THE GENERATION OF CHINESE ICE-RAY LATTICE STRUCTURES FOR 3D FAÇADE DESIGN

Rudi Stouffs¹, Mark Wieringa²
r.m.f.stouffs@tudelft.nl¹, m.wieringa@planet.nl²

Department of Building Technology, Faculty of Architecture, Delft University of Technology, Delft, PO Box 5043, 2600 GA Delft, The Netherlands.

ABSTRACT

Stiny’s shape grammar for the generation of Chinese ice-ray lattice structures is revisited and its shape rules, including additional rules for augmenting a polygon by inscribing a triangular shape, adapted to apply to 3D surfaces. The constructive process of rule application for twisted polygons is described, the implementation of the grammar is shortly described, and its use is illustrated in the design of a façade, the shape of which is defined as a twisted polygon.

Keywords Design Generation, Ice-ray Lattice Designs, Shape Grammars, Twisted Surfaces.

1. INTRODUCTION

The motivation for writing this paper comes from the design of a building façade using Chinese ice-ray lattice structures (Figure 1). The inspiration for the design came from the image of an iceberg, which led to the study of patterns that occur within ice as the basis for the façade design. Ice-ray lattices are a particular form of traditional Chinese lattices constructed between 1000 BC and 1900 AD, as catalogued by Daniel Sheets Dye (1949), which do not have a regular structure, but rather mimic the ice-lines formed in the ice-formation process. The Federation Square building in Melbourne by Lab Architecture Studio served as an example in the design of the façade, although the pattern used in the Federation Square building is far more regular. The design of the building in Figure 1 dictated the use of twisted polygons to define the shape of the façade. While Chinese ice-ray lattice structures can be projected onto the twisted polygons, here a non-projected solution based on solid geometry is considered.

Figure 1. An elevation of the building.

Stiny (1977, 2006) defines a parametric shape grammar for the generation of Chinese ice-ray lattice designs. Stiny’s exemplar grammar has four constructive rules allowing for a convex polygon to form two new convex polygons, with approximately equal areas, by placing a single line between two of the original polygon’s edges.
Specifically, the rules state that any triangle, convex quadrilateral or convex pentagon, with area greater than some given constant, can be augmented once by placing a line between two of its edges to form, respectively, a triangle and a convex quadrilateral, a triangle and a convex hexagon, two convex quadrilaterals, or a convex quadrilateral and a convex pentagon. Stiny also suggests some additional constructive rules for allowing polygons to be augmented by inscribing a triangular or other polygonal shape, generating other, slightly more complex, ice-ray designs.

Stiny’s shape grammar is designed to construct traditional Chinese ice-ray lattice designs, but can be adapted and applied to conceive more contemporary designs. Lab Architecture Studio also considers a fractal pattern, similar to a Chinese ice-ray pattern, in the design of the façades of the SOHO Shangdu building, to be completed in Beijing in 2007. These façades are also not flat but are folded along some of the lines of the fractal pattern.

In this paper, the use of Stiny’s ice-ray grammar rules is considered in the design of a façade consisting of 3D surfaces. For this purpose, these shape rules, including the additional rules for augmenting a polygon by inscribing a triangular shape, are adapted to apply to non-flat surfaces, such as twisted surfaces. The four original shape rules each specify the placing of a single line between two edges. In a twisted polygon, such a line can be placed in a similar way leading to two new twisted polygons. The additional shape rules for inscribing a triangular shape specify the placing of three lines extending from a triangle to each touch one of the edges of the original polygon. Thus, the inscribed shape is connected to the original twisted polygon by three points, forming a (flat) triangle.

2. SHAPE RULES

Figure 2 shows the original shape rules in Stiny’s grammar. These are parametric shape rules, they apply to any triangle (rule 1), convex quadrilateral (rules 2 and 3) or convex pentagon (rule 4), respectively. Note that Stiny’s shape rules are augmented with labelled points in order to guide the generation process and to prevent rule application to apply to the same shape more than once. As a result, Stiny also includes one extra rule – a termination rule – to erase a labelled point. These are omitted here (see below for an explanation in light of the implementation).

![Figure 2. Four constructive rules that split a triangle, convex quadrilateral or pentagon into two new convex polygons (either a triangle and a quadrilateral, a triangle and a pentagon, two quadrilaterals, or a quadrilateral and a pentagon) by placing a single line between two of the original polygon’s edges.](image-url)
Figure 3 shows three additional shape rules that inscribe triangular shapes into a triangle (rule 5), convex quadrilateral (rule 6) or convex pentagon (rule 7), respectively. Stiny also suggests allowing for convex polygons to be augmented by inscribing pentagonal (or hexagonal) shapes. This suggestion is not retained as it is not generally applicable in the context of a twisted surface, whereas the seven shape rules that are presented here are. In case of rules 1 through 4, a single line is placed between two points on two of the original polygon’s edges. Such a line can always be constructed. In case of rules 5 through 7, a triangular shape is constructed that connects in the plane of the triangle to three points on three of the original polygon’s edges. Since three points always (uniquely) define a plane, such a triangular shape can also be constructed (see figure 4 for the case of a twisted quadrilateral).

![Figure 3](image)

**Figure 3.** Three additional constructive rules that inscribe a triangular shape between three edges of a triangle, convex quadrilateral or pentagon.

![Figure 4](image)

**Figure 4.** The construction of a triangular shape inscribed in a twisted convex quadrilateral, starting from three points on three of the twisted quadrilateral’s edges.

### 3. RULE APPLICATION

The implementation described below is not of an actual shape grammar, that is, it does not adhere to the definition of a shape as any “finite arrangement of straight lines of limited but nonzero length” (Stiny, 1977), nor does it rely on shape recognition algorithms. The implementation instead uses the notion of a polygonal face as defined by the CAD software application it is embedded in. Furthermore, the generation is guided by the order in which the faces are created. In this way, no auxiliary information, such as labelled points, is required in the parametric rules.

Stiny’s shape rules are constrained in two ways. First, they only apply to polygons with area greater than some given constant. Secondly, in the case of rules 1 through 4, the resulting polygons have approximately equal areas. Additional constraints on the lengths of the polygon’s edges are necessary in order to ensure that an already short edge is not further subdivided. In the current implementation, all constraints are
expressed in terms of the lengths of the original polygons’ edges (Figure 5). Figure 6 illustrates the constraints on the endpoints of the single line placed in rules 1 through 4. Each endpoint lies within a section of the polygon’s edge that is defined by two parameter values specified by the user (Figure 5), e.g., 0.35 and 0.65 where the edge’s endpoints have values 0 and 1. A random number is generated between these two values that specifies the parameter value of the point within the edge; in the implementation, a single random value applies for both edges, as illustrated in Figure 6.

Figure 5. The user interface to specify the minimal length constraint (“Minimale Lengte”) and the parameter values (“Min range” and “Max range”) that serve to define the section of the polygon’s edge wherein the endpoint of the single line (rules 1 through 4) lies.

Figure 6. The construction of the single line placed between two edges of the original triangle, quadrilateral or pentagon. Each endpoint lies within a section of the polygon’s edge that is defined by two parameter values specified by the user.

Figure 7 illustrates the constraints on the endpoints of the inscribed triangular shape in rules 5 through 7. Each triangular shape is constructed from six points, three of which define the connecting points to the edges of the original polygon, and thereby the plane that embeds the triangular shape, and the other three points define the respective planes, each perpendicular to the plane of the triangle, that embed the edges of the triangular shape. Here too, a random number is generated that specifies the parameter value of a point within the edge and lies between the user-defined parameter values. In the case of a triangle (rule 5), these three points define the connecting points, the other three points are constructed using the same random value applied to the part of the edge between the connecting point and the original edge’s vertex with parameter value 1. In the case of a quadrilateral (rule 6), the same construction is applied, with the exception of the first edge, where the first point defines one of the auxiliary points and the second, connecting point lies between the first point and the original edge’s vertex
with parameter value 0. In the case of a pentagon (rule 7), further variations on this construction technique are considered in order to define the three connecting points and the three auxiliary points (Figure 7).

![Figure 7. The construction of the triangular shape inscribed between three edges of the original triangle, convex quadrilateral or pentagon. Each triangular shape is defined by three connecting points and three auxiliary points. Each of these points lies within a section of the polygon’s edge that is defined by two user-specified parameter values, or within a similarly constructed section of part of the segment as defined by a previous point.](image)

4. GRAMMAR IMPLEMENTATION

The ice-ray grammar for twisted surfaces is implemented in MEL (Maya Embedded Language), the scripting language of the Autodesk® Maya® software. The generation process is guided by the order in which the polygonal faces are created. Rule selection is naturally dependent on the number of sides to the polygon and is further defined randomly. Also, the selection of edges containing the endpoints of the single line or triangular shape is randomly defined among the number of possible permutations. Rule application results in one or more new polygonal faces, but each edge that is created is also separately represented as a linear curve. Each face and curve is assigned a level designator in the process. The level designator of a face is the level designator of the original face it is derived from through rule application plus one. Thus, this level designator corresponds to the number of rule applications that have directly led to the creation of this face (Figure 8). The level designator of a curve equals the level designator of the face it is created as an edge of. Upon completion of the generation process, each linear curve is extruded according to a profile, where the size of the profile (inversely) reflects upon the level designator of the curve (Figure 9). As such, the generation specifies not only the final form of the façade but also its building technical structure, which can then be exported and analyzed for stability using a finite element analysis application. Figure 9 shows an exemplar generation. Figure 10 shows the stability analysis of an (different) exemplar structure using DIANA.
Figure 8. A hierarchy of rule applications (twice rule 3 and once rule 2) and resulting polygonal faces. The number in each face is the level designator and corresponds to the depth of the face in the hierarchy.

Figure 9. The design of a façade using the ice-ray grammar. The initial shape of the façade is a twisted convex quadrilateral.

5. DISCUSSION

Stiny’s ice-ray grammar has been implemented repeatedly, either as a specific implementation (e.g., Liew, 2001) or as an exemplar application of a (more general purpose) shape grammar interpreter (e.g., McCormack and Cagan, 2002). In this paper, another specific implementation of this grammar is described, however applied to 3D surfaces. Both the four original shape rules and the additional shape rules for inscribing a triangular shape apply equally to twisted polygons. Rather than projecting the resulting lattice structures onto the 3D surface, the application of the shape rules to
twisted polygons is described using solid geometry. The four original shape rules, each, specify the placing of a single straight line between two edges of the original (twisted) polygon, thereby, creating two new (twisted) polygons. The additional shape rules, each, specify the inscription of a triangular shape in the form of three straight lines extending from a triangle to three edges of the original (twisted) polygon, thereby, creating a planar triangle and three new (twisted) polygons. These rules could be easily extended to apply to polygons with numbers of sides greater than five, such as hexagons. Allowing for polygons with larger numbers of sides could help in better approximating the original 3D surface with a twisted polygon. Alternatively, the original 3D twisted surface can also be approximated as an initial collection of twisted polygons with common edges.

**Figure 10.** The stability analysis of an exemplar (twisted) ice-ray lattice structure using DIANA, showing the deflection under a uniform load on the top edge of the structure.

The exemplar façade design in Figure 9 clearly illustrates the limitations of the current implementation. Obvious is the variation in the sizes of the polygons that constitute the design. This is a result of the fact that rule application is constrained by the specification of a minimal length for each of the edges of the original polygon. While this may be sufficient in the case of a triangle – a short edge results in a small surface area, even if it is an elongated triangle – it is overly restrictive in the case of a quadrilateral or pentagon. While the constraint could be reformulated in terms of the surface area of the polygon, some constraint(s) on the lengths of the polygon’s edges remains necessary in order to ensure that an already short edge is not further subdivided. However, rather than restricting rule application altogether, an edge length constraint should (only) apply when selecting the edges that the rule application applies to, that is, the edges that will be subdivided by the rule application.

Also obvious in Figure 9 is the high amount of inscribed triangles. This is a direct result of the random selection of the rule that applies to any polygon, from those rules that apply to a polygon with the same number of sides. In the case of a triangle or convex pentagon, two rules may apply, one placing a single line and the other
inscribing a triangular shape. The chance for each rule to be selected is one out of two. In the case of a convex quadrilateral, three rules may apply, and the chance for the rule inscribing a triangular shape to be selected is one out of three. Instead, the rule selection process can be modified in order to favour the original shape rules or, alternatively, the user could be offered the opportunity to pick which rule to apply at every step in the generation process.

Less obvious in Figure 9 is the fact that in the current implementation a single random value is generated that specifies the parameter value of a point within an edge or part thereof, for all such points that assist in the application process of the rule. It would be straightforward to alter the implementation such that a different random value is generated each time a point is selected.

As stated before, the implementation described is not of an actual shape grammar as it does not rely on shape recognition algorithms to identify the polygons, but instead relies on the polygonal face objects that are defined by the CAD software application it is embedded in. This does not influence the generation process but it does simplify the implementation very much. Implementing it instead as a real shape grammar would only serve the flexibility of the implementation and be worthwhile if the implementation also allows for other shape grammars. On the other hand, using a grammar, or a collection of design rules, as the basis of the generation process does aid in the understanding of the design and construction process of ice-ray lattice structure. As Stiny (1977) describes, “the steps in the ice-ray lattice generation […] could well comprise the frames in a motion picture of the artisan creating his design.”

6. CONCLUSION

The shape rules in Stiny’s ice-ray grammar can be adapted to apply to 3D surfaces. A variety of constraints can be considered for rule application in order to guide design generation. The implementation described above is not of an actual shape grammar; instead, it combines the simplicity of the shape rules with the strengths of the underlying CAD software application.

7. ACKNOWLEDGMENT

This paper extends on a short paper presented at the eCAADe 2006 conference, Volos, Greece, 6-9 September 2006. The implementation of the generative system and its application to 3D façade design form part of an MSc final design project.

8. REFERENCES

