UDC 515.2

V.P. Tkachenko,
Chelombitko V.F.
Kharkiv National University of Radio Electronics, Ukraine

IMPLEMENTATION OF SOME OF THE PRINCIPLES OF DESIGN PATTERNS MEANS OPERATORS LANGUAGE MAPLE

Abstract: The principles of the construction and ornamentation among the algorithmic language Maple.

Keywords: pattern, geometric transformation, recursive construction, maple

Statement of the problem. Ornament - a pattern that consists of rhythmically ordered items. The ornament is based on special laws by using certain tools. The main tool - a geometric transformation. Therefore, the current research will be aimed at building a formalization of computer technology ornaments, implemented by means of operators algorithmic language, eg, Maple.

Analysis of recent research and publications. Geometric transformation of the plane - a vzayemnoodnoznachne display this plane over. The most important geometric transformation is the movement, ie the transformation that preserves distance and zoom [1-3]. Examples of central symmetry is parallel transfer, rotate and axial symmetry. Motif - a major element of ornament, its main component. A further generalization is building ornaments using computer technology.

The wording of Article goals. Definitions Opportunities implementation of the principles of construction activ g patterns in an environment algorithmic language.

The main part. Certainly create graphic designs for decorative wallpapers, advertising, demonstration equipment. Software for PC patterns usually based on relatively simple algorithms, which can be regarded as tests of skill programming. The main issue here is the idea of creating a decorative effect.

Problem. Develop algorithms for compiling and building ornaments on a plane among the algorithmic language Maple, based on the main principles of creating patterns.

The main part. If you move a shape without paying respect to their "center", we obtain a plane-parallel movement of the figure, when any straight line in the figure is parallel to itself. According to the "center" of the figures can be accepted any point rigidly connected with the figure.

Of course the "center" figures (xf, yf) is moved relative to the center of the pattern (xc, yc) by the specific law **xf = xc + Fx; yf = yc + Fy,** where Fx, Fy - a function of the parameters. In general, the piece can move, rotating about their "center" and deformed. The parameters of procedure the figure must include all the coordinates of the points that connect the lines. Coordinates the i-th point shapes are determined by the formulas:

xxi: = xf + Kxi * ((xi-xf) * cos (A) - (yi-yf) * sin (A)),

yyi: = yf + Kyi * ((yi-yf) * cos (A) + (xi-xf) * sin (A)),

where *A* - the angle of rotation with respect to shape its "center", measured on the left side of the screen coordinate system clockwise in relation to the axis *X*,

xi, yi - initial coordinates of the i-th point of the figure,

xxi, yyi - new coordinates of the i-th point of the figure,

Khi, Kyi - ratios coordinates of the i-th point of the axes *X and Y.*

Let us give an example of Maple - in building a pattern in which the realized set of laws of motion lines with respect to its "center."

```
q: = evalf (W);
m: = [[0,0], [1,0], [cos (q), sin (q)], [0,0]]:
w: = []: t: = T:
for k to N do
w: = [op (w), m]:
sm: = [m [2], m [3], m [1], m [2]]
m: = (1-t) * m + t * sm:
od:
```

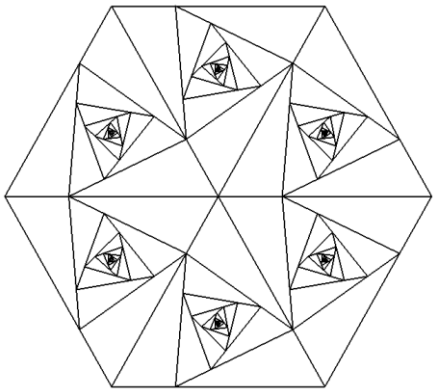
After the above preparatory unit should make six blocks that differ coefficients in pairs of trigonometric functions

```
a1: = plot (w, axes = none, scaling = constrained):
m: = [[0,0], [cos (2 * q), sin (2 * q)], [cos (1 * q), sin (1 * q)], [0,0]]:
w: = []: t: = T:
for k to N do
w: = [op (w), m]:
sm: = [m [2], m [3], m [1], m [2]]
m: = (1-t) * m + t * sm:
od:
```

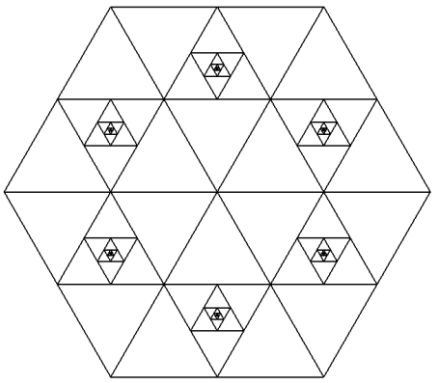
As a result of all six units will have to build a workpiece image using the operator

```
display (a1, a2, a3, a4, a5, a6, thickness = 2);
```

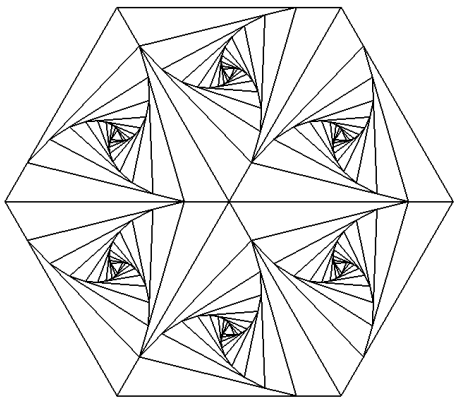
In Fig. 1 shows at **N = 10** examples of program construction pattern depending on the parameters **W** and **t.**



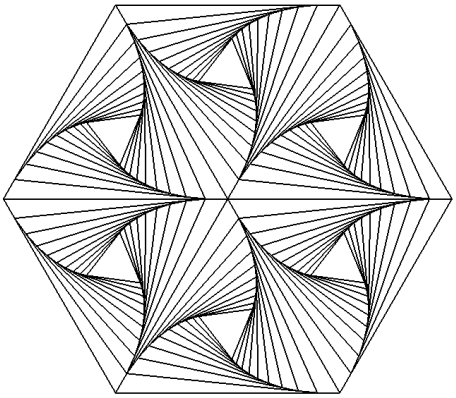
T = 0.3; W = Pi/3



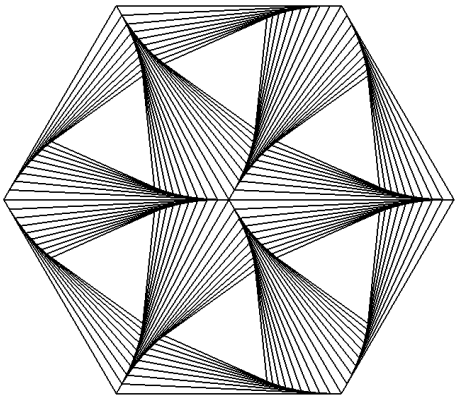
T = 0.5; W = Pi/3



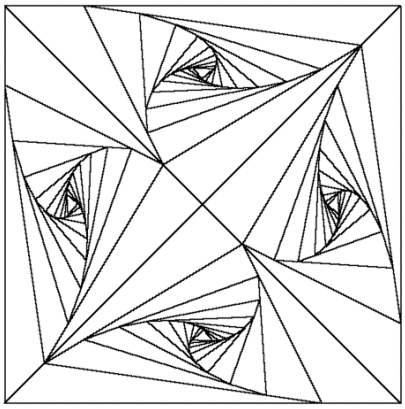
T = 0.8; W = Pi/3



T = 0.9; W = Pi/3



T = 0.95; W = Pi/3



T = 0.2; W = Pi/2

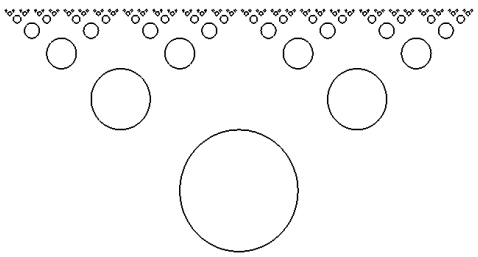
*Rice. 1. Examples of program construction pattern depending on the parameters **T** and **W.***

In programming often use recursive operators, such **procedures include reference to themselves.** Such an appeal may be direct - that is the challenge procedure within the procedure or indirect - call other procedures, within which is a challenge to the original procedure.

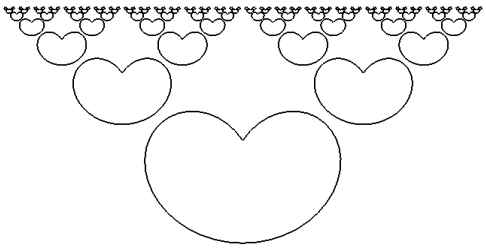
Here is an example of recursive program construction pattern where **drevo** procedure calls to the same procedure **drevo.**

```
drevo: = proc (L, N, x0, y0)
local i; global s, p;
options remember;
s: = s +1;
p [s]: = plot ([x0 + L * sqrt (M) * sin (M * t) * cos (t),
y0 + L * sqrt (M) * sin (M * t) * sin (t), t =-W .. W]):
if N> 1 then drevo (L / 2, N-1, x0-L, y0 + L);
drevo (L / 2, N-1, x0 + L, y0 + L) fi;
RETURN (display ([seq (p [i], i = 1 .. 2 ^ N-1)], axes = NONE,
scaling = constrained, numpoints = 1000, thickness = 2))
end:
s: = 0: drevo (100, 7, 0, 0);
```

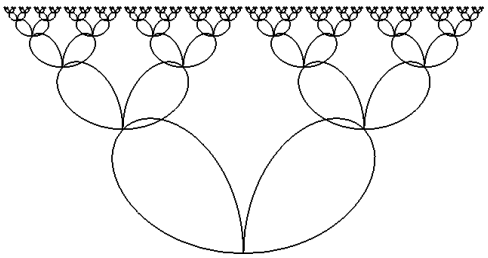
In Fig. 2 shows examples of recursive program construction pattern depending on the parameters **W** and **M.**



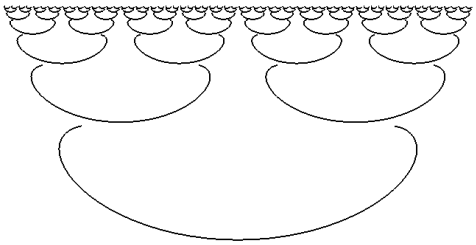
W = Pi/2; M = 1



W = Pi/2; M = 1,5



W = Pi / 2; M = 2



W = Pi / 5; M = 3

*Rice. 2. Examples of recursive program construction pattern depending on the parameters **W** and **M.***

It is interesting to create building designs mirror images of the figure. In this case, the principle of building designs similar to the image mirrored kaleidoscope. In a kaleidoscope of three mirror system creates the effect of multiple reflections sixfold set of colored crystals. Mathematically, this principle of construction of a pattern can be described as follows. There are three output beams from one point - the axes of symmetry. The angle between the beams is 2 * π / 3. We construct the first (initial) figure in the sector between the two beams. Then the second figure is constructed as a mirror image of the first figure relative to the second beam, then the third figure, like a mirror image of the other shapes of the beam with respect to the third and so on.

Here is an example program to create a pattern construction reflections on three primary figure axis reflection.

fosi1: = x * sin (Pi / 3) - y * cos (Pi / 3);
fosi2: = y;
fosi3: = x * sin (Pi / 3) + y * cos (Pi / 3)

where the angle measured from the axis Ox counterclockwise.

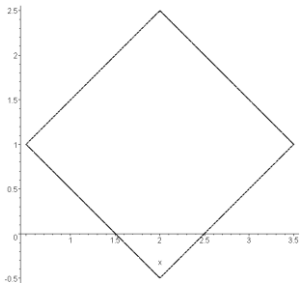
The initial shape (eg, square) specifies the equation in the form

f: = (x, y) -> a - abs (x-x0) - abs (y-y0):

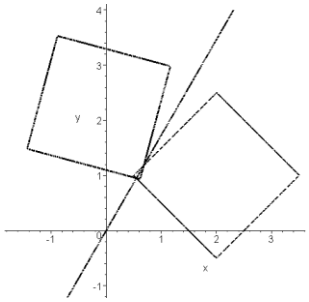
To construct the result of reflection symmetric with respect to the first, second and third axis of symmetry formula should be used:

X1: = (x, y) -> y * sin (2 * Pi / 3) + x * cos (2 * Pi / 3);
Y1: = (x, y) -> x * sin (2 * Pi / 3) - y * cos (2 * Pi / 3);
f1: = (x, y) -> f (X1, Y1);
X2: = x;
Y2: =-y;
f2: = (x, y) -> f1 (X2, Y2);
X3: = (x, y) -> y * sin (-2 * Pi / 3) + x * cos (-2 * Pi / 3);

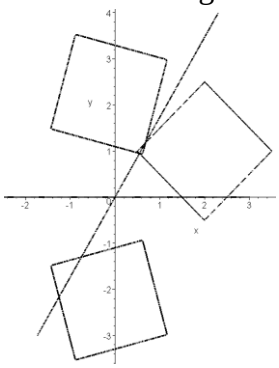
Y3: = (x, y) -> x * sin (-2 * Pi / 3) - y * cos (-2 * Pi / 3);
f3: = (x, y) -> f2 (X3, Y3);
Constructs when **a = 1,5; x0 = 2; y0 = 1** milestones reflection operators carry
F: = implicitplot (f (x, y), x = -3 .. 4, y = -3 .. 4);
F1: = implicitplot (f1 (x, y), x = -3 .. 4, y = -3 .. 4);
F2: = implicitplot (f2 (x, y), x = -3 .. 4, y = -3 .. 4);
F3: = implicitplot (f3 (x, y), x = -3 .. 4, y = -3 .. 4);
The resulting image (which goes on in the cycle) is constructed by using the
display (F, F1, F2, F3, thickness = 3);
In Fig. 3 shows examples of constructed images.



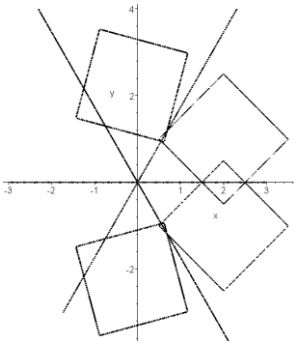
a) n ochatkove image



b) The results of the first reflection region



c) The result of the second reflection



d) the result of the third reflection

Rice. 3. Building a symmetrical mirror image kaleidoscope

Conclusions. G activ principles of the patterns can be realized in the environment of the algorithmic language by creating algorithms for image plane.

Prospects for further research. Further study the principles of design patterns.

Literature

1. Классификация орнаментов – http://TMN.FIO.ru/works/80x/311/classif.htm
2. http:// graphfunk.narod.ru/ «Графики функций».
3. Котов Ю.В. Как рисует машина. – М.: Наука. 1988. – 224 с.

Аннотация

Ткаченко В.Ф., Челомбитько В.Ф. Реализация некоторых принципов построения узоров средствами операторов языка MAPLE. Рассмотрены принципы составления и построения орнаментов в среде алгоритмического языка Maple.

Ключевые слова: орнамент, геометрические преобразования,

рекурсивные построения, MAPLE.

Анотація

Ткаченко В.Ф., Челомбітько В.Ф. Реалізація деяких принципів побудови візерунків засобами операторів мови MAPLE. Розглянуто принципи складання та побудови орнаментів в середовищі алгоритмічної мови Maple.

Ключові слова: орнамент, геометричні перетворення, рекурсивні побудови, MAPLE.