On an Alternate Construction Method for Generating Spidrons and New Tiling Patterns Generated by them

T. Gangopadhyay
XLRI
C.H.Area(E), Jamshedpur, India

ABSTRACT
Spidrons have been constructed by repeatedly connecting the alternate vertices of a regular polygon. In the present paper an alternate construction is presented. Also new polygonal designs and tiling patterns are created using these spidrons.

General Terms
Tiling, Algorithm, Turbo C++, Program

Keywords
Spidron, polygon, isosceles

1. INTRODUCTION
In earlier papers, the present author has studied tiling with stars having an odd number of vertices (Gangopadhyay(3],[4)). In the present paper, the author studies tiling patterns using a fractalisesque structure called spidron, first studied by Erdely[2]. An n-part spidron is a figure in plane geometry consisting entirely of an alternating sequence of two isosceles triangles, each with its own base angles[2], [6], [8]. The two triangles are juxtaposed in such a way that together they form a third bigger triangle. The sequence is then drawn at an appropriate angle on a smaller scale and the process is repeated ad infinitum. A regular n-sided polygon can be split into n n-part spidrons[7]. A 6-part spidron is displayed in Figure 1. Erdely[2], has called the same structure a semispidron. The usual method of constructing spidrons consists of starting with a regular polygon, connecting its alternate vertices and producing a smaller version of the regular polygon inside. The process is then repeated for the smaller polygon and so on Ad infinitum. This divides the polygon into spidrons.

n-part spidrons have been largely studied for n=6 and 8[5]. The present method illustrates n-part spidrons for other values of n as well.

Apart from generating nested spidrons inside a regular polygon, the paper also explores diverse tiling techniques using these. These are primarily the distinctive features of this paper.

2. THE ALGORITHM
The algorithm is best explained in terms of Figure 2, which depicts two juxtaposed isosceles triangles, this being the basic building block of a spidron. Let the base angle ABC have value ag and let AB have length s. Then AC=CD=AB=s and angle ACB=ag. Also angle CAD=angle ACD = ag/2. From this it follows that angle CAD=angle ACD = ag/2. Also Bc=2s(cos(ag)) and AD=2s(cos(ag/2)). With this it is easy to construct the two triangles in Figure 2. It is to be noted that for n n-part spidrons to be nested inside an n-sided regular polygon, ag=360/n.

For the recursive step, one refers to Figure 3, in which the next iteration is shown. It can be easily calculated that BE = s(cos(ag)/cos(ag/2)). Accordingly s is replaced by this value before the recursive call.
Figure 3 depicts a 12-part spidron after 8 iterations.

In the next section one submits a programming code in Visual C++ that captures the algorithm and generates its output.

3. THE CODE

The code uses a function spid which is declared first. The function has two parameters – s, which gives the value of the two equal sides of the first isosceles triangle and l, which specifies the number of recursive iterations. The function draws a single n-part spidron. There is also another function, polyspidron, which draws n n-part spidrons in such a way that they are nested inside an n-sided regular polygon. This is achieved by calculating the appropriate angle between two consecutive spidrons. The spidrons are then colored using different colors thereby creating a spiral-like structure. The functions fd, rt, lt, pu and pd are adapted from turtle graphics[1] and have their usual connotations. Fd draws a line of specified length, rt and lt respectively rotates the pen right and left by a specified angle, pu puts the pen up (no drawing) and pd puts the pen down. The various functions and the code are given below:

```c
void spid(float s, int l)
{
float a, b, c = a, px, py, c = ang; float ag = 360/n;
if (l == 0) {rt(90); fd(s); lt(180-ag); fd(s);
        fd(2*s*cos(ag*3.14/180));
        lt(180-ag/2); fd(2*s*cos(ag/2*3.14/180));
        lt(180-ag/2); fd(s); rt(180-ag);
        pu(); fd(s*2*cos(ag*3.14/180)); pd();
        return;}
spid(s, 0); lt(90+ag+ag/2);
spid(s*cos(ag*3.14/180)/cos(ag/2*3.14/180), l-1);
poly(s, 12);
}
```

```c
void polyspidron(float s, int l)
{
    for(int i=0; i<n; i++)
    {
        setcolor(i); spid(s, l);
        pu(); fd(2*s*cos(180/n*3.14/180));
        lt(90+360/n+180/n);
    }
}
```

```c
int main()
{
    initwindow(1000, 800);
    setlinestyle(SOLID_LINE, 0, THICK_WIDTH);
    float s = 70; px = 380, py = 620;
    polyspidron(s, 16);
    getch();
    closegraph();
}
```
The output of the sample code is illustrated in Figure 3, a regular dodecahedron showing the spiralling nested spidrons.

Fig 5 : Output of the sample code

4. INSCRIBING A REGULAR POLYGON USING 6-PART SPIDRONS

Using $n$ Six-part spidrons one can also construct other symmetric designs that inscribe a regular polygon with $n$ sides. For this, the variable $ag$ in the function $spid$ has to be set to 360/6 or 60. Also the function polypidron has to be changed to a function polypidron1 which is defined below:

```c
void polypidron1(float s,int l)
{for(int i=0;i<n;i++)
{spid(s,5); rt(90); fd(s);
rt(90-(2*n-4)*90/n);}
}
```

Figures 6 and 7 depict the respective outputs for $n=6$ and $n=9$.

Fig. 6 : Output of the modified code for $n=6$

Fig. 7 : Output of the modified code for $n=9$

It can be easily observed from figures 6 and 7 that the new design will always inscribe an $n$-sided regular polygon.

5. TILING THE PLANE USING SPIDRONS

Both the nested spidrons of section 3 and the new inscribed polygonal designs of section 4 can be used for tiling the plane. Figures 8 and 9 depict such tiling for $n=6$ and $n=8$ respectively. To produce the output in Figure 9, the calling of the function polypidron in the portion main() should be replaced by the following segment of code.

```c
for(int i=0; i<8; i++)
for(int j=0; j<7; j++)
{px=152*i,py=152*j;
for(int i=0;i<n;i++)polypidron(20,16);}
```

Fig. 8 : Tiling with nested spidrons for $n=6$
Fig. 9: Tiling with nested spidrons for n=8

Figures 10 to 13 explore various tiling patterns created with the new inscribed polygonal designs for n=3, 4, 5 and 8. For instance, to produce the output in Figure 13, the calling of the function polyspidron in the portion main() should be replaced by the following segment of code

```java
for(int i=0; i<8; i++)
for(int j=0; j<7; j++)
{px=126*i,py=126*j;
for(int i=0;i<n;i++)polyspidron1(20,16);}
```

Here polyspidron1 is the function that has been defined in section 4.

Fig. 10: Tiling pattern using the new design for n=3

Fig. 11: Tiling pattern using the new design for n=4

Fig. 12: Tiling pattern using the new design for n=6

Fig. 13: Tiling pattern using the new design for n=8

6. CONCLUSION

This paper presents an alternate method for constructing spidrons which are used to construct diverse polygonal designs and also explores various tiling designs using the same. In subsequent studies one will further study the effect of superimposing two or more such polygonal designs on tiling patterns. One will also study aperiodic tiling patterns created with polygonal designs using an odd number of spidrons. These are the aspects that would be explored in future work.

7. ACKNOWLEDGMENTS

The author wishes to acknowledge his debt to the referee(s) for their constructive suggestions and encouragement

8. REFERENCES


