1. Особенности графики

Начиная с версии 4.0, в состав Паскаля включена мощная библиотека графических подпрограмм Graph, остающаяся практически неизменной во всех последующих версиях. Библиотека содержит в общей сложности более 50 процедур и функций, предоставляющих программисту самые разнообразные возможности управления графическим экраном. Для облегчения знакомства с библиотекой все входящие в нее процедуры и функции сгруппированы по функциональному принципу.

1.1 Переход в графический режим и возврат в текстовый

Стандартное состояние ПК после его включения, а также к моменту запуска программы из среды Паскаля соответствует работе экрана в текстовом режиме, поэтому любая программа, использующая графические средства компьютера, должна определенным образом инициировать графический режим работы дисплейного адаптера. После завершения работы программы ПК возвращается в текстовый режим.

Для работы в графическом режиме системы -Паскаль 5.0, 6.0, 7.0 имеется специально подключаемый программный модуль ***graph*.** Это подключение описывается в начале программы после строки *program* за служебным словом *uses*. Но при написании программ очень часто необходимо пользоваться специальными функциями обработки клавиатуры и управления дисплеем, которые находятся в модуле *crt*, поэтому обычно подключают *graph* и *crt*. Отметим, что система координат в графическом режиме не соответствует системе координат текстового режима. Текстовый режим обеспечивает, как правило, выдачу символов в 25 строк по 80 символов в строке. Графические же режимы обеспечивают выдачу точек (пикселей), различную для различных средств (для различных типов мониторов). 640 200 – машины класса CGA (файл cga.bgi), 640 350 EGA, 640 480 VGA.(файл egavga.bgi) Есть и другие размеры. Работа программы с графикой начинается с инициализации графического режима, которая обеспечивается записью в основной программе команды *initgraph* (*grd, grm, 'путь')*, где *initgraph* – служебное слово (команда вывода процедуры); *grd* и *grm* – имена переменных целого типа. Эти переменные должны быть описаны в разделе *var*. Значения переменной *grd* задает драйвер (специальная программа системы, обеспечивающая управление различными периферийными устройствами) в соответствии типу дисплея. *grm* – значение режима для работы этого дисплея. Эти значения могут задаваться как по именам, так и цифрами, т. е. можно записать: *grd:=VGA* или *grd:=9*, – эти записи равносильны. Значения присваиваются до инициализации. Для тех, кто не знает тип дисплея, используются специальные команда *detect (*константа), которая автоматически определяет тип дисплея. Поэтому мы будем писать: *grd:=detect* или *grd:=0.* После выполнения этой команды присваивания автоматически вызывается нужный драйвер и устанавливается наиболее подходящий режим для дисплея. После загрузки драйвера происходит его настройка на режим, заданный переменной *grd.* В апострофах указывается путь по каталогам, где хранится подключаемый драйвер. У нас он хранится на диске O:\bp70\bg*i, но каталог bgi необходимо скопировать на в папку c:\work , grd* – какие процедуры для данного типа дисплея, *grm* – разрешимость.

1.2 Начало и окончание программы для работы с графикой:

```
program vv;
  uses graph, crt;
  var grd, grm:integer;
BEGIN
  grd:=detect;
  initgraph (grd, grm, 'c:\work\bgi' ); (вызов процедуры)
                    РИСУНОК
readln;  {ожидание нажатия клавиши ввода, чтобы видеть рисунок}
closegraph; { закрыть графический режим }
END.
```

После инициализации графического режима можно выполнять графические построения. Текст также можно выводить, но буквы рисуются . В конце работы с графическим режимом необходимо обязательно выйти из него с помощью команды *closegraph.*

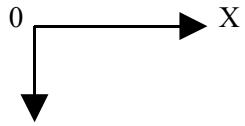
Установка и управление цветом и фоном в -Паскале.

0	черный
1	синий
2	зеленый
3	голубой
4	красный
5	лиловый
6	коричневый
7	серый
8	темно-серый
9	светло-синий
10	светло-зеленый
11	светло-голубой
12	светло-красный
13	светло-фиолетовый
14	желтый
15	белый

Кроме того, можно специальными командами изменять палитру. *setcolor (c)* – текущий цвет изображения, используя заданную палитру, где с – конкретное число или имя переменной, которая должна быть описана в разделе *var* типом *byte*, *integer*, *word (0,65.535)* – целые числа.

setbkcolor (c) – устанавливает текущий цвет фона.

Управление текущим курсором (указателем). Текущий указатель и конец рисования линии – это разные вещи. Система координат стандартная (начало – левый верхний угол экрана):



Условимся, что переменные x и y в дальнейших обозначениях имеют значения целого типа.

getmaxx – значение, которое может максимально принимать x для данного типа дисплея. Если $600 \quad 200$, то max *x* = 599; *getmaxy* – max *y*=199. У нас экраны 640x480.

getx – определяет текущее положение курсора по оси *X*.

get y – то же самое по оси *Y*.

1.3 Рисование контурных геометрических изображений.

Для этих изображений цвет фона и текущий цвет (setcolor – изменяется только цвет «арандаша») устанавливаются описанными выше средствами.

putpixel *(x,y,c)* – установить точку с координатами x, y и цветом c.

line (*x1,y1,x2,y2)* – нарисовать линию, соединяющую точки *(x1, y1)* и *(x2,y2)* текущим цветом.

circle *(x,y,z)* – окружность с центром (*x*,*y*) радиусом *r*.

rectangle *(x1,y1,x2,y2)* – контур прямоугольника с диагональю *(x1, y1, x2, y2)*.

ellipse *(*x, y, n_u, k_u, x_r, y_r*)* – эллипс (или дуга), x, y – координаты центра, n_u, k_u – начальный и конечный угол в градусах, x_r, y_r – горизонтальная и вертикальная полуоси.

arc *(x,y,* n_u, k_u*,r)* – дуга окружности.

1.4 Рисование закрашенных изображений.

При закраске таких изображений команда *setcolor* не действует. Для рисования и закрашивания используется специальная команда цвета со своим шаблоном и номера цвета: ***setfillstyle*** *(sh,c1)* (тип и цвет «кисти).

sh – шаблон закраски, число от 0 до 12; *c1* – номер цвета закраски.

0 – область заполняется цветом фона; 1 – сплошная закраска цветом *c1*; 2 – шаблон закраски линиями; 4 – толстыми линиями; …… . 9 – заполнение клеткой; 10 – редкими точками; 11 – частыми точками.

Все шаблоны цветом *c1*.

Закрашенный прямоугольник ***bar*** *(x1,y1,x2,y2).*

Пример: ***setfillstyle*** *(11,12); bar (10,20,100,200).*

Параллелепипед ***bar3d*** *(x1,y1,x2,y2,d,e)*, где *x1*, *y1*, *x2*, *y2* – координаты диагонали передней грани, *d* – длина изображения; *e* – значение булевого типа. Если e истинно (*true*), то рисуется верхняя грань; если *e false* – не рисуется.

После выполнения этой команды на экране появляется соответствующий параллелепипед с закрашенной передней гранью.

Пример: *setfillstyle (1,12); bar 3d (50,70,150,200,40,true);*.

Закрашенный эллипс (круг) ***fillellipse*** *(x,y,* x_r *,* y_r *)*;.

Сектор круга ***pieslice*** *(x,y,* n_u *,* k_u *,r)*; сам контур сектора рисуется текущим цветом.

sector *(x,y,* n_u *,* k_u *,* x_r *,* y_r *)* – сектор эллипса закрасится тем цветом, который в шаблоне, контуры будут нарисованы текущим цветом.

Закрашивание (заливка) замкнутых областей

floodfill *(x,y,g)* – закрашивает замкнутую область текущим цветом закраски, который установлен в команде *setfillstyle*. Здесь *x*, *y* – координаты любой точки внутри замкнутой области, а *g* – цвет границы этой замкнутой области.

setcolor(4); line (0,0,100,0); line (100,0,0,50); line (0,50,0,0) – треугольник красного цвета (4);

setfillstyle(1,2); floodfill (20,10,4) – закраска треугольника зеленым цветом (2).

1.5 Вывод текста на графический экран

Очистка графического экрана: *cleardevise*. Команды *readln(n), writeln(n)* действуют только в текстовом экране. Для вывода текста на экран используются различные шаблоны. Для вывода текста используются две команды:

outtext*(s)*; *s* – переменная или константа строкового типа. Эта команда вызывает вывод строки *s*, начиная с позиции текущего курсора текущим цветом. После вывода текста текущий указатель смещается в его конец.

outtextxy*(x1,y1,s); x1,y1* – координата начала вывода текста, *s* – переменная или константа строкового типа.

1.6 Очитска экрана, тип и толщина линии

cleardevice – очистка графического экрана

setlinestyle *(а,к,в)* – устанавливает текущую толщину и тип линии, где *а*, *к*, *в* – величины типа типа *word*. Величина а задает тип линии (0 – сплошная, 1 – пунктирная, 2 – штрих-пунктирная, 3 – штриховая, 4 – стиль специальными командами создается программистом).

Пример. Рисование 30 разноцветных линий, выходящих из одной точки разными стилями. (Так как всех номеров цветов 16 (от 0 до 15), то во всех случаях, где номер цвета больше 15 берется остаток от деления этого числа на 16).

```
program ris;
uses crt,graph;
var dr,dm,x,y,i:integer;
BEGIN
  dr:=detect;
  initgraph(dr,dm,'o:\bp70\bgi');
  for i:=1 to 30 do
      begin
          setcolor(i);
          setlinestyle(4,60*i,3);
          line(0,0,20*i,300)
```

```
    end;
  readln;
  closegraph;END.
```

Вывод чисел на графический экран.

Для вывода чисел необходимо использовать процедуру перевода числа в строковую переменную *str (i,s)*, и команды вывода текста на графический экран *outtext(s)* (*s* – переменная или константа строкового типа выводится с текущей позиции курсора); *outtextxy(x1,y1,s);* (*x1*, *y1* – координата начала вывода текста, *s* – переменная или константа строкового типа).

Пример:

```
var i:integer; s:string;
.....
  i:=5;
  str(i,s);
  outtextxy(100,30,s);.
```

Шрифт

settextstyle(a,b,p) – команда установки шрифта для вывода в графическом режиме. Здесь *a* – задает шрифт (целые числа от 0 до 11), *b* – задает ориентацию (0 –горизонтально, 1 – вертикально), *p* – размер шрифта)

```
program kkk;
uses graph;
var dr,dm, i:integer;
BEGIN
  dr:=detect;
  initgraph(dr,dm,'O:\BP70\BGI');
  setcolor(14);
  for i:=0 to 11 do
      begin
          settextstyle(i,0,5);
          outtextxy(20,40*i,'PRIVET');
      end;
  readln;
END.
```

setusercharsize (шх,кх,шу,ку) – еще один способ установки величины шрифта. Делением *шх* на *кх* задается ширина шрифта, а делением *шу* на *ку* задается высота.

Пример. Вывод слова PRIVET разной ширины 5 раз и разной высоты 5 раз.

```
program kkk1;
uses graph;
var dr,dm,i:integer;
BEGIN
  dr:=detect;  initgraph(dr,dm,'O:\BP70\BGI');
  setcolor(14);
  settextstyle(1,0,0);
  for i:=1 to 5 do
      begin
          setusercharsize(5,i,1,1);
          outtextxy(20,50*(i-1),'PRIVET');
          readln;
      end;
  cleardevice;
  outtextxy(0,0,'PRIVET');
  for i:=1 to 5 do
      begin
          setusercharsize(1,1,10,i);
          outtextxy(100*(i-1), 50,'PRIVET');
          readln;
```

```
    end;
  readln;
END.
```

1.7 Смещение в точки и смещение на вектор

moveto (x,y) – перенести текущий указатель в точку (x, *y*).(Сместиться в точку карандаш не рисует след)

moverel (dx,dy) – переместиться в точку с координатами *x+dx*, *y+dy*. (Сместиться на вектор карандаш не рисует след)

lineto (x,y) – соединяет точку текущего указателя с точкой (*x*, *y*). (Сместиться в точку карандаш рисует след)

linerel (dx,dy) – линия, соединяющая точку (*x*, *y*) с текущим указателем и точку с координатами (*x+dx, y+dy*). (Сместиться на вектор карандаш рисует след)

Пример. Нарисовать треугольник с помощью команд смещения.

1) в точку.

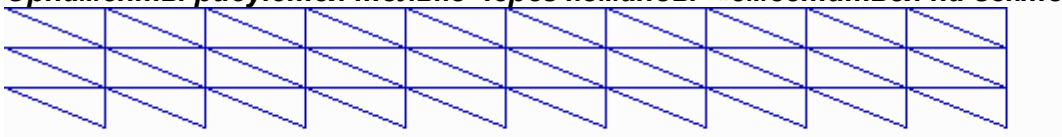
```
program ristreug;
  uses crt,graph;
  var dr, dm:integer;
BEGIN
dr:=detect;
initgraph(dr,dm,'o:/bp70/bgi');
moveto(100,100); lineto(150,100); lineto(150,120); lineto(100,100);
readln; closegraph;
END.
```

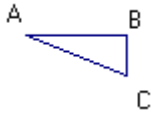
2) на вектор (тотже рисуно, что и в п.1)

```
program ristreug;
  uses crt,graph;
  var dr, dm:integer;
BEGIN
dr:=detect;
initgraph(dr,dm,'o:/bp70/bgi');
moveto(100,100);
linerel(50,0); linerel(0,20); linerel(-50,-20);
readln;
closegraph;
END.
```

3) нарисовать орнамент из треугольников (3 строки по 10 треугольников. «ширина» -50 пикселей, «высота» -20 пикселей для одного треугольника)

Орнаменты рисуются только через команды «сместиться на вектор






```
program ristreug;
  uses crt,graph;
  var dr, dm,i,j:integer;
BEGIN
dr:=detect;
initgraph(dr,dm,'e:/bp70/bgi');
moveto(0,0);
```

```
for i:=1 to 3  do
  begin
     for j:=1 to 10 do
        begin              начинаем рисовать от точки A
        linerel(50,0);      отрезок AB
        linerel(0,20);      отрезок BC
        linerel(-50,-20);  отрезока CA
        linerel(50,0);      обязательно переместитьсяв точку B
        end;
  moverel(-500,20);   переместиться в начало нового ряда
 end;
readln;
closegraph;
END.
```

Иммитация движения.

При имитации движения используется встроенная процедура delay(ms) – пауза на ms миллисекунд, зависит от скорости процессора.Например, delay (1000)– пауза приблизительно на 1 сек. Это процедура работает только при использовании модуля crt;

Для имитации движения используются команда цикла (обычно одна, даже если двигается несколько фигур). Если мы хотим, чтобы двигающаяся фигура не оставляла следа (мультипликация), то аистема команд внутри цикла иммет следующую сртуктуру «нарисовать фигуру – пауза (delay) – стереть фигуру – пауза– изменить координты для движения».

Если след на экране остается, то нет необходимости стирать изображение фигуры.

Пример. Изобразить движение точки от середины экрана до конца экрана вдоль оси X.

```
x:=320; y:=240:  (x,y – только целого типа)
while x<=640 do
  begin
     putpixel (x,y,15);
     delay(300);
     x:=x+5;
  end;
```

Пример. Изобразить движение точки от середины экрана до конца экрана вдоль побочной диагонали вверх.

```
x:=320; y:=240:  (x,y – только целого типа)
while x<=640 do
  begin
     putpixel (x,y,15);
     delay(300);
     x:=x+5;
      y;=y-5
  end;
```

Пример. Изобразить движение отрезка без следа от начала экрана до конца экрана Оси X через центр.

```
x:=320; y:=240:  (x,y – только целого типа)
while x<=640 do
  begin
     setcolor(14);     line (x,y,x+5,y);    - нарисовать
     delay(300);                            - пауза
     setcolor(0);       line (x,y,x+5,y);   - стереть
     delay(300);
x:=x+15;                                    - изменить шаг движения
    end;
```

Элементы мультипликации. Рассмотрим эти вопросы на примере следующих программ.

Пример 1. Движение эллипса.

```
program ellip;
uses crt,graph;
var dr,dm,x,y:integer;
BEGIN
  dr:=detect; initgraph(dr,dm,'o:\bp70\bgi');
  setcolor(2);setfillstyle(1,2);
  x:=50;y:=80;
  fillellipse(x,y,40,40);
  while x<=600 do
      begin
          setcolor(0); |
          setfillstyle(1,0); {стереть старое изображение}
          fillellipse(x,y,40,40);
          x:=x+10; {изменить координаты}
          setcolor(2);
          setfillstyle(1,2); {нарисовать новое изображение}
          fillellipse(x,y,40,40);
          delay(80); пауза
      end;
  repeat until keypressed;{ ждать нажатия какой-либо клавиши}
  closegraph;
END.
```

Пример . Человечек машет рукой.

```
uses crt,graph;
var dr,dm,i:integer;
BEGIN
  dr:=detect;
  initgraph(dr,dm,'o:\bp70\bgi');
  setcolor(2);
  setfillstyle(1,4);
  fillellipse(300,240,40,40);
  setfillstyle(1,2);
  bar(340,240,460,275);
  line(360,240,390,200);
  rectangle(460,250,500,260);
  putpixel(320,220,15);
  for i:=1 to 10 do
      begin
          setcolor(0);
          line(360,240,400,200);
          delay(500);
          setcolor(2);
          line(360,240,370,200);
          delay(500);
          setcolor(0);
          line(360,240,370,200);
          delay(500);
          setcolor(2);
          line(360,240,400,200);
          delay(500);
```

```
    end;
  readln;
  closegraph;
END.
```

Пример. Движение точки и линии 4 раза по окружности с заданным центром и радиусом.

```
program okru;
uses crt,graph;
var dr,dm,x,y,x0,y0,r:integer; f,x1,y1:real;
BEGIN
  dr:=detect;
  initgraph(dr,dm,'o:\bp70\bgi');
  x0:=300;y0:=200;r:=100;
  circle(x0,y0,2);

f:=0;
  while f<=4*pi do
     begin
        x1:=r*cos(f);
     y1:=r*sin(f);
        x:=x0+trunc(x1);          y:=y0+trunc(y1);
        putpixel(x,y,14);
        delay(100);
        putpixel(x,y,0);
        f:=f+0.1;
     end;
  f:=0;
  while f<=4*pi do
     begin
        x1:=r*cos(f);   y1:=r*sin(f);   x:=x0+trunc(x1);  y:=y0+trunc(y1);
        setcolor(14);            line(x,y,x0,y0);
        delay(100);
        setcolor(0);          line(x,y,x0,y0);
        f:=f+0.1;
     end;
  readln;
  closegraph;END.
```

Построение графиков математических функций. Для построения графиков математических функций важно знать, сколько точек экрана по оси OX и по оси OY находится в одном единичном отрезке. Будем обозначать эти величины через Mx и My. Рассмотрим построение графика функции $y = x^2$ на отрезке [-3,2] (5 единичных отрезков по оси OX). Мы знаем, что при этих условиях минимальное значение $y = 0$, а максимальное 9 (всего 9 единичных отрезков по оси OY (для красоты возьмем еще один отрезок 9+1=10)). Будем строить график на весь экран (640 480 пикселей, rx и ry).

```
program grafik;
uses crt,graph;
var dr,dm,x0,y0,x,
  y,rx,ry,i:integer;
  mx,my,x1,y1:real;s:string;
BEGIN
  dr:=detect;initgraph(dr,dm,'o:\bp70\bgi');
  rX:=640;        {Максимальная координата по X, можно использовать фукнцию getmaxx}
  ry:=480;        {Максимальная координата по X, можно использовать фукнцию getmaxy}
  mx:=rx/5;     {Масштаб по оси Ox, т.е. количество точек экрана в одном единичном отрезке }
 my:=ry/10;    { Масштаб по оси Oy }
  x0:=trunc(3*mx);  y0:=trunc(9*my);   {Координаты точки O(0,0.) – начала координат на экране}
            {Рсование оси Ox}
```

```
  line(0,y0,640,y0);line(640,y0,635,y0-5);line(640,y0,635,y0+5);outtextxy(630,y0-15,'X');
                {Рсование оси Oy}
  line(x0,0,x0,480);line(x0,0,x0-5,5);line(x0,0,x0+5,5);  outtextxy(x0+5,15,'Y');
              {Разметка оси Ox}
  for i:=1 to 4 do      begin
          x:=trunc(i*mx); line(x,y0-2,x,y0+2);
      end;
              {Разметка оси Oy}
  outtextxy(x0+trunc(mx),y0-10,'1');
  for i:=1 to 9 do
      begin
          y:=trunc(i*my);              line(x0-2,y,x0+2,y);
      end;
  outtextxy(x0-10,y0-trunc(my),'1');
          {Построение графика}
  x1:=-3;
  while x1<=2 do
      begin
          y1:=x1*x1;
          x:=x0+trunc(x1*mx);
          y:=y0-trunc(y1*my);
          putpixel(x,y,15);
          x1:=x1+0.01;
      end;
  repeat until keypressed;
  closegraph;
END.
```

Пример. Изображение моделей полета тела брошенного под углом к горизонту с заданной скоростью, но разными углами. Здесь масштабы *ex* и *ey* сразу вычисляются целыми числами. Начальная скорость бросания *v=20*, углы от 10 до 70 через 5. Время с шагом 0.1.

```
program polet;
uses crt,graph;
var dm,dr,cx,cy,x,y,ex,ey,l,h:integer;x1,y1,f,v,g,t:real;
BEGIN
  dr:=detect;
  initgraph(dr,dm,'o:\bp70\bgi');
  cx:=10;cy:=340; {координаты отображения на экране начала координат – левый нижний угол}
  l:=50;h:=20;            {предполагаемая максимальная дальность и высота полета}
 f:=45;v:=20             {начальные данные: угол в градусах, скорость бросания}
 g:=9.8;
  ex:=630 div l;ey:=340 div h; {масштаб}
  line(cx,0,cx,350);   line(0,cy,640,cy); {оси координат}
  y:=cy;       {разметка оси Oy}
  while y>=0 do
      begin
          y:=y-ey;          line (cx-2,y,cx+2,y);
      end;
  {разметка оси Ox}

x:=cx;
  while x<=640 do
      begin
          x:=x+ex;          line(x,cy-2,x,cy+2);
      end;
  {построение графиков}
  f:=10;
  while f<=70 do
```

```
    begin
      y1:=0;t:=0;      {Построение траектории полета для одного заданного угла f }
      while y1>=0 do
        begin
                                          {Математические расчеты  }
          x1:=v*cos(f*3.14/180)*t;
          y1:=v*sin(f*3.14/180)*t-g*t*t/2;

        {Преобразование значений к целому виду с учетом масштаба для вывода на экран }
          x:=trunc(x1*ex);
          y:=trunc(y1*ey);
          putpixel(cx+x,cy-y,15);
          t:=t+0.1; {изменение времени}

        end;
      f:=f+5; {изменение величины угла}
    end;
  readln;
  closegraph;
END.
```