Смольянов Андрей Григорьевич
Зав. кафедрой фундаментальной информатики Мордовского государственного университета им. Н.П. Огарёва, г. Саранск.

Программирование и рисование круговых диаграмм

В статье обсуждается вопрос применения стандартных графических подпрограмм при решении практических задач компьютерной графики. Рассматриваются теория и практика, лежащие в основе написания достаточно универсальной подпрограммы пользователя для рисования круговых диаграмм.

Графический экран и системы координат

Многие системы разработки программ, включающие средства программирования компьютерной графики, предлагают широкий набор встроенных процедур, значительно облегчающих разработку приложений. К примеру, в семействе систем Delphi одной из таких процедур является Pie, которая предназначена для рисования сектора круга. Её заголовок выглядит так:

```
procedure Pie(X1: Integer; Y1: Integer;
                    X2: Integer; Y2: Integer;
                    X3: Integer; Y3: Integer;
                    X4: Integer; Y4: Integer);
```

Координаты (X_1,Y_1) и (X_2,Y_2) задают прямоугольную область, в которую вписан эллипс. Роль точек (X_3,Y_3) и (X_4,Y_4) показана на рисунке 1. Сектор ограничен отрезками лучей, выходящих из центра эллипса и проходящих через точки $A(X_3,Y_3)$ и $B(X_4,Y_4)$. Однако начинающему пользователю стандартных графических подпрограмм не совсем понятно, как всё же практически нарисовать нужные «куски пирога», работая с реальными данными?

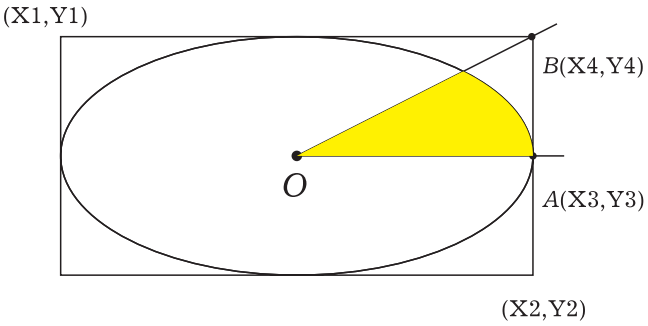

Рис. 1

Попытаемся изобразить сектор, показанный на рисунке 1, с помощью программы. Её фрагмент выглядит просто:

```
with Image1,Canvas do
  begin
    Pen.Color:=clNavy; Brush.Color:=clWhite;
    Rectangle(0,0,width,height);
    Brush.Color:=clYellow;
    Pie(1,1,width-1,height-1,width-1,
        height div 2,width-1,1)
  end;
```

Результат рисования совпал с ожидаемым (рис. 2):

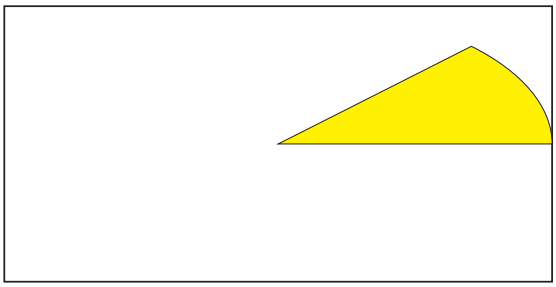

Рис. 2

Однако этот успех совершенно не проясняет технику рисования реальных круговых диаграмм. Картина станет более ясной, если обратиться к известной единичной окружности (рис. 3).

Будем рисовать сектор от луча *OA* к лучу *OB*, двигаясь против часовой стрелки. Заметим, что точка *B*(X4,Y4) луча *OB* выбрана иначе, а именно, – на единичной окружности.

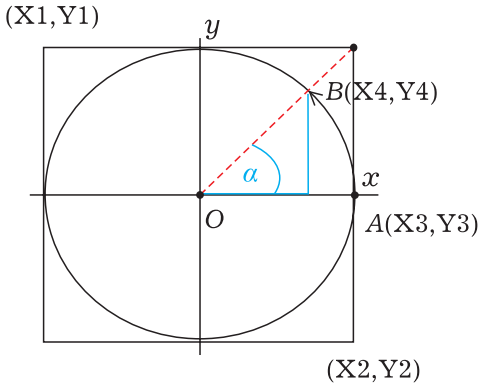

Рис. 3

Из рисунка 3 следует, что абсцисса X4 и ордината Y4 новой точки *B* вычисляются обычным образом: X4 = $= \cos(\alpha)$, $Y4 = \sin(\alpha)$. Следующая проблема: рисование на графическом экране. Напомним, что система координат графического экрана выглядит так, как показано на рисунке 4.

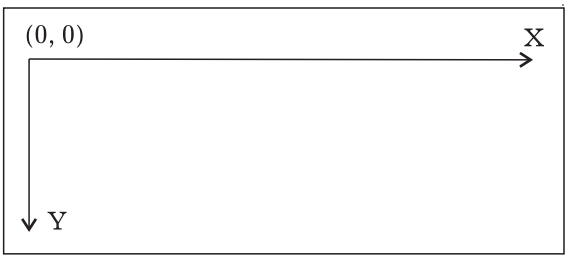

Рис. 4

Рассматривая точку (X*,Y*) в декартовой системе координат, мы должны правильно отобразить её на графическом экране (рис. 5). Обсудим данный вопрос на примере рисования графика функции.

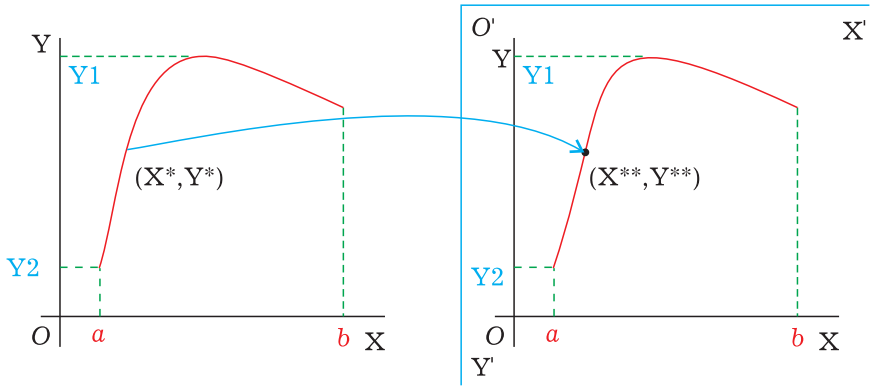

Рис. 5

Пусть XOY – исходная декартова система координат (рис. 5, слева), а X'O'Y' – система координат графического экрана (рис. 5, справа). Пусть в случае с функцией в системе XOY Y_1 – максимальное значение функции f(x) на отрезке [X1,X2], а Y_2 – минимальное значение функции f(x) на этом отрезке. В системе координат X'O'Y' для рисования графика функции используем часть прямоугольной области $[I_1,I_2]x[J_1,J_2]$, где $[I_1,I_2]$ – интервал по оси O'X', $[J_1,J_2]$ – интервал по оси O'Y'. Тогда для расчёта абсциссы X* из XOY в системе X'O'Y' необходимо проделать следующие вычисления:

$$X^{**} = I_1 + Trunc((I_2-I_1)*(X^*-X_1)/(X_2-X_1)).$$

Аналогичное выражение получаем для ординаты Y* из XOY в системе X'O'Y':

$$Y^{**} = J_1 + Trunc((J_2-J_1)*(Y^*-Y_1)/(Y_2-Y_1)).$$

Запишем функции для проведения указанных вычислений:

```
function ScrX(x: real): integer;
  begin
     ScrX := I1+Trunc((x-x1)*(I2-I1)/(X2-X1))
  end;
function ScrY(y: real): integer;
  begin
     ScrY := J1+Trunc((y-Y1)*(J2-J1)/(Y2-Y1))
  end;
```

Кисти, краски и холст: рисуем диаграмму

Покажем работу рассмотренного инструментария на простом примере.

Пусть требуется представить в виде круговой диаграммы процентное соотношение выпущенной за день продукции кондитерской фабрики (в килограммах). Используем комбинированный тип данных ValType:

```
type ValType=record Val: real; //показатель
     Name: string;     //наименование
     MyColor: TColor; //цвет сектора
                  end;
```

Назначение следующих переменных понятно.

```
Var I1,J1,I2,J2: integer;
    x1,x2,y1,y2: real;
    Values: array of ValType;
```

В программе используем наши функции ScrX() и ScrY().

Зададим значения для параметров задачи:

```
      // Отрезки в системе координат X'O'Y':
      with Image1 do //область рисования
        begin
          I1:=10; I2:=width-10;
          J1:=10; J2:=height-10;
        end;
      // Отрезки в системе координат XOY:
      x1:=-1; x2:= 1;
      y1:= 1; y2:=-1;
```

Зададим исходные данные:

```
    nCount:=5; SetLength(Values,nCount);
    Values[0].val:=250;
    Values[0].name:='Пряники мятные';
    Values[1].val:=510;
    Values[1].name:='Печенье "К чаю"';
    Values[2].val:=670;
    Values[2].name:='Зефир в шоколаде';
    Values[3].val:=120;
    Values[3].name:='Пастила ванильная';
    Values[4].val:=320;
    Values[4].name:='Хлебцы овсяные';
    // Цвета секторов сформируем случайным образом:
    randomize;
    for i := 0 to nCount - 1 do
        Values[i].MyColor:=RGB(random(240)+10,
            random(240)+10,random(240)+10);
```

Собственно рисование реализует подпрограмма `Sector`. Её вызов имеет вид: `Sector(Values,nCount);`

Опишем эту подпрограмму:

```
      procedure     Sector(Values:array     of     ValType;
nCount:integer);
      var Ax,Ay,Bx,By:real;
          alpha:real;
          i,y_start:integer;
          sum_val:real;
          WorkValues:array of real;
          WorkNCount:integer;
      begin
        WorkNCount:=0; SetLength(WorkValues,WorkNCount);
        WorkNCount:=nCount;
SetLength(WorkValues,WorkNCount);
        // Image1 - область рисования диаграммы
        with Form1,Image1,Canvas do
          begin
```

```
            Brush.Color:=clCream; Pen.Color:=clNavy;
            Rectangle(0,0,width-1,height-1)
          end;
        // Image2 - область вывода легенды
        with Form1.Image2 do
          begin
            Height:=20*nCount+10;
            Width:=Form1.ScrollBox1.width-24;
            Top:=2; Left:=2;
            with Canvas do
              begin
                Pen.Color:=clNavy;
                Brush.Color:=clCream;
                Rectangle(0,0,width-1,height-1);
                Font.Name:='Comic Sans MS'; Font.Size:=10;
                Font.Color:=clNavy
              end
          end;
        sum_val:=0;
        for i := 0 to nCount - 1 do
             sum_val:=sum_val+Values[i].Val;
        if sum_val = 0 then
           begin
             ShowMessage
                ('Нет данных для построения диаграммы...');
             Exit
           end;
        // От абсолютных величин Values[i].Val
        // переходим к относительным величинам:
        // 1) к долям:
        for i := 0 to nCount - 1 do
            WorkValues[i]:=Values[i].Val/sum_val;
        // 2) к градусам и далее - к радианам:
        for i := 0 to nCount - 1 do
            WorkValues[i]:= (WorkValues[i] * 360.0) / 180.0
* pi;
        Bx:=x2; By:=0; //инициализация
                       //для начальной точки
        alpha:=0;
        for i := 0 to nCount - 1 do
          begin
            //углы накапливаем:
            alpha:=alpha+WorkValues[i];
            // (Ax,Ay) - начальная точка:
            Ax:=Bx; Ay:=By;
            // (Bx,By) - конечная точка:
            Bx:= cos(alpha);
            By:= sin(alpha);
            if Values[i].Val > 1E-5 then // Val > 0 ?
```

```
        with Form1,Image1,Canvas do
          begin
            Brush.Style:=bsSolid;
            Brush.Color:=Values[i].MyColor;
            Pie(ScrX(x1),ScrY(y1),ScrX(x2),ScrY(y2),
                ScrX(Ax),ScrY(Ay),ScrX(Bx),ScrY(By))
          end
      end;
    with Form1,Image1,Canvas do
      begin
        Brush.Color:=clCream; Pen.Color:=clNavy;
        Ellipse(ScrX((x1+x2)/2)-(I2-I1)div 10,
                ScrY((y1+y2)/2)-(J2-J1)div 10,
                ScrX((x1+x2)/2)+(I2-I1)div 10,
                ScrY((y1+y2)/2)+(J2-J1)div 10)
      end;
    with Form1.Image2 do
      begin
        y_start:=Top+2;
        with Canvas do
          begin
            Pen.Color:=clNavy;
            for i := 0 to nCount - 1 do
              begin
                Brush.Color:=Values[i].MyColor;
  rectangle(left+2,y_start,left+22,y_start+15);
                Brush.Color:=clCream;
                textout(left+27,y_start-2,
                        '- '+Values[i].Name+' '+
                FloatToStrF(Values[i].val/sum_val*100,
                        ffFixed,5,1)+'%');
                //высота строки легенды - 20 пикселей:
                y_start:=y_start+20
              end
          end
      end
  end;
```

После вызова подпрограммы `Sector` освобождаем память:

```
nCount:=0;
SetLength(Values,nCount);
```

На рисунке 6 показан результат работы программы.

Заметим, что он теперь не зависит ни от параметров прямоугольной области, ни от её расположения на графическом экране (рис. 7, 8).

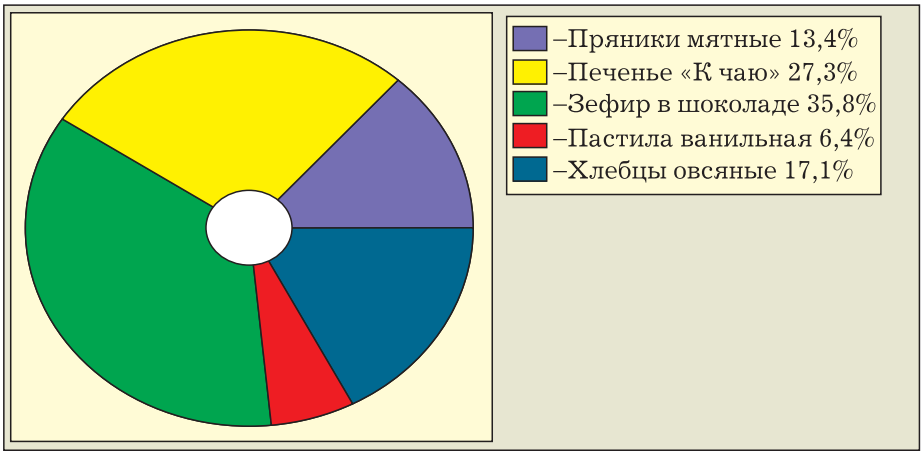

Рис. 6

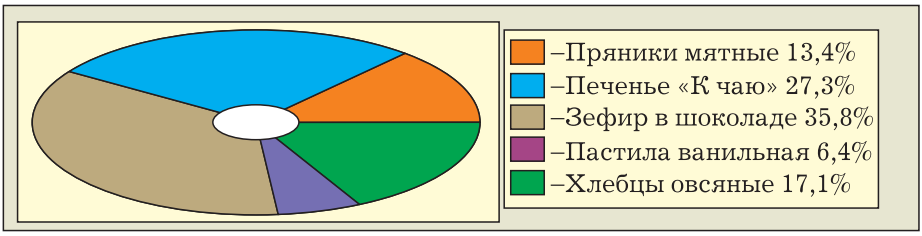

Рис. 7

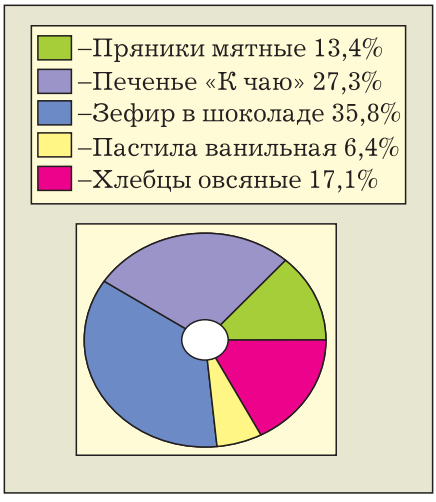

Рис. 8