

Computer-generated Gothic Tracery with a Motif-oriented Approach

Joe Takayama*

*Kyushu Sangyo University, takayama@ip.kyusan-u.ac.jp

Abstract: Drawing decorative ornaments as computer-generated imagery (CGI) is both time- and energy-intensive because it usually involves complicated motifs. Therefore, a procedural approach that is applicable to a variety of fields may be useful for drawing ornaments efficiently. Consequently, an algorithmic approach for generating complicated ornaments has been proposed in this study. This research specifically focuses on Gothic tracery, which is widely found in European architecture from the 12th to 15th centuries. Tracery constitutes stonework elements that support glass panes in windows. Although Gothic tracery comprises a wide variety of complicated patterns, it usually consists of combinations of circles with a few straight lines. These combinations and geometric operations can generate various motifs. However, it is difficult to extract the rules for obtaining desirable patterns from circle arrangements because of variations in geometric situations. Therefore, we attempted through this study to draw Gothic tracery using a motif-oriented approach whereby an algorithm automatically seeks an appropriate position in a closed 2D space for placement of the motif. Thus, the algorithm determines the arrangement of circles required to draw the motif. This algorithm can produce some principal Gothic-style motifs automatically, and recursive execution of the algorithm can generate complicated patterns.

Keywords: *Computer-generated imagery, Ornaments, Gothic tracery, Procedural approach*

1. Introduction

A procedural approach in CGI has been used mainly to represent natural objects and natural phenomena that are too complicated to be drawn or animated by artists [5]. However, applications of the approach to artificial entities such as cities and buildings have increased gradually in recent years [12, 13]. As CGI becomes increasingly indispensable for feature films and video games, it is said that various artificial entities must be generated procedurally. For example, Whitehead [14] notes that there is room for further research on a procedural approach for generating decorative ornaments to save production costs, especially for numerous video games set in the medieval period.

Therefore, this research deals with the procedural generation of decorative ornaments, especially Gothic ornaments. Unlike fine arts, which are based on the artist's sensitivity, most decorative arts are based on highly standardized patterns. They usually have distinct formative rules, and they have been developed and sophisticated according to these rules. Decorative arts can be widely classified from primitive geometric patterns, to exquisite ornaments that consist of realistic motifs. Because formative rules regarding ornaments (e.g., repetition and symmetry) are clear, we expected to extract the rules applicable to some decorative styles and represent them in a computer algorithm. Clear geometric rules apply to Gothic tracery, a traditional form of decoration and the focus of this research. Through this research, therefore, we have attempted to generate complicated Gothic tracteries using a simple algorithm.

2. Related Works

Although few research cases deal with a procedural approach for designing decorative ornaments, some previous works pertain to the early years of CGI. For instance, Alexander [1] described a FORTRAN program for generating the 17 plane symmetry patterns. However, most research studies from that time focused on simple geometric patterns or tiling patterns. As applications of CGI expanded in the 1980s and 1990s, advanced representations of ornaments emerged gradually. A popular topic in this field is the guilloche, an ornate motif consisting of woven ribbon. Guilloches are widely observed in various traditional Western designs such as Celtic, medieval Russian and Armenian ornaments. M. Kaplan et al. [10] attempted to construct an algorithm for generating woven ornaments called “Celtic knots” in an enclosed 2D space. Islamic patterns are relatively popular in procedural generation because a clear geometric rule is applicable in their production. C. S. Kaplan et al. [9] attempted to draw the Islamic star pattern using an exquisite tiling configuration. Another study focused on floral patterns, which are present in medieval Western illuminated manuscripts. Most floral patterns in illuminated manuscripts are filled with repeated motifs based on a complicated iteration rule. Wong et al. [15] proposed a drawing method called “adaptive clip art” to fill an enclosed 2D space with floral ornaments. By changing the rule of stem branching and the type of floral motif, this method can be employed to represent various floral patterns in enclosed spaces.

The abovementioned works, however, have focused exclusively on planar decorative patterns, and very few attempts have been made at procedural generation of 3D or semi-3D ornaments, such as reliefs, wall plaques, moldings, and grilles. With regard to Gothic tracery, Havemann et al. [8] attempted to generate Gothic tracery using Generative Modeling Language (GML). However, their approach covers a limited number of simple motifs, such as arches and a rosette, and it is impossible to generate complicated tracery patterns from the late Gothic period. Therefore, this research attempts to represent complicated Gothic tracery efficiently by combining various motifs.

3. Principles of Gothic Tracery

Tracery describes the stonework elements that support glass in a window; it is found mainly in Gothic architecture. Gothic tracery is characterized by its geometrical simplicity based on combinations of circles with a few straight lines. Artful arrangements of the circles and lines enable various motif designs, and a combination of the motifs can generate various patterns. Generally, motifs are arranged within a window frame with a spired arch, and the arch itself also consists of a combination of circles and lines.

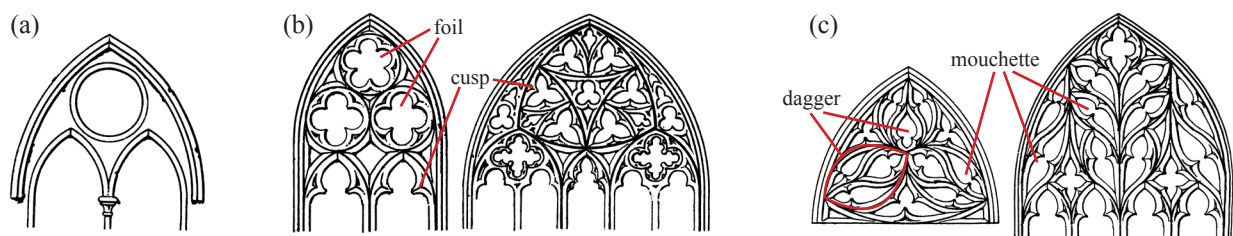


Figure 1 A transition of Gothic style [6 *public domain]

The Gothic period spanned the 12th to 15th centuries, and the style of decoration changed considerably during that era [3]. In early Gothic style, tracery consisted of simple arches alone, or arches with a few circles at the top, as shown in Figure 1a. In this type of tracery, called plate tracery [2], frames surrounded the arches and circles were thick and flat, and molding was rarely used. Gradually, frames became more slender, and this type of tracery

was called bar tracery [3]. In addition, new design elements were introduced (e.g., a cusp or inward-pointing spur, and a foil or lobe motif separated by cusps, as shown in Fig. 1b). These elements also consisted of combinations of circles, and the elaborateness of arrangements intensified as time progressed. In late Gothic style (Fig. 1c), motifs and their arrangements were so highly sophisticated that it was almost impossible to recognize with a mere glance how the circles were combined. Specifically, motifs called daggers and mouchettes were frequently used; these motifs became tangled with each other. However, each motif consists of circles and a few straight lines.

Traceries can be found not only in windows but also in various objects such as furniture, altars, and tomb decorations. The largest application of tracery would probably be a rose window in a cathedral. The most important role of window traceries, including those in rose windows, is for supporting glass. In religious architecture (e.g., cathedrals and cloisters), stained glass windows were commonly used to impart the Christian dogma via pictures. Thus, a tracery is designed occasionally based on the stained-glass content [4]. In general, glass parts of the window tracery are recognized as figures when viewed indoors, and frames are considered as backgrounds (Fig. 2a). This figure-ground relationship inverts outdoors, and frames are recognized as figures when tracery is viewed from outside (Fig. 2b). In this research, frames are regarded as the main parts of traceries; thus, the design of frames is a focus of this study.



Figure 2 Examples of a figure-ground relationship in window tracery (Photos by author), (a) Philadelphia Museum of Art , (b) Cathédrale Notre Dame de Paris

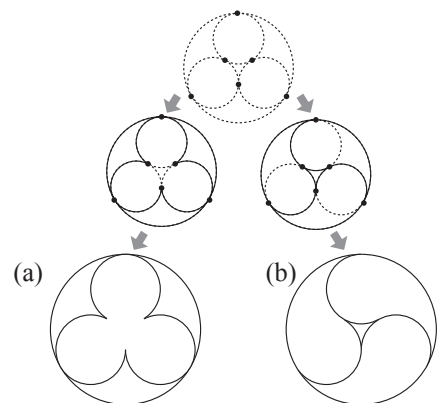


Figure 3 Different appearances from the same circle arrangement

4. Our Approach

A motif-oriented algorithm was used in this research to draw Gothic tracery efficiently. Although traceries consist of combinations of circles, their resultant appearances vary depending on which parts of the circles are extracted as arcs. For example, the trefoil, which includes three foils, can be drawn by extracting arcs as shown in Figure 3a. On the other hand, Figure 3b displays a different configuration; nevertheless, the circle arrangement is the same as that in Figure 3a. Thus, motifs can be drawn by extracting a certain part of an arc from each circle, and an appropriate circle arrangement is very important for the production of a desirable motif. In traditional tracery, the circle arrangement is conducted initially according to a combination of geometric rules and Christian creeds [11]; then, a designer or an architect extracts motifs from preliminarily arranged circles. In this approach, circles are used as clues for discovering a motif from almost infinite choices. However, it is difficult to anticipate an appropriate circle arrangement without human intuition because variations depend on the geometry and desired style. Further, it is difficult to develop an algorithm that results in the selection of appropriate arcs from numerous circles. To discover a correct arc from the desired motif, not only geometric- and analytical-approaches, but also a computer-vision-approach would be necessary.

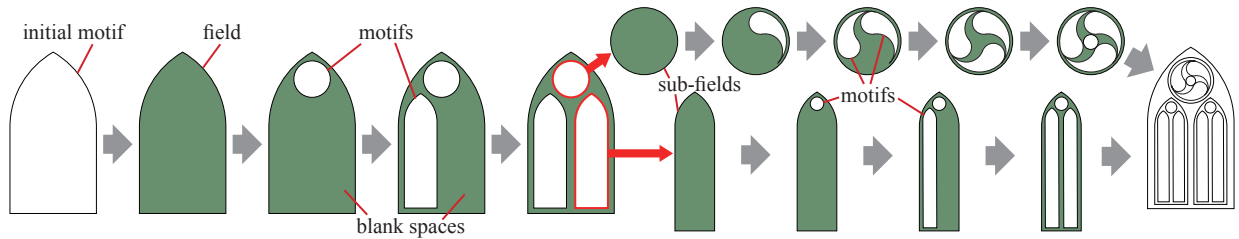


Figure 4 An overview of motif arrangements

Therefore, a motif-oriented approach was adopted for this research. Unlike the process of finding arcs from preliminarily arranged circles, this approach begins by setting properties of the motif (e.g. size, position, and angle) in an enclosed 2D space. Then, the algorithm calculates and arranges the necessary circles and lines automatically in accordance with the properties and type of motif. As illustrated in Figure 4, the algorithm sets another motif in the blank space that is formed when the current field space is separated from a motif. This process is iterated until the blank space becomes smaller than a certain size. Additionally, the arranged motif itself can become a new field space in which small motifs can be arranged. By executing these processes recursively, complicated tracery can be generated. Details of the algorithm are discussed in the subsequent subsections.

Regarding a rendering process, a resultant image can be rendered from 2D data alone. Instead of rendering from a 3D model, a semi-3D image (just like a relief) was generated in this research by detecting concavity and convexity from 2D tracery data. Details of the rendering process are explained in the section 5.

4-1. Drawing Motifs

Some principal motifs characterize Gothic tracery. In this research, an algorithm positions a motif automatically; then, necessary circles and lines are set. The drawing processes for some basic motifs are described in this subsection.

Arch

The foundational motif in Gothic ornamentation is a pointed arch. In this research, an arch was used as an initial field space, and tracery was arranged within the initial arch. A lancet arch (Fig. 5) is the most popular one in Gothic style. A lancet arch can be drawn by setting circles with a certain radius on each side of an arch at first. Then, the point where two circles intersect is determined as the top of the arch (Fig. 5a). Changing the radius of each circle allows adjustments in the appearance of the resultant arch (Fig. 5b). A coordinate value for the top intersection point can be calculated by solving a simultaneous equation of two circles. Among two possible solutions, the one with the higher Y value is adopted as the top point of the arch. An ogee arch, as shown in Figure 6, is another popular arch in Gothic style, especially the late French Gothic style called Flamboyant. It can be drawn by combining three circles with the same radius. By drawing these arches recursively, various window patterns can be generated (Fig. 7).

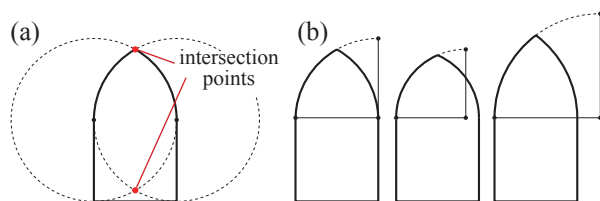


Figure 5 A lancet arch drawn by using two circles (a) and its variations (b)

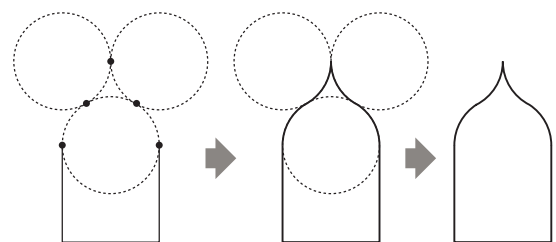


Figure 6 An ogee arch

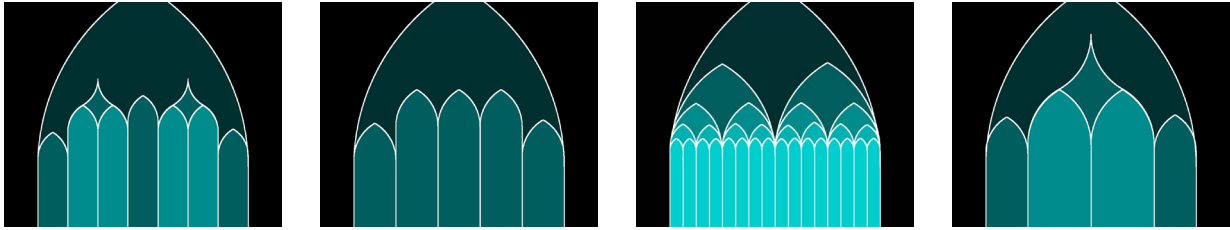


Figure 7 Examples of arch arrangements

Foil

One of the characteristic motifs in Gothic style is a foil, a lobe decoration where cusps (discussed later) create a near-circle curve. It can be generated by drawing arcs at intersection points with circularly arranged circles, as shown in Figure 8. Three lobes (trefoil) and four lobes (quatrefoil) are used frequently in every part of traceries, and multiple foils with more than five lobes are sometimes used as the top rosette of windows.

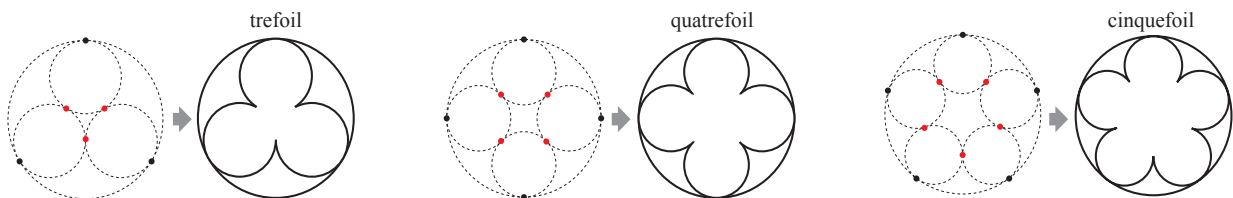


Figure 8 Examples of foil motifs

Dagger

A dagger is a bilaterally symmetric motif that consists of four circles. It is a feature of the late Gothic period. One side is usually tapered, and another side is rounded (Fig. 9 right). To draw a dagger, circles on the tapered side must be placed without an intersection between P and T on the center axis. An algorithm shown in Figure 9 has been used in this research.

Algorithm1 Dagger

```

Declare a variable  $r$  // a radius of the temporary circle
Define a center axis on the current field
Detect which side of the axis is larger
Find  $P$  on the center axis
Place a circle  $A$  on the larger side of the center axis at a distance of  $I$ 
while a radius of the circle  $B > 0$ 
  for all pixels on the arc of  $A$ 
    Set a circle  $B$  at the current pixel temporarily with the radius  $r$ 
    if  $B$  is tangent to  $T$ , and  $B$  does not overlap with the axis
      Fix  $B$  at the current position break
    end if
  end for
   $r \leftarrow r -$  a certain decremental value
end while
Duplicate  $A$  and  $B$  to the opposite side of the center axis
  
```

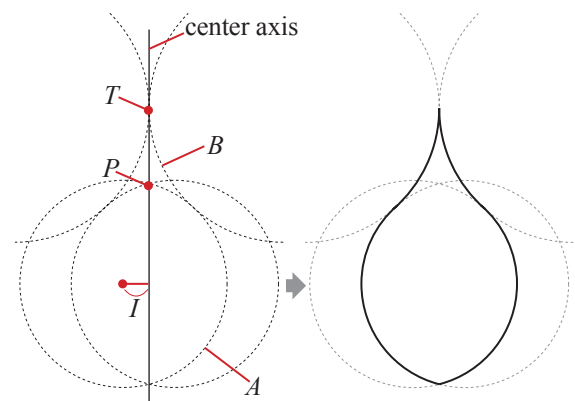


Figure 9 Pseudo-code for drawing a dagger (left), and its circle arrangements (right)

In the algorithm, to detect whether the tapered arcs overlap with the center axis, all pixels in the current field are divided to either the left or right of the center axis. Then, it is possible to detect whether the arcs are aligned correctly by using the left/right information.

Algorithm2 Mouchette

```
Declare a variable  $r$  // a radius of the temporary circle
Find a base circle on the current field
Place a head circle on the base circle at the head point  $H$ 
while  $r > 0$ 
  for all pixels on the arc of the head circle
    set a tail circle at the current pixel temporarily with the radius  $r$ 
    if the tail circle is tangent to the tail point  $T$ 
      if the tail circle does not protrude from the base circle between  $H$  and  $T$ 
        Fix the tail circle at the current position break
      end if
    end if
  end for
   $r \leftarrow r -$  a certain decremental value
end while
```

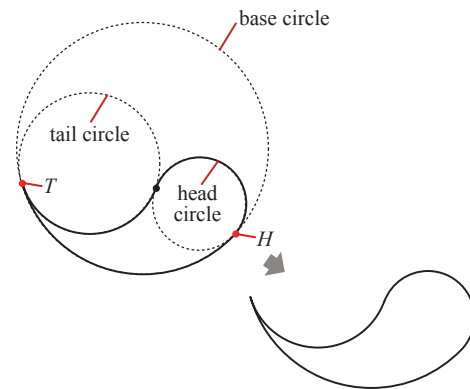


Figure 10 Pseudo-code for drawing a mouchette (left), and its circle arrangements (right)

Mouchette

A mouchette (Fig.10 right) is a curved version of a dagger, and it is paisley shaped. Like the dagger, the mouchette became popular with late Gothic-style architecture. A mouchette consists of three circles, and it is drawn by using the algorithm shown in Figure 10. There is another version of the mouchette with a hemisphere head (Fig. 11) that can be drawn with the abovementioned algorithm. Unlike the mouchette in Figure 10, however, a head circle should be placed in contact with the head point H to be of practical use for the main algorithm and corresponding arrangement of the motif in an appropriate position. Hence, a head circle is set at H temporarily. Then, an intersection point of the temporary head circle and the base circle is calculated using a simultaneous equation. Between the two solutions of the equation, the point closest to tail point T is adopted as the center position of the head circle. The process follows that used for pseudo-code on Figure 10.

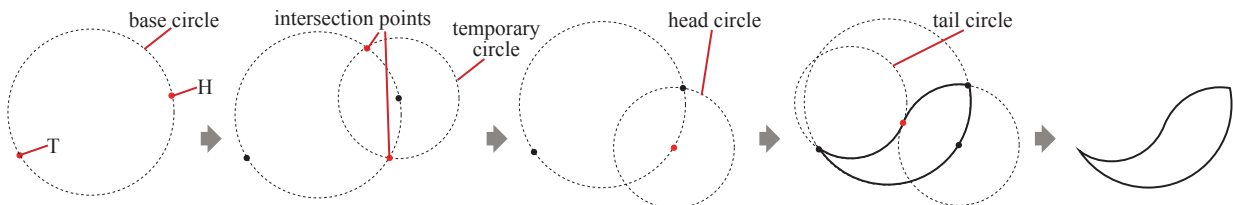


Figure 11 A mouchette with a hemisphere head

Cusp

A cusp is a spur decoration that emerges where two curves meet (Fig. 12) inside a motif, and it is used to distinguish a motif with a Gothic-like thorny shape. Spaces separated by cusps are called foils, as mentioned previously. A cusp usually emerges at the intersection point of circles or lines, or at the midpoint of arcs or lines. However, there is no consistency regarding its location in the intersection or midpoint because the position depends on the motif. In this research, the position of the cusp was predefined manually, and the start point and end were set based on the position. Then, the intersection point of two circles was calculated as a tip of the cusp. When the current motif is to be used as the next field to be filled with smaller sub-motifs, cusps are not drawn.

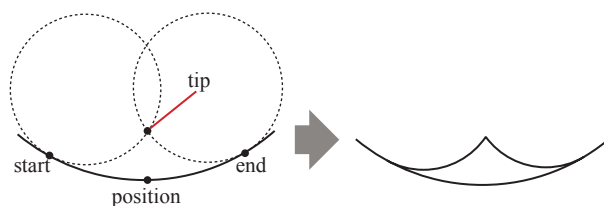


Figure 12 A cusp

Other Motifs

In addition to the motifs described above, some others were used in this research. They can be drawn based on the algorithms mentioned in this section. Figure 13 shows examples of these motifs.

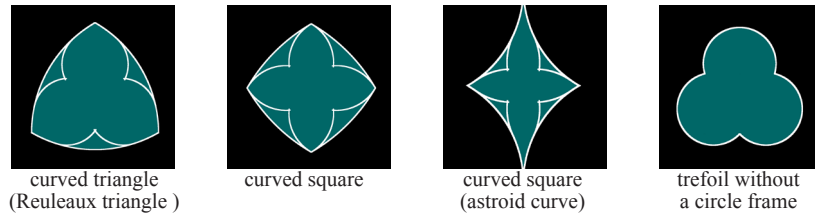


Figure 13 Examples of other motifs

4-2 Motif Arrangements

This research used an approach whereby an algorithm arranges an appropriate motif in the given field, as described previously in Figure 4. In short, a motif is separated from the current field space, and the rest of the field is used as blank space for arranging another motif. This process is repeated until the rest of the field becomes smaller than a certain size, as described in the Figure 14.

Algorithm 3 Motif Arrangement

```

while the radius of the inscribed circle > threshold
  Set arrangement type (e.g., symmetry) based on the parent motif
  Set a motif type
  Set properties (e.g., position, rotation, and size)
  Recursive call of the algorithm 3
  Find the largest inscribed circle (Figure 15)
end while
  
```

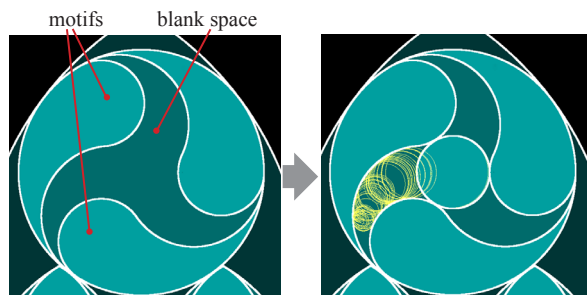


Figure 14 Pseudo-code for motif arrangements. The next motif is placed based on the inscribed circles (yellow lines on the bottom right image)

Algorithm 4 Inscribed Circle

```

Declare integer variables  $c, max$ 
Declare a float variable  $r$  //a radius of the temporary circle
for all pixels on the current field
  Initialize all variables with 0 (zero)
  while  $c = 0$ 
    Set a temporary circle with radius  $r$  at the current pixel
    for all pixels on the temporary circle
      if the current pixel on the circle is outside of the current field
         $c \leftarrow c + 1$ 
      end if
    end for
    if  $c = 0$  and  $r > max$ 
       $max \leftarrow r$ 
    else if  $c > 0$ 
      break
    end if
     $r \leftarrow r + \text{a certain incremental value}$ 
  end while
end for
return  $max$  //a radius of the largest inscribed circle
  
```

Figure 15 Pseudo-code for finding an inscribed circle

To detect whether the current field is large enough for setting a motif, the largest inscribed circle within the current field must be searched to detect the size of its radius compared to a designated threshold value (Fig. 15). If the radius is smaller than the value, the current field is considered overcrowded, and a motif is no longer placed in it (Fig. 14 bottom). In this way, there is no problem even if the current field has an isolated space like in Figure 16 because all pixels in the field are evenly scanned to seek the largest space.

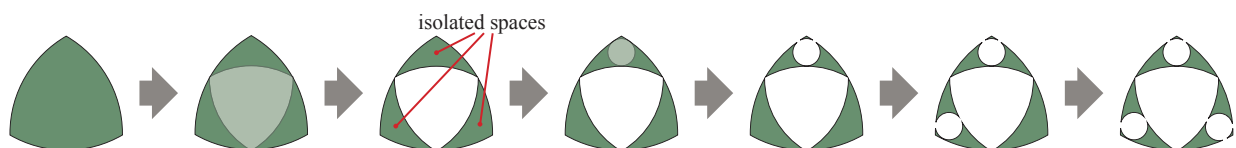


Figure 16 An example of isolated spaces

The biggest problem in this approach is how the algorithm selects the next motif for placement. Therefore, the following two properties were inspected in this research. First, the current field was assessed for symmetry. This information is inherited from the type of parent motif, namely, the current field. A symmetry detection algorithm was not used in this research. If the current field reflected bilateral symmetry, motifs tended to be arranged in the same manner (Fig. 17a). Similarly, motifs reflecting rotational symmetry, such as a precise circle and a curved triangle, are shown in Figure 17b. When the current field was characterized by rotational symmetry, motifs were placed clockwise or counterclockwise circularly in most cases. Incidentally, a rest field after the motif arrangement was considered as symmetry only when the symmetrical motif was placed in the center position in the current symmetrical field (Fig. 17c). In this case, the current field maintained its symmetrical status, and next motifs tended to be arranged in the same manner.

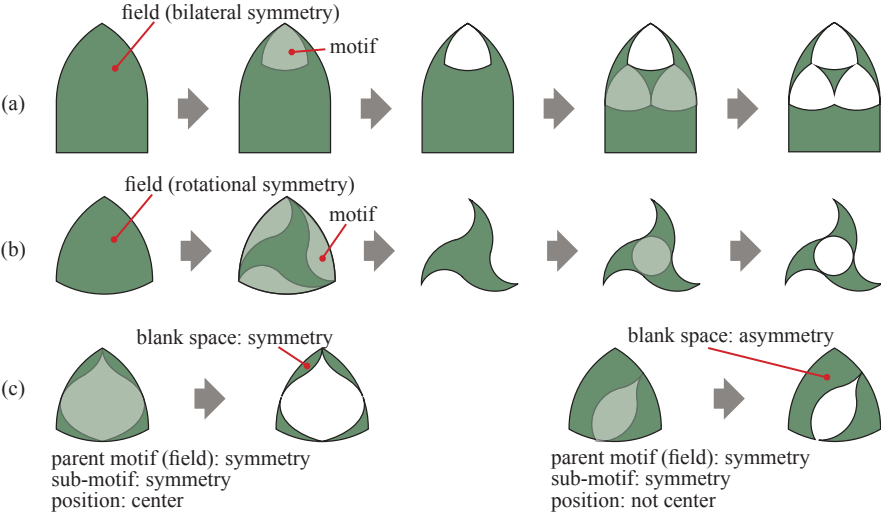


Figure 17 Various conditions in motif arrangements

Second, the current field was inspected for roundness by placing a few of the inscribed circles in the current field, and calculating the change rate of the radius in the circles. For example, Figure 18 shows the large difference between circles A and B with a round shape and the small difference between circles C and D with a slender shape. In the case that the current field is more slender than a certain threshold value, the next motif is limited to slender motifs.

Repeating the abovementioned processes recursively can fill a given enclosed 2D space efficiently (Fig. 19). Placement position is limited in some motifs. For example, sub-arches are usually located in the lower part of a tracery. Therefore, the motif should be placed below a certain threshold value, and other motifs should be placed over the threshold. Moreover, the rate of emergence in each motif depends on the parent motif (shape of the current field). Therefore, priority rankings are applied for each motif (Table 1), and subsequent motifs are selected based on rankings combined with random numbers.

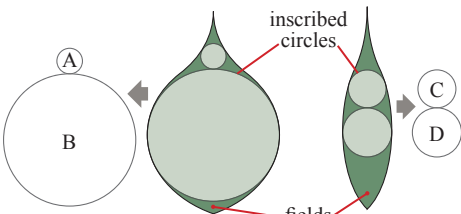


Figure 18 Roundness inspection

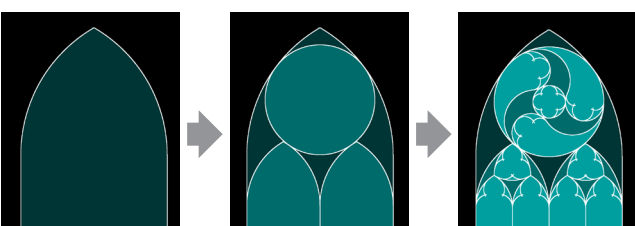


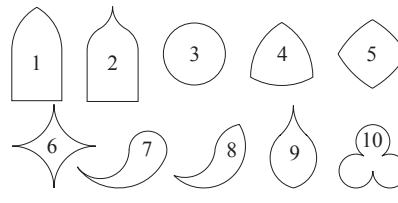
Figure 19 A process of motif arrangements

Table 1 An example of a rankings list of the sub-motif priority. The list shows the initial status when a field space is given. The priorities vary depending on their geometrical conditions in the actual algorithm.

motif No.	high ← sub-motif priority → low									
1	3	4	5	9	8	7	10	1	2	4
2	5	1	10	2						
3	8	7	9	3	4	6	5			
4	8	9	4	3	5	10				
5	8	6	9							

motif No.	high ← sub-motif priority → low									
6	9	5								
7	8	7	3							
8	8	7	3							
9	9	8	7	3						
10										

index of motif No.



Algorithm 5 Rendering

```

Declare and initialize a height map array  $HM \leftarrow 0$ 
for all pixels on the screen  $(x, y)$ 
  for all circles and lines
    if the current pixel within an edge of the circle or line
      if the current pixel within an arc of the circle or a segment of the line
        Calculate the height value at the current pixel  $(0 < \text{value} < 1)$ 
        if the height value  $> HM[x][y]$ 
           $HM[x][y] \leftarrow$  the current height value
        end if
      end if
    end if
  end for
end for

```

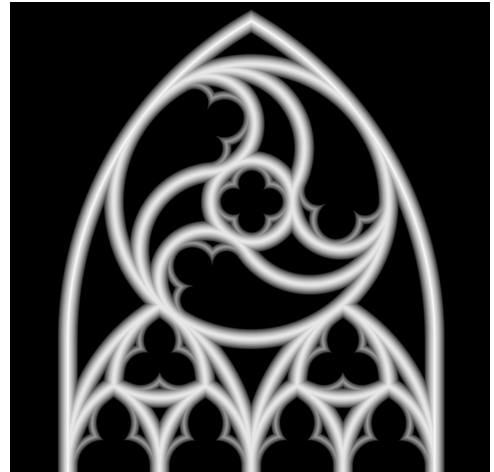


Figure 20 Pseudo-code for rendering

Figure 21

5. Implementation and Rendering

The algorithms mentioned above were implemented with use of C programming language. External libraries were not used for the algorithms. Further, the software used is characterized by a simple structure that can produce an image file alone.

With regard to the rendering process, a semi-3D approach was used in this research. Arranged motifs by using the abovementioned algorithm have 2D frame data only, and they do not provide 3D concavity and convexity information. However, relief-like images can be represented with 3D pseudo-shading. To render such images, normal vectors are applied to the arcs and lines of the motifs. Then, a height-map for shading process can be drawn by using the algorithm shown in Figure 20. The algorithm can generate a black and white image, as shown in Figure 21. In Figure 21, a black area indicates a lower part of the molding profile, and a white area indicates a higher part. Based on the height-map, normal vectors $N(N_x, N_y)$ are calculated as follows.

$$\begin{aligned}
 N_x &= H_x - H \\
 N_y &= H_y - H
 \end{aligned}
 \tag{1}$$

Here, H is the height value in the current pixel. H_x and H_y are height values calculated in the positions that have shifted from the current pixel according to certain distances in the X- and Y-directions, respectively. The normal vectors N calculated using the above formula 1 face inward or outward in relationship to the arcs or lines. In addition, changing directions by adding a Z-vector element in the current N enables representation of concavity and convexity. The resultant profile appearance is variable depending on how normal vectors are applied to the arcs and lines (Fig. 22).

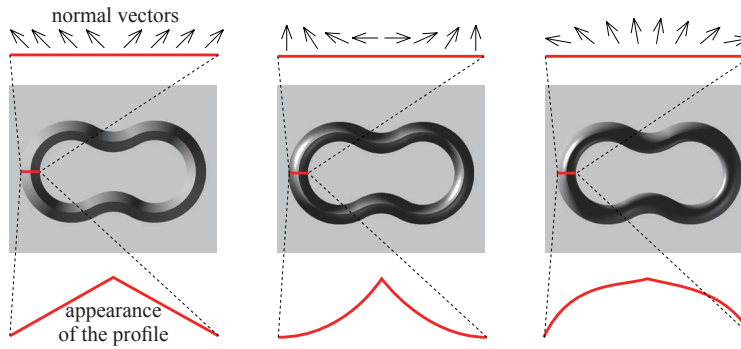


Figure 22 Normal vectors (top arrows), their resultant images (middle), and their appearance of molding profiles (bottom)

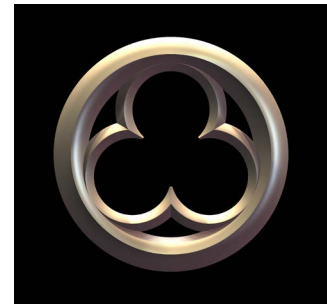


Figure 23 An example of shading

To perform a shading process, a pseudo-3D coordinate space must be considered, and one or more light sources must be placed in the space. Then, the luminance value in the current pixel may be calculated using the light vector(s) and the normal vector N . Figure 23 shows an example of the resultant image of this shading method. In the image, Lambert's cosine law and Phong's highlight model were used for shading. Shadows can be drawn by emitting another vector toward the light source from the current pixel. If the vector is interrupted by a higher part of the tracery profile within a certain distance, the current pixel is considered to be inside of a shadow.

6. Result, Discussion and Conclusion

Figure 24 shows various resultant images generated by using the algorithm in this research. Motifs were selected and arranged according to the algorithm mentioned in section 4. Several types of motifs were used; nevertheless, differences in formation patterns generated a variety in types of traceries. Since the arrangement was determined using random numbers, few results were exactly the same for existing traceries. However, all conformed to Gothic-style (noted in section 3), and they look authentic in appearance.



Figure 24 Resultant images

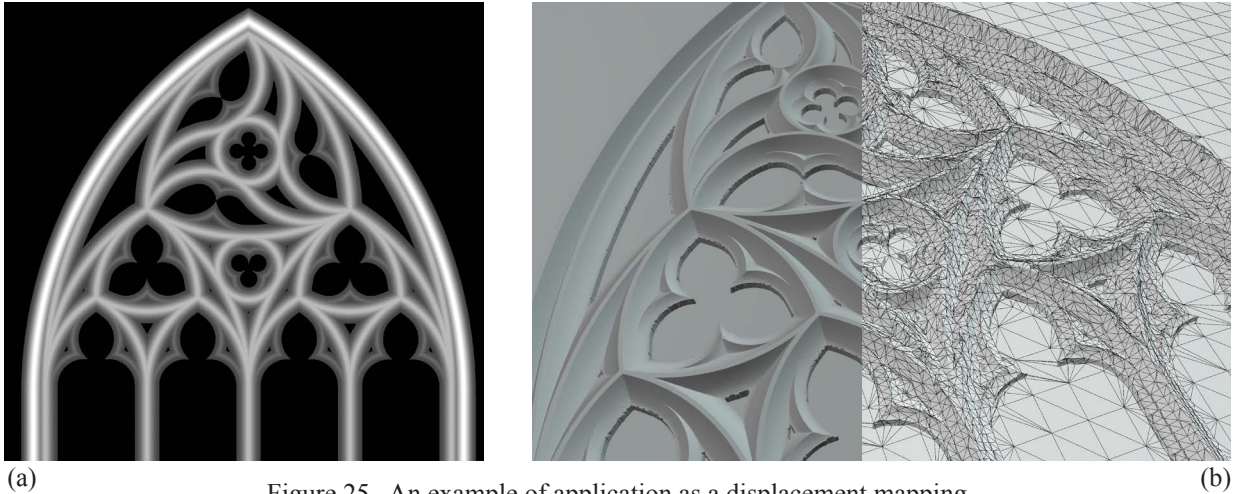


Figure 25 An example of application as a displacement mapping. An exported B/W heightmap (a), and a rendered image (b).

As an algorithm application, a height-map of tracery profiles can be output as a black and white image, as shown in Figure 25a; they can be used for displacement mapping of commercial 3D software (Fig. 25b). This technique is expected to expand applications of the algorithm to various fields, such as architectural simulation and digital fabrication.

As discussed previously, an approach for procedural generation of complicated Gothic traceries was attempted in this research. Limitations of the current state and future works follow.

6-1. Limitations

The current algorithm has some limitations. Although the algorithm can arrange an appropriate motif in the given enclosed 2D space, the shape of a rest field separated by a motif is not recognized by the algorithm. The algorithm contains only initial shape information inherited from the parent motif. Therefore, the case in which the shape of the rest field itself is another motif is overlooked. For example, the field separated by the motif in Figure 26 looks similar to another motif (mouchette). The algorithm in this research does not detect this likeness, and tries to fill the blank space by another motif inefficiently, though this unclear figure-ground relationship is also characteristic of Gothic style. To realize this feature, the algorithm would have to recognize the shape of the blank space.

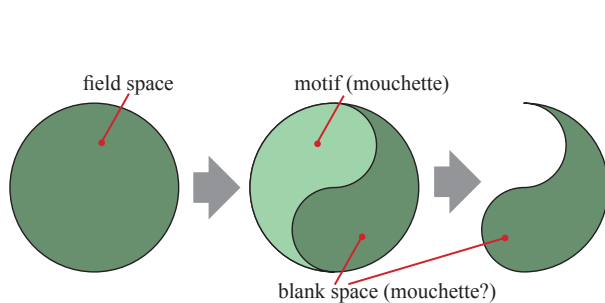


Figure 26 A motif-like shape found in the blank space

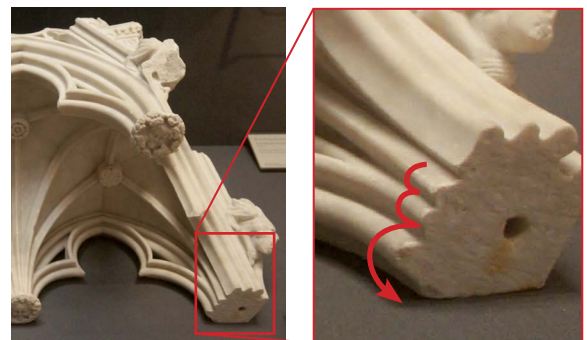


Figure 27 An example of a molding profile (Musée du Louvre, *Photo by author)

In addition, the rendering process has a limitation. Since the algorithm in this research renders images by using height data exclusively, it is impossible to render a complicated frame appearance when the profile turns inward, as shown in Figure 27. To correct this limitation, data must be modeled in 3D.

6-2. Future Work

A greater variety of motifs and flexible arrangements is necessary for representing Gothic style with greater precision. In addition, a parametric control of each motif would enable imitations of a certain era in Gothic style. For instance, mouchettes and daggers became more slender as time progressed, especially in the late Gothic period. Therefore, adopting some intuitive parametric controls, such as “slenderness” and “roundness,” would be beneficial for the usability of the algorithm. Besides, adding interactivity and a user interface would be valuable; such additions would assist in the design of complicated tracery.

In addition to Gothic style, numerous ornament styles can be represented by combining circles. For example, elegant curved motifs found in Art Nouveau and scroll patterns found in various ornamental styles can be approximated using a combination of circles. Attempting to apply the approach used in this research to other styles may help represent a wide variety of ornamental designs.

Acknowledgment

This work was supported by JSPS KAKENHI Grant Number 24700106.

References

- [1] Alexander, H. (1975) *The Computer/Plotter and the 17 Ornamental Design Types*, Proceedings of ACM SIGGRAPH'75, pp.160-167.
- [2] Bloomer, K. C. (2000) *The Nature of Ornament: Rhythm and Metamorphosis in Architecture*, W. W. Norton & Co. Inc.
- [3] Cole, E. ed, (2002) *The Grammar of Architecture*, Bulfinch.
- [4] Cowen, P. (2005) *The Rose Window: Splendor and Symbol*, Thames & Hudson.
- [5] Ebert, D. S., Musgrave, F. K., Peachey, D., Perlin, K., and Worley, S. (2003) *Texturing and Modeling: A Procedural Approach*, 3rd Ed., Morgan Kaufmann Publishers.
- [6] Freeman, E. A. (1851) *An Essay on the Origin and Development of Window Tracery in England: With Nearly Four Hundred Illustrations*, J.H. Parker.
- [7] Hart, S. (2010) *Medieval Church Window Tracery in England*, Boydell Pr.
- [8] Havemann, S. and Fellner, D. (2004) *Generative Parametric Design of Gothic Window Tracery*, 5th International Symposium on Virtual Reality, Archaeology and Intelligent Cultural Heritage (VAST 2004), pp. 193-201.
- [9] Kaplan, C. S. and Salesin, D. H. (2004) *Islamic Star Patterns in Absolute Geometry*, ACM Transactions on Graphics, v.23 n.3, pp.97-119.
- [10] Kaplan, M. and Cohen, E. (2003) *Computer Generated Celtic Design*, Proceedings of the 14th Eurographics workshop on Rendering, pp.9-19.
- [11] Lundy, M. (2001) *Sacred Geometry*, New Ed., Wooden Books Ltd.
- [12] Müller, P., Wonka, P., Haegler, S., Ulmer, A. and Gool, L. V. (2006) *Procedural Modeling of Buildings*, ACM Transactions on Graphics (TOG), v.25 n.3, pp.614-623
- [13] Parish, Y. I. H. and Müller, P. (2001) *Procedural Modeling of Cities*, Proceedings of ACM SIGGRAPH2001, pp.301-308.
- [14] Whitehead, J. (2010) *Toward Procedural Decorative Ornamentation in Games*, In Proceedings of the Workshop on Procedural Content Generation in Games (PCGames'10). ACM, 9:1-9:4
- [15] Wong, M. T., Zongker, D. E., and Salesin, D. H. (1998) *Computer-Generated Floral Ornament*, Proceedings of ACM SIGGRAPH'98, pp.423-434.