Л А Б О Р А Т О Р Н А Я Р А Б О Т А N2.

Преобразование изображений: перенос, масштабирование, поворот, комбинированные преобразования.

Цель работы: познакомиться с назначением, областями применения, сущностью операций преобразования; научиться разрабатывать программы, осуществляющие преобразования изображений на плоскости.

Результат: должна быть разработана программа, реализующая операции преобразования. Выбор операции, задание параметров, определяющих каждую операцию, должно выполняться с помощью меню или с помощью системы кнопок и окон для ввода параметров. Должны быть реализованы функции вывода исходного изображения, текущего изображения, возврата к предыдущему изображению. Каждое преобразование должно применяться к текущему изображению.

При построении изображений часто приходится иметь дело с ситуациями, когда общее изображение (рисунок) включает в себя целый ряд компонент (подрисунков), отличающихся друг от друга только местоположением, ориентацией, масштабом, т.е. отдельные подрисунки обладают значительным геометрическим сходством.

В этом случае целесообразно описать один подрисунок в качестве базового, а затем получать остальные требуемые подрисунки путем использования операций преобразования.

С помощью операций преобразования можно выполнять следующие действия: 1) перемещать рисунки из одного места экрана в другое; 2) создавать рисунок из более мелких элементов (составных частей); 3) добавлять к существующему рисунку новые элементы; 4) увеличивать размер рисунка для улучшения его наглядности или отображения более мелких деталей; 5) уменьшать размер рисунка для внесения, например, поясняющих надписей или отображения на экране новых рисунков; 6) создавать движущиеся изображения.

Все изменения рисунков можно выполнить с помощью трех базовых операций: 1) переноса (перемещения) изображения; 2) масштабирования (увеличения или уменьшения размеров) изображения; 3) поворота изображения (употребляют также термины вращение, изменение ориентации).

Для реализации перечисленных операций используется аппарат линейных преобразований.

Линейное преобразование на плоскости - это такое отображение плоскости в себя, при котором прямая переходит в прямую. Произвольная точка с координатами (X,Y) переходит в результате линейного преобразования в точку с координатами (X1,Y1) в соответствии с выражениями:

$$X1 = A*X + B*Y + C$$
$$Y1 = D*X + E*Y + F, \qquad (1)$$

где A,B,C,D,E,F - коэффициенты данного преобразования, однозначно его определяющие.

Последовательное применение двух линейных преобразований можно заменить одним третьим, эквивалентным, которое называется их произведением.

Формулы линейного преобразования (1) можно записать в матричной форме:

$$(X1,Y1,1) = (X,Y,1)*\begin{vmatrix} A & D & 0 \\ B & E & 0 \\ C & F & 1 \end{vmatrix} = (X,Y,1)*M, \qquad (2)$$

где через M обозначается матрица преобразования.

Здесь используются однородные координаты, которые были введены в геометрии и ныне широко используются в машинной графике. В однородных координатах точка P(X,Y) записывается в виде P(W*X,W*Y,W) для любого масштабного множителя W<>0. При этом если для точки задано ее представление в однородных координатах P(X,Y,W), то можно найти ее двумерные декартовы координаты как x=X/W, y=Y/W. Для двумерного пространства W=1, поэтому операция деления не требуется. Однородные координаты можно представить как вложение промасштабированной с коэффициентом W двумерной плоскости в плоскость z=W (здесь z=1) в трехмерном пространстве. Следует добавить, что с помощью матрицы преобразования

$$(X1,Y1)=(X,Y)*\begin{vmatrix} A & B \\ C & D \end{vmatrix}$$

нельзя описать операцию переноса.

Для двух последовательно выполняемых линейных преобразований можно записать следующее выражение:

(X2,Y2,1) = (X1,Y1,1)*M2 = (X,Y,1)*M1*M2 = (X,Y,1)*M, (3)

где X,Y - координаты исходной точки;

X1,Y1 - координаты точки после первого преобразования;

X2,Y2 - координаты точки после второго преобразования; M1,M2,M - матрицы соответственно первого, второго и результирующего преобразований.

Поскольку в общем случае операция умножения матриц не является коммутативной, то в общем случае и два последовательных линейных преобразования некоммутативны.

Если определитель матрицы преобразования отличен от нуля, то такое преобразование будет являться аффинным. При аффинном преобразовании плоскость не может вырождаться в линию или точку, параллельные прямые переводятся в параллельные и всегда имеется обратное преобразование.

Аффинное преобразование может быть представлено суперпозицией трех преобразований: переноса, масштабирования, поворота.

Перенос изображения

Перенос изображения заключается в перемещении отображенного объекта из одного места экрана в другое место. Перенос изображения позволяет построить рисунок в произвольном месте
экрана и затем перенести его в другую, требуемую, часть экрана. При этом можно изменить компоновку рисунка или создать единый рисунок из набора готовых элементов.

Для переноса точки из позиции с координатами (X,Y) в позицию с координатами (X1,Y1) надо к координате X добавить DX, а к координате Y - DY единиц, причем DX = X1-X, DY = Y1-Y.

Матрица преобразования M для операции переноса имеет следующий вид:

$$M = \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ DX & DY & 1 \end{vmatrix} \qquad (4)$$

При подстановке ее в (2), получим

$$(X1,Y1,1) = (X,Y,1)* \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ DX & DY & 1 \end{vmatrix}$$

или

X1 = X+DX, Y1 = Y+DY.

Положительное значение DX означает перемещение точки вправо по горизонтали, отрицательное - влево; положительное значение DY - перемещение вниз по вертикали, отрицательное - вверх.

Необходимо задавать такие значения DX, DY, чтобы после преобразования точка оставалась в пределах экрана, иначе она высвечиваться не будет. Не следует задавать слишком малые значения DX, DY (DX < 0,5, DY < 0,5), так как в этом случае точка повторно высвечивается на старом месте.

Перенос рисунка из одной области экрана в другую эквивалентен переносу всех точек рисунка и последующему высвечиванию соединяющих их линий. Для исключения искажения рисунка все точки необходимо переместить на одинаковое расстояние, т.е. использовать одну и ту же матрицу преобразования.

Для фигур, обладающих симметрией, или с границами, вычисляемыми с помощью уравнений, при переносе изображения нет необходимости добавлять смещение к координатам всех точек. Например, для переноса окружности достаточно перенести ее центр и вычертить ее, зная радиус. Таким же образом осуществляется перенос эллипса.

Масштабирование изображения

При создании изображения на экране дисплея может возникнуть необходимость изменения его размеров с целью повышения его наглядности, для вставки созданного изображения в уже имеющийся рисунок. Размер рисунка можно изменить, если умножить все расстояния между точками на некоторую постоянную величну (коэффициент масштабирования). Если коэффициент масштабирования больше единицы, то рисунок увеличивается, если меньше единицы - уменьшается.

Наряду с коэффициентом масштабирования для выполнения масштабирования надо указать новое положение рисунка (после выполнения масштабирования). Новое положение рисунка определяется центром масштабирования - некоторой центральной точкой, относительно которой выполняется масштабирование.

В качестве такой точки может быть выбрана центральная точка рисунка, точка, лежащая на границе рисунка, точка, лежащая вне рисунка и даже вне экрана.

Масштабирование может быть однородным (коэффициенты масштабирования вдоль осей абсцисс и ординат одинаковы), в этом случае пропорции рисунка сохраняются. Масштабирование может быть и неоднородным (коэффициенты вдоль осей абсцисс и ординат различны), в этом случае пропорции рисунка изменяются.

Неоднородное масштабирование может применяться при подборе пропорций рисунка, в процессе конструирования и проектирования при подборе размеров объекта.

Матрица преобразования при масштабировании имеет вид

$$M = \begin{vmatrix} KX & 0 & 0 \\ 0 & KY & 0 \\ 0 & 0 & 1 \end{vmatrix} \qquad (5)$$

При подстановке ее в (2) получим:

$$(X1,Y1,1) = (X,Y,1)* \begin{vmatrix} KX & 0 & 0 \\ 0 & KY & 0 \\ 0 & 0 & 1 \end{vmatrix} \text{ или}$$

X1 = X*KX, Y1 = Y*KY,

где KX - коэффициент масштабирования по оси абсцисс;

KY - коэффициент масштабирования по оси ординат. Применяя преобразование (5) ко всем точкам рисунка,

получим рисунок промасштабированный относительно начала координат. При этом, если KX > 1 и KY > 1, то рисунок увеличивается в размере и удаляется от начала координат; если KX < 1 и KY < 1, то рисунок уменьшается в размерах и приближается к началу координат.

Координаты промасштабированной точки определяются из следующих выражений:

X1=X*KX+(1-KX)*XM

Y1=Y*KY+(1-KY)*YM (6)

где X,Y - координаты исходной точки;
X1,Y1 - координаты промасштабированной точки;
XM,YM - координаты центра масштабирования;
KX,KY - коэффициенты масштабирования.

Масштабирование рисунка

Для масштабирования рисунка необходимо в соответствии с (6) вычислить новые координаты всех точек нового изображе-
ния, а затем полученные точки соединить линиями. При этом размеры рисунка равномерно увеличиваются или уменьшаются, если KX=KY.

Однако не всегда надо вычислять координаты всех точек нового рисунка. Например, при масштабировании окружности, достаточно вычислить новые координаты ее центра, а в качестве радиуса взять величину k*R (k - коэффициент масштабирования, R - радиус исходной окружности). Таким же образом можно поступить при масштабировании эллипсов, прямоугольников.

При вычислении координат точек нового рисунка следует иметь в виду, что коэффициент масштабирования, как правило, величина действительная, а координаты точек на экране должны быть целыми, поэтому в программе необходимо использовать в этом случае операцию округления.

С помощью масштабирования можно растянуть или сжать изображение вдоль одной координатной оси, оставив его без изменения вдоль другой оси. Например, масштабируя квадрат с коэффициентами масштабирования KX = 1, KY =2, получим прямоугольник, у которого большая сторона имеет вертикальное расположение.

Неравномерное масштабирование окружности приводит к тому, что будет изображен эллипс. Но в этом случае программист должен воспользоваться процедурой рисования эллипса, а не окружности. В такой ситуации в целях общности целесообразно далее окружности вычерчивать процедурой рисования эллипса (задавая равные значения полуосей).

Поворот изображения

Наиболее часто поворот изображения используется при создании движущихся изображений в моделирующих, игровых программах. Однако, иногда бывает удобно повернуть на 90 график, гистограмму. В процессе проектирования также необходимо поворачивать изображение создаваемого объекта, чтобы рассмотреть его с разных сторон и избежать возможных ошибок.

Поворот точки.

Для выполнения поворота надо указать величину угла, на который необходимо осуществить поворот, и координаты точки, которая берется за центр вращения. Если исходную точку A с координатами (X,Y) по дуге окружности с центром в точке C с координатами (Xc,Yc) поворачивают на угол t, то координаты (X1,Y1) повернутой точки могут быть записаны в следующем виде:

X1 = Xc + (X-Xc)*cost+ (Y-Yc)*sint*rx/ry
Y1 = Yc + (Y-Yc)*cost- (X-Xc)*sint*ry/rx, (7)

где rx, ry - разрешающие способности вдоль оси X и Y соответственно.

Значения rx/ry (ry/rx) могут быть получены в Турбо-Паскале с помощью процедуры GetAspectRatio, они являются обратными величинами по отношению к коэффициентам, выдаваемым процедурой: rx/ry= ya/xa, ry/rx=xa/ya (см. лабораторную работу N1).

Если центр вращения совпадает с началом координат (Xc=0, Yc=0), то матрица преобразования имеет вид:

$$M = \begin{vmatrix} cost & -sint & 0 \\ sint & cost & 0 \end{vmatrix} \quad (8)$$

| 0 0 1 |
- -

Если начало координат расположено в левой верхней точке экрана, то угол поворота измеряется в направлении против часовой стрелки. Если же начало координат лежит в левой нижней точке экрана, то угол поворота должен измеряться в направлении по часовой стрелке.

Центр поворота может быть расположен в любом месте экрана, а также за пределами его границ.

Угол поворота лежит обычно в пределах от 0 до 360 , другие углы поворота также допустимы, однако поворот при этих углах эквивалентен повороту при углах из указанного диапазона.

Поворот рисунка.

Для поворота всего рисунка прежде всего необходимо выбрать центр поворота и определить угол, на который необходимо повернуть исходное изображение. После этого в соответствии с (7) вычислить координаты всех точек рисунка и соединить вновь полученные точки линиями.

Однако не всегда требуется расчет новых координат всех точек изображения: для прямоугольника это только четыре точки - его вершины, для окружности необходимо рассчитать только координаты центра и вычертить ее на новом месте.

При вычислении координат следует иметь в виду, что для хранения новых координат точек нельзя использовать ту же область памяти, где хранятся исходные координаты. Это объясняется тем, что при вычислении координаты Y1 в выражении (7) используется значение исходной координаты X. Таким образом, используя одну область памяти, мы будем затирать нужную нам старую координату X и неправильно вычислять новое значение координаты Y1.

Следует обратить внимание на то, что результат вычисления значений функций синуса и косинуса - действительный, а координаты точек должны быть целыми, поэтому в программе необходимо использовать операцию округления.

Преобразованное изображение (повернутое, промасштабированное или перенесенное) может высвечиваться вместе с исходным. Если же это нежелательно, то перед рисованием нового изображения необходимо произвести очистку экрана.

При выполнении преобразований желательно заранее предвидеть расположение рисунка, т.к. точки, выходящие за границы экрана, не высвечиваются. Если в программе введен контроль значений координат точек, то он только может обеспечить поиск ошибки, но программу в этом случае все равно придется корректировать.

Программа lab_31 строит исходную астроиду (см. работу N1) с осями, параллельными координатным осям, а затем может построить ее промасштабированное, перенесенное или повернутое на некоторый угол изображение. Поскольку заранее не известно количество точек, аппроксимирующих кривую, то для хранения координат точек исходной астроиды используется динамический список.

ТЕКСТ ПРОГРАММЫ ПОСТРОЕНИЯ АСТРОИДЫ И ВЫПОЛНЕНИЯ ПРЕОБРАЗОВАНИЙ

```
program lab_31;
  Выполнение преобразований на плоскости на примере
    плоской кривой - астроиды
uses Graph,Crt;
label 1;
const b=120; параметр астроиды
          fon=7;  цвет фона
```

```
          col=12; цвет рисования
type  spis=^p;
          p=record
            x,y:integer;
            praw:spis;
          end;
var i,              параметр цикла
      gd,gm, тип и режим работы адаптера
      xn,yn, координаты центра астроиды
      x,y,   координаты текущей точки исходной астроиды
      x1,y1, координаты текущей точки преобразованной астроиды
      xnc,ync, координаты начальной точки астроиды
        n,            количество точек, аппроксимирующих астроиду
      xc,yc,   координаты центра поворота
      xm,ym,   координаты центра масштабирования
      dx,dy          величины смещений
      :integer;
      a,c,d,f,       рабочие переменные
      dt,            шаг изменения аргумента при аппроксимации
      t,             текущее значение аргумента
      kx,ky,         коэффициенты масштабирования
      kx1,ky1,       единица минус коэффициент масштабирования
      ug,            угол поворота в градусах
      ur,            угол поворота в радианах
      rx,ry,         отношения коэффициентов, учитывающих разрешающие
                     способности
      sur,cur: значение синуса и косинуса угла поворота
      real;
      xa,ya:word;       коэффициенты учета разрешающих способностей
      nach,             указатель на начало списка
      kon,              указатель на конец списка
      rab:spis;         текущий указатель
      pr:char;          признак выбираемой функции
begin
          nach:=nil;
          kon:=nil;
          rab:=nil;
          gd:=detect;
          InitGraph(gd,gm,'d:\turbo');
          GetASpectRatio(xa,ya);
          xn:=GetMaxX div 2;  координаты центра
          yn:=GetMaxY div 2;                    кривой
          dt:=2.0/3.0/b; шаг изменения аргумента
          n:=round(2*pi/dt);
            вычисление координат точек исходной астроиды
            и формирование списка, хранящего координаты
          for i:=1 to n do
          begin
                    t:=(i-1)*dt;
                    a:=cos(t);
                    c:=a*a*a;
                    x:=xn+round(b*c);
                    f:=sin(t);
                    d:=f*f*f;
                    y:=yn-round(b*d*xa/ya);
                    new(rab);
```

```
        rab^.x:=x;
        rab^.y:=y;
        rab^.praw:=nil;
        if t=0 then nach:=rab
                else kon^.praw:=rab;
        kon:=rab;
end;

repeat
        RestoreCrtMode;
        writeln('Выберите нужную функцию');
        writeln('Исходное изображение - i(I)');
                writeln('Перенесенное изображение - p(P)');
        writeln('Промасштабированное изображение - m(M)');
        writeln('Повернутое изображение - w(W)');
        writeln('Конец работы - e(E)');
        pr:=Upcase(ReadKey);
        case pr of
 'P': begin
                writeln('Введите величины смещений (dx,dy)');
                readln(dx,dy);
        end;
 'M': begin
                writeln('Введите координаты центра масштабирования');
                writeln('(координаты центра астроиды',xn:5,yn:5,')');
                readln(xm,ym);
                writeln('Введите коэффициенты масштабирования');
                readln(kx,ky);
        end;
 'W': begin
                writeln('Введите координаты центра поворота');
                writeln('(координаты центра астроиды',xn:5,yn:5,')');
                readln(xc,yc);
                        writeln('Введите угол поворота');
                readln(ug);
        end;
 'E':  goto 1;
        else ;
      end;
        SetGraphMode(GetGraphMode);
        SetBkColor(fon);
        SetColor(col); rab:=nach; kx1:=1-kx;
        ky1:=1-ky; ur:=ug*pi/180; sur:=sin(ur);
        cur:=cos(ur); rx:=xa/ya; ry:=ya/xa; while
        rab<>nil do begin
                x:=rab^.x; y:=rab^.y; case pr of
  'I':  begin
                        x1:=x; y1:=y;
            end;
  'P':  begin
                        x1:=x+dx; y1:=y+dy;
            end;
  'M': begin x1:=round(x*kx+xm*kx1);
                    y1:=round(y*ky+ym*ky1); end;
  'W':  begin
                    x1:=round(xc+(x-xc)*cur+(y-yc)*sur*ry);
```

```
                        y1:=round(yc+(y-yc)*cur-(x-xc)*sur*rx);
                    end;
                    else ;
                      end;
                    if nach=rab then
                    begin
                        xnc:=x1;
                        ync:=y1;
                        MoveTo(xnc,ync); end
                    else LineTo(x1,y1);
                    rab:=rab^.praw;
                end;
                LineTo(xnc,ync); readln;
                    until (pr='E');
1:          CloseGraph;
    end.
```

Композиция преобразований.

Композиция преобразований представляет собой совокупность последовательного выполнения базовых преобразований: переноса, масштабирования, поворота. Зная матрицы преобразований, можно заранее вычислить результирующую матрицу преобразований как произведение отдельных матриц и использовать уже полученную матрицу для определения итоговых координат точек рисунка, не вычисляя промежуточных координат.

Однако операция умножения матриц достаточно трудоемка, поэтому желательно знать такие варианты преобразований, когда можно не использовать перемножение матриц.

Полезно знать, что перенос и поворот являются аддитивными операциями, а масштабирование - мультипликативной операцией, то есть для нахождения результирующего преобразования в первом случае надо суммировать значения отдельных переносов (в случае поворота итоговый угол равен сумме отдельных углов поворота), во втором случае коэффициенты масштабирования следует перемножить для нахождения результирующего коэффициента масштабирования.

В ряде частных случаев наблюдается коммутативность операций преобразования. В таблице приведены варианты таких преобразований:

M1	M2
Перенос	Перенос
Масштабирование	Масштабирование
Поворот	Поворот
Масштабирование (однородное)	Поворот