```
**************************************************
**************************************************
***                                            ***
***  Name:                                     ***
***                                            ***
***  Project:    1        Programmer:  MWK     ***
***                                            ***
***  File Name: DISSER.PUB[DOC,JEG]            ***
***                                            ***
***  File Last Written:  15:46 13 Jul 1973     ***
***                                            ***
***  Time: 20:01          Date: 15 Jul 1973    ***
***                                            ***
***            Stanford University             ***
***       Artificial Intelligence Project      ***
***         Computer Science Department        ***
***           Stanford,  California            ***
***                                            ***
**************************************************
**************************************************
```

Outline

Figures so far

Section 1

1 inscribed squares
  a Shape grammar
  b a generation
  c language

2 circumscribed squares
  a shape grammar
  b a generation
  c language

3 inscribed triangles with one marker
  a shape grammar
  b language

4 inscribed triangles with four markers
  a shape grammar
  b language

5 reversible figures
  a reversible figure
  b variation

6 reversible figures
  a shape grammar
  b language

7 Huffman coding of reversible figures
  a reversible figure
  b variation

8 Snowflake curve - 6 levels

9 first approx sg for snowflake
  a shape grammar
  b curve in language

10 SG for snowflake curve

11 Peano's curve variation - 3 levels

12 Peano's curve variation
  a shape grammar
  b generation ?

13 Hilbert's curve - levels 1, 2, 3, and 6

14 Hilbert's curve - shape grammar

15 Hilbert's curve derivation

1.1 Definitions

A shape grammar, SG, is a 4-tuple : SG = <Vt,Vm,R,I> where

(1) Vt is a finite set of shapes.

(2) Vm is a finite set of shapes such that $Vt* \cap Vm* = \emptyset$.

(3) R is a finite set of ordered pairs (u,v) such that u is a shape consisting of an element of Vt* combined with one or more elements of Vm and v is a shape consisting of an element of Vt* combined with zero or more elements of Vm.

(4) I is shape consisting of an element of Vt* combined with one or more elements of Vm.

Elements of the set Vt are called terminal shape elments (or terminals). Elements of the set Vm are called non-terminal shape elements (or markers). Elements of the set Vt* are formed by the finite arrangement of an element or elements of Vt in which any element of Vt may be used a multiple number of times in any location, scale, and orientation. The set Vm* is defined similarly. The sets Vt* and Vm* must be disjoint. (For example, Vt could contain a straight line as its only element and Vm could contain a circle as its only element.) Elements (u,v) of R are called shape rules and are written $u \rightarrow v$. u is called the left side of the rule; v the right side of the rule. u and v usually are enclosed in identical dotted rectangles to show the correspondence between the two shapes. I is called the initial shape and normally contains a u such that there is a (u,v) which is an element of R.

A shape is generated from a shape grammar by beginning with the initial shape and recursively applying the shape rules. The result of applying a shape rule to a given shape is another shape consisting of the given shape with the right side of the rule substituted in the shape for an occurrence of the left side of the rule. Rule application to a shape proceeds as follows :

(1) Find part of the shape that is geometrically similar to the left side of a rule in terms of both terminal and non-terminal elements.

(2) Find the geometric transformations (scale, translation, rotation, mirror image) which make the left side of the rule identical to the corresponding part in the shape.

(3) Apply those transformations to the right side of the rule.

(4) Substitute the transformed right side of the rule for the part of the shape which corresponds to the left side of the rule.

The generation process is terminated when no rule in the grammar can be applied.

For any given shape grammar, the dimensionality of the shapes in Vm and Vt and of the geometric transformations used to combine these shapes must be constant. This number is called the dimensionality of the shape grammar. While three-dimensional shape grammars have been used to generate sculpture [Stiny and Gips 1972], in this report only two-dimensional shape grammars are considered. All elements of Vt and Vm will be two-dimensional and all transformations will be planar.

The sentential set of a shape grammar, SS(SG), is the set of shapes (sentential shapes) which contains the initial shape and all shapes generatable from the initial shape using the shape rules. The language of a shape grammar , L(SG), is the set of sentential sets that do not contain any markers, i.e., L(SG) = SS(SG) ∩ Vt*.

Types of shape grammars can be defined by putting further restrictions on the allowable form of shape rules. Two types of shape grammars, non-erasing shape grammars and unimarker shape grammars, are especially useful.

A non-erasing shape grammar is a shape grammar in which the element of Vt* that appears in the left side of each rule appears identically in the right side of that rule. The result of this restriction is that once a terminal is added during the generation process using a non-erasing shape grammar, it canot be erased.

A unimarker shape grammar is a non-erasing shape grammar in which the initial shape contains exactly one marker, the left side of each rule contains exactly one marker, and the right side of each rule contains zero or one markers. The result of this restriction is that each sentential shape of a unimarker shape grammar that is not in the language of the shape grammar contains exactly one marker.

1.2 Examples

In this section some examples of the use of shape grammars in defining various shapes is examined. First, four simple, related shape grammars are presented for pedagogical purposes. Next is a shape grammar that generates a language of reversible figures. In the third part of this section, the use of shape grammars in defining shape-filling curves is explored. Finally, a method for constructing a shape grammar for an arbitrary Turing machine is presented.

1.2.1 Embedded squares and triangles

A very simple (unimarker) shape grammar, SG1, is shown in Figure 1a. Vt contains a square as its only element. Vm contains a circle as its only element. There are two shape rules. The initial shape contains a square inscribed by a circle.

The generation of a shape in L(SG1) is shown in Figure 1b. Because the two shape rules in SG1 contain identical left sides, the two shape rules are applicable in identical circumstances, i.e., wherever there is a square inscribed by a circle. Application of rule 1 to a shape results in the addition of an embedded square and the shrinking of the marker. Application of rule 1 forces the continuation of the generation process as both rules are applicable to the new sentential shape. Application of rule 2 to a shape results in the addition of an embedded square and the removal of the marker, thereby halting the generation process. In the generation shown in Figure 1b, the process is begun with the initial shape, rule 1 is applied three times, and then rule 2 is applied. The language defined by SG1 is shown in Figure 1c.

A somewhat similar shape grammar, SG2, is shown in Figure 2a. The generation of a shape using SG2 is shown in Figure 2b. Where in the generation process using SG1 squares are succesively inscribed, using SG2 squares are successively circumscribed. The language defined by SG2 is shown in Figure 2c. Note that the area contained in the shapes in L(SG1) is constant, where the area contained in successive shapes in L(SG2) doubles.

The purpose of the marker in these two examples may not be apparent. The use of the marker makes the rules applicable only to the most recently added square. If the marker were not used, rule 1 could be applied over and over to the same square. The importance of markers is further illustrated by the next two examples.

The shape grammar SG3, shown in Figure 3a, is similar to SG1 but embeds triangles instead of squares. The placement of the marker in the right side of rule 1 insures that triangles can only be inscribed in the center triangle of the four triangles that are added. The language generated by SG3 is shown in Figure 3b.

In the shape grammar SG4, shown in Figure 4a, the right side of rule 1 contains four markers. This rule allows triangles to be inscribed subsequently in any of the four added triangles. L(SG3) is a small subset of the language generated by SG4 (see Figure 4b). Other languages of inscribed triangles can be generated using shape grammars with different configurations of markers. Markers are important because they restrict rule application to specific parts of a shape and help determine the transformations (eg., scale) allowable in applying the rules.

1.2.2 Reversible figure

A new reversible figure [Gips 1972], similar to the Necker cube, the Schroeder reversible staircase, etc. [Luckiesh 1965] is shown in Figure 5a. The figure can represent two different three-dimensional objects. The central three lines can be perceived either as outer (convex) edges of a cube or as inner (concave) edges where a cube was cut from the closest corner of a larger cube. Either the outer walls of the object appear to have width and be solid or the outer walls appear to have no width and be infinitely thin.
A variation is shown in Figure 5b.

A shape grammar, SG6, that generates these figures is shown in Figure 6a. The shape grammar is similar to SG2; with each additional application of rule 1, a new and larger terminal is added around the outside of the shape. The language defined by SG6 is shown in Figure 6b. Each shape in L(SG6) is a reversible figure.

As a short digression, it is interesting to analyze these reversible figures in terms of a contemporary computer vision algorithm. In particular, how does Huffman's algorithm for interpreting
two-dimensional figures as three-dimensional objects [Huffman 1971],
[Duda and Hart 197x] interpret
these reversible figures ? Are the figures reversible (ambiguous) for this algorithm ? Implicit in Huffman's algorithm is that all ojects have discernible width. Thus for this algorithm the figures are not ambiguous. Only one interpretation of Figure 5a is possible and only one interpretation of Figure 5b is possible. But the two interpretations are different !
All lines that are interpreted as convex in Figure 5a are interpreted as concave in Figure 5b and vice versa. The Huffman labeling of Figures 5a and 5b are given in Figure 7a and 7b.
Following Huffman, a "+" denotes a line interpreted as a convex line in the three-dimensional object, a "-" denotes a concave line, and an "→" denotes a convex line with ....
Because for the algorithm all objects
have width, the convexity or concavity of the central lines of these figures is determined by the number
of surrounding hexagons (i.e., the number of times rule 1 was applied in the generation of the shape). If the number of hexagons is odd, as in Figure 5a, the three central lines are interpreted as convex by the algorithm. If the number of hexagons is even, as in Figure 5b, the three central lines are interpreted as concave using Huffman's algorithm.

1.2.3 Mathematical curves

Shape grammars can be used to define a number of classical mathematical curves. Previously, these curves were defined either analytically or by displaying instances of the curves and giving informal English descriptions. Shape grammars provide a method for the precise, algotithmic specification of these curves that at the same time yields insights about the construction of the curves.

1.2.3.1 Snowflake curve

The first six stages, S1 - S6, of the Snowflake curve [Kasner and Newman 1965] are shown in Figure 8. The Snowflake curve is interesting because in the limit, the area enclosed by the curve is finite while the length of the curve is infinite. (In the limit, the area of the curve is 1.6 times the area of the original triangle. At each successive stage, the length of the curve increases by a factor of 4/3. Clearly, $(4/3)^n$ does not converge as n increases.)

A first approximation of a shape grammar for the Snowflake curve is shown in Figure 9a. Note that the right side of the first shape rule contains two markers and that the initial shape contains three markers. The generation of a shape using SG9 is shown in Figure 9b. Rules 1 and 2 are applicable under identical circumstances. They are applicable at three different places in the initial shape, at four different places in the next shape, etc. Application of rule 1 to part of a shape results in the continuation of the generation process in that part of the shape; application of rule 2 halts the generation process in that part of the shape. As the generation process can proceed to different depths in different parts of the shape, the language defined by SG9 includes not only the completed stages of the Snowflake curve (S1, S2, ...), but also many intermediate curves similar to the shape generated in Figure 9b.

For a shape grammar to define a language containing only completed stages of the Snowflake curve, it cannot allow the generation process to proceed independently in different parts of the shape. The generation process must be controlled to generate the shape uniformly. A shape grammar, SG10, that generates just the curves S1, S2, ... is shown in Figure 10a. The generation of S2 using SG10 is shown in Figure 10b. The strategy implicit in SG10 is to trace around the shape (using rules 1 and 2), expanding lines as the trace proceeds. The asymmetry of the marker forces the genration process to always proceed counter-clockwise around the shape. Whenever a complete trace is made, the generation process can either be halted (by applying rule 4) or allowed to proceed (by applying rule 3) for at least another complete trace. Rule 5 is only applicable to the initial shape. Without rule 5, the language would not include S1. There may well be shape grammars that are simpler than SG10 that generate S1, S2, ...

1.2.3.2 Peano's curve variation

this section needs work !

Peano's curve [Peano 1890] is a curve that passes through every point of the unit square. Peano defined the curve analytically, roughly in terms of a parameter t that varies from 0 to 1 and continuous functions $\alpha(t)$ and $\beta(t)$ defined such that for every (x,y) where $0 < x,y < 1$ there exists a t with $\alpha(t)=x$ and $\beta(t)=y$. Moore [1900] represented Peano's curve geometrically as the limit of a series of curves made up of polygonal arcs. The first curve of the series passes through the center of the unit square. Next, the unit square is subdivided into nine equal squares; the second curve of the series passes through the center of each of these subsquares. The third curve passes through the centers of each of the 81 subsquares of the unit square, etc.

A new variation on Peano's curve is illustrated in Figure 11. The first three polygonal curves of the series are shown with the unit square. The centers of the subsquares that the curves pass through are marked with dots. This curve differs from Peano's curve in terms of the order that the curves pass through the subsquares.

A shape grammar, SG12, that generates exactly this series of curves is shown in Figure 12a.
Examination of the curves reveals that each section of a curve that passes through a subsquare is identical to either the terminals in the left side of rule 1 or the terminals in the left side of rule 2 (or their mirror images).
The effect of applying either rule 1 or rule 2 is to replace the section of a curve that passes through (the center of) a square with a curve that passes through (the centers of) the nine subsquares.
The center of the marker in the left sides of rules 1 and 2 shows the exact location of the beginning of the bottom edge of the terminals added in the right sides of the rules.
As with the generation of the Snowflake curves using SG10, the generation, using SG12, of the nth curve in this series involves the succesive generation of the first n-1 curves of the series. The generation of the third curve (see Figure 11) is shown in Figure 12b.
Both rule 3 and rule 4 are applicable to the initial shape. If rule 4 is applied, the generation terminates yielding the first shape in the series.
If rule 3 is applied, as in Figure 12b, the generation is forced to continue.
Application of
The generation proceeds by expanding (using rule 1 or rule 2) each successive section of the current curve. When the last section of the curve is reached i.e., when the marker reaches the the edge of the unit square, the generation is either halted (by applying rule 4) or forced to continue (by applying rule 3) for another complete trace back along the just generated curve.

1.2.3.3 Hilbert's curve

Hilbert's curve [Moore 1900] is the best known space-filling curve and has appeared in the popular literature of both mathematics [Hahn 1954] and art [Munari 1965], frequently labeled erroneously as Peano's curve. The sequence of curves, Hi, used in the definition of Hilbert's curve is similar to the sequence Pi for Peano's curve, but is generated by recursively subdividing the unit square into four subsquares rather than nine. Curves H1, H2, H3, and H6 are shown in Figure 13. H1 passes through the four subsquares of the unit square, H2 through the sixteen subsquares, etc. The subsquares have been added to the drawings of H1 and H2 as a convenience to the reader.

A shape grammar, SG14, that generates just the sequence of curves Hi is shown in Figure 14. In the generation of curve Hn using SG14, curves H1 ... Hn-1 are first generated.
The generation of H3 using this shape grammar is shown in Figure 15.
The grammar contains two markers, a curved diamond
and a circle. The diamond is used to mark the endpoints of the curve during the generation process. This is necessary because the locations of the endpoints of curve Hi is different than the locations of the endpoints of curve Hi-1 and there are no convenient landmarks.
The circle is used to trace around the curves. The grammar contains seven shape rules. Rule 1 is used at the beginning of the eneration of each Hi to expand the section of the curve in the initial subsquare. Rules 2 - 4 are the core of the shape grammar; they are used to successively expand the section of the curve contained in all but the initial and final subsquares. Rule 5 is used at the end of the generation of each Hi to expand the section of the curve in the final subsquare. Rule 6 is an alternative to rule 1; application of rule 6 causes the erasure of the circle marker and one of the diamond marker and results in the end of the generation process. Rule 7 is used to erase the diamond marker not erased by rule 6. While rule 7 is applicable at each step in the generation, if it is applied prematurely the generation comes to a dead end as it becomes impossible to apply rule 6 and thereby erase the circle marker. The language defined by SG14 contains exactly the curves Hi.

References so far

Duda and P. Hart, Pattern Classification and Scene Analysis,

Gips, J., "A new reversible figure", Perceptual and Motor Skills, Vol. 34, 1972, p.306.

Hahn, H., "Geometry and intuition", Scientific American, April 1954.

Huffman, D., "Impossible objects as nonsense sentences", in B. Metzler and D. Michie (Eds.), Machine Intelligence 6, American Elsevier, New York, 1971.

Kasner, E. and J. Newman, Mathematics and the Imagination, G. Bell and Sons, Ltd., 1949, 1965.

Luckiesh, M., Visual Illusions, Dover, New York, 1922, 1965.

Moore, E.H., "On certain crinkly curves", Transactions of the American Mathematical Society, vol. 1, pp. 72-90, 1900.

Munari, B., Discovery of the Square, George Wittenborn, Inc., New York, 1965.

Peano, G., "Sur un courbe, qui remplit toute un aire plane", Mathematische Annalen, Vol. 36, pp. 157-160, 1890.

Stiny, G. and J. Gips, "Shape grammars and the generative specification painting and sculpture", Information Processing 1971, North Holland Publishing, Amsterdam, 1972. Also, in O. Petrocelli (Ed.), The Best Computer Papers of 1971, Auerbach, Inc., 1972.