Для того, чтобы нарисовать множество многоугольников, различных по цвету на языке Pascal ABC.Net, необходимо подключить два модуля, позволяющих работать с экраном и графикой:

**uses** ABCObjects,GraphABC;

**type**

pointR = **record**

x,y : real;

**end**;

//координаты в рациональных числах

pointS = **record**

x,y : word;

**end**;

//координаты в натуральных числах

screenCalc = **record**

kx,ky : real;

x0,y0 : word;

xm0,ym0 : real;

**end**;

//вспомогательный тип для перевода координат обычных в координаты экранные

PList = ^PointList;

PointList = **record**

point : PointR;

next : PList;

**end**;

//список для хранения координат

//Константа, в которой задается количество точек-координат фигуры:

**const**

StarVertex = 10;

//Необходимые переменные:

**var**

c: Color;

star, test : PList;

i : word;

start1,start2 : pointR;

screen : screenCalc;

p : Picture;

//Необходимые процедуры:

// Расчет коэффициентов преобразования в экранные координаты

**function** calcScreen( x0,y0, x1,y1 : word; xm0,ym0,xm1,ym1 : real

) :screenCalc;

**begin**

Result.kx := (x1-x0)/(xm1-xm0);

Result.ky := (y1-y0)/(ym1-ym0);

Result.x0 := x0;

Result.y0 := y0;

Result.xm0 := xm0;

Result.ym0 := ym0;

**end**;

// Перевод математических координат в экранные

**function** toScreen( math : pointR; screen : screenCalc ) :

pointS ;

**var**

res : pointS;

**begin**

res.x := round((math.x-screen.xm0)\*screen.kx+screen.x0);

res.y := round((math.y-screen.ym0)\*screen.ky+screen.y0);

toScreen := res;

**end**;

// Поворот точки на угол (в радианах)

**function** Rotate( math : pointR; angle : real ) : PointR ;

**var**

res : pointR;

**begin**

res.x := math.x\*cos(angle)- math.y\*sin(angle);

res.y := math.x \* sin(angle) + math.y \* cos(angle);

rotate := res;

**end**;

// Масштабирование относительно центра координат для следующей звезды

**function** Scale( math : pointR; k : real ) : PointR ;

**var**

res : pointR;

**begin**

res.x := math.x\*k;

res.y := math.y\*k;

Scale := res;

**end**;

// добавление следующей точки с ее координатами в список координат фигуры

**function** addPoint( coord : pointR; next : PList ) : PList;

**begin**

**new** (Result);

Result^.point := coord;

Result^.next := next;

**end**;

//Процедура построения линий от точки до точки

**procedure** Polyline( p : Picture; vertex : PList; screen

:screenCalc; c: Color );

**var**

test : PList;

p1,p2 : pointS;

**begin**

p1 := toScreen(vertex^.point,screen );//Первая точка списка

test := vertex^.next; //Со второй точки

**while** test <> nil **do** // Пройдем по всему списку

**begin**

p2 := p1;

p1 := toScreen(test^.point,screen );

p.Line(p2.x,p2.y,p1.x,p1.y,c);

test := test^.next;

**end**;

p2 := toScreen(vertex^.point,screen );

p.Line(p2.x,p2.y,p1.x,p1.y,c); // Замкнем ломаную

**end**;

//Тело программы:

**Begin**

//С самого начала зададим координаты двух первых точек

start1.x := 9;

start1.y := 2;

start2.x := 9;

start2.y := 2;

star := addPoint(start1,nil);//добавление первой точки в список координат фигуры

star := addPoint(Rotate(start2,2\*Pi/StarVertex),star);//добавление следующей за первой точки с учетом коэффициента поворота

//Далее с третьей точки и до определенного константой количества точек-координат они будут добавляться в динамический список циклом с учетом поворота.

**for** i:=3 **to** StarVertex **do**

**if** i **mod** 2 = 1 **then**

star := addPoint(Rotate(start1,(2\*Pi/StarVertex)\*(i-1)),star)

**else**

star := addPoint(Rotate(start2,(2\*Pi/StarVertex)\*(i-1)),star);

screen := calcScreen(0,0,500,500,-36,-36,36,36);

p := **new** Picture( 500,500);

//Цикл, который будет каждый раз рисовать новую, большую, чем предыдущая, фигуру.

**for** i:=1 **to** 14 **do**

**begin**

c:=clRandom; // для того, чтобы каждая следующая фигура получалась другого цвета, применим процедуру из модуля работы с графикой, возвращающую случайный цвет. Таким образом, не нужно будет каждый раз возвращать случайные три переменные для определения цвета в RGB кодировке, тем более что такой способ в Pascal ABC.Net работает с перебоями.

PolyLine(p,star,screen,c);

test := star; // Пройдем по всему списку для того, чтобы отмасштабировать следующую за нынешней звезду

**while** test <> nil **do**

**begin**

test^.point := Scale(test^.point,1.1 );

test := test^.next;

**end**;

**end**;

//Выводим полученный результат на экран:

p.Draw(0,0);

**while** true **do**

;

**end**.