the Psychology of Decorative Art*, 2nd edition ed. Phaidon Press Limited, 1998.

[18] Grünbaum, B., and Shephard, G. C. *Tilings and Patterns*. W. H. Freeman & Co., New York, NY, USA, 1986.

[19] Hsu, S. C., and Lee, I. H. H. Drawing and animation using skeletal strokes. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1994), ACM, pp. 109–118.

[20] Hsu, S. C., Lee, I. H. H., and Wiseman, N. E. Skeletal strokes. In *UIST '93: Proceedings of the 6th annual ACM symposium on User interface software and technology* (New York, NY, USA, 1993), ACM, pp. 197–206.

[21] Jensen, R., and Conway, P. *Ornamentalism: the New Decorativeness in Artchitecture and Design*. Clarkson N. Potter, Inc, 1982.

[22] Jones, O. *The Grammar of Ornament*. DK ADULT, London, 1910. http://digicoll.library.wisc.edu/cgi-bin/DLDecArts/DLDecArts-idx?id=DLDecArts.GramOrnJones.

[23] Kaplan, C. S. *Computer Graphics and Geometric Ornamental Design*. PhD thesis, University of Washington, 2002.

[24] Kaplan, C. S., and Salesin, D. H. Escherization. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2000), ACM Press/Addison-Wesley Publishing Co., pp. 499–510.

[25] Kaplan, M., and Cohen, E. Computer generated celtic design. In *EGRW '03: Proceedings of the 14th Eurographics workshop on Rendering* (Aire-la-Ville, Switzerland, Switzerland, 2003), Eurographics Association, pp. 9–19.

[26] Kaplan, M., and Cohen, E. Generative parametric design of gothic window tracery. In *SMI '04: Proceedings of the Shape Modeling International 2004* (Washington, DC, USA, 2004), IEEE Computer Society, pp. 350–353.

[27] Mainard, J. personal communication, 2008.

[28] Martindale, A. H. R. Gothic art and architecture. Accessed 22 Jan 2008.

[29] Meyer, F. S. *Handbook of Ornament*. Dover Publications, New York, 1957.

[30] Monks. The book of kells. Trinity College, multiple rebindings, http://www.tcd.ie/about/trinity/bookofkells/, 800.

[31] Morris, W. *Ideal Book: Essays and Lectures on the Arts of the Book*. University of California Press, Berkeley, 1982.

[32] Peterson, J. Albert technical memo: How to use knot vectors. Accessed 24 Jan 2008.

[33] Piacenza. A catalogue of the harleian manuscripts in the british museum. The British Library, Multiple rebindings, http://www.bl.uk/catalogues/illuminatedmanuscripts/ILLUMIN.ASP?Size=mid&IllID=11679, 1295.

[34] Pisanski, T. Computer approach to ornamental plaitwork. Accessed 30 March 2008.

[35] Power, J. L., Brush, A. J. B., Prusinkiewicz, P., and Salesin, D. H. Interactive arrangement of botanical l-system models. In *I3D '99: Proceedings of the 1999 symposium on Interactive 3D graphics* (New York, NY, USA, 1999), ACM, pp. 175–182.

[36] Prusinkiewicz, P. Simulation modeling of plants and plant ecosystems. *Commun. ACM 43*, 7 (2000), 84–93.

[37] Prusinkiewicz, P., Hammel, M. S., and Mjolsness, E. Animation of plant development. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1993), ACM, pp. 351–360.

[38] Prusinkiewicz, P., and Lindenmayer, A. *The algorithmic beauty of plants*. Springer-Verlag New York, Inc., New York, NY, USA, 1990. http://algorithmicbotany.org/papers/abop/abop.pdf.

[39] Prusinkiewicz, P. P. Algorithmic botany: Virtual laboratory. Accessed 20 Dec 2007.

[40] Reid, D. A. Frieze patterns. david.reid@acadiau.ca.

[41] Siromoney, G., and Siromoney, R. Rosenfeld's cycle grammars and kolam. In *Proceedings of the 3rd International Workshop on Graph-Grammars and Their Application to Computer Science* (London, UK, 1987), Springer-Verlag, pp. 564–579.

[42] Smith, A. R. Plants, fractals, and formal languages. In *SIGGRAPH '84: Proceedings of the 11th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1984), ACM, pp. 1–10.

[43] ThinkQuest, M. Classical art. Accessed 22 Jan 2008.

[44] Twigg, C. Catmull-rom splines. Accessed 10 May 2008.

[45] VanNocker, J. The cathedral in seville, spain. spaintravel@voyager.net.

[46] Various. Tattoo designs, the largest tattoo gallery online. Accessed 19 March 2008.

[47] Ward, J. *The Principles of Ornament*. Scribner, New York, 1896.

[48] Wong, M. T., Zongker, D. E., and Salesin, D. H. Computer-generated floral ornament. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1998), ACM, pp. 423–434.

[49] Zongker, D. Celtic knot thingy. Accessed 17 Feb 2008.