МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Рубцовский индустриальный институт (филиал)
федерального государственного бюджетного образовательного
учреждения высшего образования
«Алтайский государственный технический университет им. И.И. Ползунова»
(РИИ АлтГТУ)

Е.А. ДУДНИК

ИНЖЕНЕРНАЯ И КОМПЬЮТЕРНАЯ ГРАФИКА

Часть I

Основные направления и методы компьютерной графики

Методическое пособие
для бакалавров направления подготовки
«Информатика и вычислительная техника»

Рубцовск 2021

УДК 519.6

Дудник, Е.А. Инженерная и компьютерная графика. Часть I. Основные направления и методы компьютерной графики: Методическое пособие для бакалавров направления подготовки «Информатика и вычислительная техника» /Е.А. Дудник; Рубцовский индустриальный институт. – Рубцовск, 2021, – 47 с. [ЭР].

Данное пособие состоит из двух частей, приведены описания методов компьютерной графики: преобразования графических объектов на плоскости и в пространстве, растровые алгоритмы, интерполирование сплайнами, базовые геометрические алгоритмы, возможности графического стандарта OpenGL. Составлены задания на лабораторный практикум, приведены темы курсовых работ по курсу компьютерной графики.

В первой части подробно изложены методы работы и способы построения изображений на персональных компьютерах, необходимые для выполнения лабораторных работ.

Рассмотрено и одобрено на заседании
кафедры прикладной математики.
Протокол № 9 от 18.03.21 г.

Содержание

ВВЕДЕНИЕ 4
1. СОЗДАНИЕ ИЗОБРАЖЕНИЙ ПЛОСКИХ ГЕОМЕТРИЧЕСКИХ ОБЪЕКТОВ 5
1.1. Построение простейших геометрических объектов 5
1.2. Однородные координаты точки на плоскости 7
1.3. Построение правильного многоугольника 8
1.4. Построение кривых на плоскости 8
2. АФФИННЫЕ ПРЕОБРАЗОВАНИЯ НА ПЛОСКОСТИ И В ПРОСТРАНСТВЕ 10
2.1. Аффинные преобразования на плоскости 10
2.2. Перенос точки на плоскости 11
2.3. Масштабирование расстояния точки от центра на плоскости 11
2.4. Поворот точки относительно центра на плоскости 12
2.5. Аффинные преобразования составных графических объектов. Композиция преобразований 13
2.6. Аффинные преобразования в пространстве 14
3. ДЕЛОВАЯ ГРАФИКА 16
3.1. Построение графика функции 16
3.2. Построение гистограммы, круговой диаграммы 17
4. ВИДЫ ПРОЕКТИРОВАНИЯ 18
4.1. Параллельные проекции 19
4.2. Перспективные проекции 20
5. РАСТРОВЫЕ АЛГОРИТМЫ 21
5.1.Основные понятия 21
5.2. Построение отрезка 22
5.3. Заполнение сплошных областей 23
6. ИНТЕРПОЛЯЦИЯ СПЛАЙНАМИ 24
6.1. Сплайн – функции 25
6.2 Сплайновые кривые 28
7. ГЕОМЕТРИЧЕСКОЕ ИЗОБРАЖЕНИЕ ГРАФИКА ФУНКЦИИ ДВУХ ПЕРЕМЕННЫХ 29
8. РАБОТА С ОСНОВНЫМИ ГРАФИЧЕСКИМИ УСТРОЙСТВАМИ 33
8.1. Основные операции работы с мышью 33
8.2. Организация простейшего меню 34
9. ЗАДАНИЯ НА ЛАБОРАТОРНЫЙ ПРАКТИКУМ 37
ЛАБОРАТОРНАЯ РАБОТА № 1 37
ЛАБОРАТОРНАЯ РАБОТА № 2 38
ЛАБОРАТОРНАЯ РАБОТА № 4 40
ЛАБОРАТОРНАЯ РАБОТА № 5 41
ЛАБОРАТОРНАЯ РАБОТА № 6 42
ЛАБОРАТОРНАЯ РАБОТА № 7 42
ЛАБОРАТОРНАЯ РАБОТА № 8 43
ПРИЛОЖЕНИЕ 1 44
СПИСОК ЛИТЕРАТУРЫ 47

ВВЕДЕНИЕ

Курс компьютерной графики является составной частью профессиональной подготовки студентов специальности 073002 «Прикладная математика». Компьютерная графика имеет большой практический интерес благодаря своей наглядности, эмоциональному и эстетическому воздействию. Проникновение компьютерных технологий во все сферы жизни общества увеличивается с каждым годом. Работа на персональном компьютере с информацией, представленной в виде изображений, удобна, перспективна и имеет массу практических приложений. Необходимость широкого использования графических программных средств особенно ощутима с развитием Интернета; Web-страница, оформленная без компьютерной графики, не привлекает к себе внимания.

При обработке информации, связанной с изображением на мониторе, принято выделять три основных направления: ***распознавание образов***, ***обработку изображений*** и ***машинную графику***. Направление «распознавание образов» представляет совокупность графических методов, с помощью которых преобразуется уже имеющееся изображение на формально понятный язык символов. Направление «обработка изображений» представляет совокупность графических методов, которые позволяют получить на экране изображение предмета по его фотографии или телевизионному изображению. Направление «машинная графика» представляет совокупность графических методов, с помощью которых воспроизводится изображение по тому или иному представлению объектов. Все направления взаимосвязаны, и развитие одного направления приводит к решениям задач другого направления.

Основными техническими средствами компьютерной графики являются ***видеоконтроллер*** (адаптер) и ***графический дисплей*** (монитор). Адаптеры могут быть следующих типов: CGA, EGA, VGA, SVGA, - различающиеся по возможностям, аппаратным устройствам и принципам работы. Однако большинство адаптеров строится по принципу совместимости с предыдущими. Например, адаптер EGA поддерживает все режимы с адаптером CGA.

Понятие ***разрешения экрана*** – это свойство компьютерной системы (зависит от монитора и видеокарты) и операционной системы. Разрешение экрана измеряется в пикселях и определяет размер изображения, которое может поместиться на экране целиком. ***Разрешение изображения*** – это свойства самого изображения, оно может измеряться в точках на дюйм.

Различают три вида компьютерной графики: ***растровая, векторная, фрактальная графика***. Они различаются принципами формирования изображения при отображении на экране монитора и при печати на бумаге.

В растровой графике основным элементом изображения является точка. Изображение на мониторе строится по значениям точек (пикселей), которые вычисляются и выводятся в видеопамять компьютера, а затем при отображении выбираются из памяти и воспроизводятся в соответствующем месте экрана. Изображения представляются в виде двумерной прямоугольной целочисленной матрицы пикселей (растра). Недостатком растровой графики является большой объем памяти данных и то, что растровые изображения невозможно увеличить

без визуального искажения. Основным достоинством является возможность получения качественного реалистического изображения.

В векторной графике элементом изображения является линия (кривая). Простейшие объекты объединяются в сложные, элементы изображения строятся из прямолинейных отрезков, задается направление в точку с координатами (x, y) и интенсивность луча. Например, четырехугольник можно представить как четыре связанные линии. Все объекты имеют свойства, например: форма линии, толщина, цвет. Векторная графика устраняет оба недостатка растровой графики: значительный объем массивов данных и невозможность масштабирования без потери качества. Однако средствами векторной графики сложно создать качественное, реалистическое изображение, используется в основном для чертежных и проектно-конструкторских работ.

Фрактальная графика, как и векторная – вычисляемая. Изображение строится по уравнению, поэтому кроме формул хранить ничего не надо. Изменив коэффициенты в уравнении, можно получить совершенно другую картину. Фрактальная фигура состоит из поколений, наследующих свойства своих родительских структур, например, снежинка. Способность фрактальной графики моделировать образ вычислительным путем используют для автоматической генерации необычных иллюстраций.

В первой части пособия подробно изложены алгоритмы и методы компьютерной графики:

- создание графических объектов;
- аффинные преобразования изображения на плоскости и в пространстве;
- методы проектирования трехмерных объектов на плоскость;
- построение растровых изображений;
- интерполяция сплайнами;
- построения изображения функции двух переменных с удалением невидимых линий.

Пособие является переработанным и дополненным изданием методического пособия «Лабораторный практикум по машинной графике» [1]. Для пополнения знаний изучаемого материала можно воспользоваться книгами [2-5]. Для реализации алгоритмов и компьютерных программ — [5-6].

1. СОЗДАНИЕ ИЗОБРАЖЕНИЙ ПЛОСКИХ ГЕОМЕТРИЧЕСКИХ ОБЪЕКТОВ

1.1. Построение простейших геометрических объектов

В растровой графике любая графическая операция сводится к работе с отдельными пикселями. Каждый компилятор имеет свою графическую библиотеку, обеспечивающую работу с основными группами графических объектов (представляющих собой объединение пикселей). В приложении 1 описаны группы операций: инициализация графического режима, линейных изображений, сплошных объектов, работа с видеопамятью.

Задача определения положения пикселя в растре является наиболее простой из графических программ. Экранные координаты задаются следующим

образом: ось X направлена слева направо, а ось Y – сверху вниз (координаты левого верхнего угла экрана (0,0), а координаты правого нижнего угла имеют максимальные значения X и Y).

Пример функции *перехода из декартовой системы координат к экранным координатам*, W, H – разрешения экрана:

```
function Xscr(x:integer):integer;
  begin
   Xscr:=x+W div 2
  end;
function Yscr(y:integer):integer;
  begin
   Yscr:=H div 2 - y
  end;
```

Пример программы *построения осей координат*:

```
uses graph;{подключение графического модуля graph}
const
  d1=10;
  d2=2;
   path=' '; {файлы с расширением *.bgi  находятся в рабочем каталоге}
var
  x0,y0,W,H,x,y:integer;
{процедура инициализации графического режима}
procedure Initc;
   Var gd,gm,rc:integer;
    begin
      gd:=detect;
      initgraph(gd,gm, path);
      rc:=graphresult;
     if (rc<>grOk) then
      begin
       writeln(' ');
       halt(1);
      end;
    end;
{переход из декартовой системы координат к экранной системе координат
(function xscr, function yscr) }
begin
 Initc;
 W:=getmaxx; {функции определения разрешения экрана}
 H:=getmaxy;
 x0:=0; y0:=0;
 x:=xscr(x0); y:=yscr(y0);
 line(0,y,W,y);
 line(x,0,x,H);
 line(w,y,w-d1,y-d2);
```

```
 line(x-d2,d1,x,0);
 line(w,y,w-d1,y+d2);
 line(x+d2,d1,x,0);
 outtextxy(x+d2,y+d2,'(0,0)');
 readln;
 closegraph;
end.
```

Замечания по построению плоских геометрических объектов:

1. Выбор алгоритма. Если изображение составное, то необходимо разбить его на элементарные фигуры (точки, отрезки, прямоугольники, дуги, окружности и т.д).
2. При выводе объекта на экран нужно выбрать опорную точку, относительно которой вычисляются все остальные координаты. Тогда изменение координат опорной точки автоматически повлечет за собой измение координаты всех других точек.
3. Желательно использовать минимальное количество параметрических чисел. ***Параметрическим числом*** геометрического объекта называется минимальное количество параметров, задающих этот объект. Объекты могут задаваться разным количеством и сочетанием параметров (например, квадрат можно задать координатами центра и величиной его стороны).
4. В выражениях не следует использовать константы, затрудняющие последующую корректировку программы, лучше использовать именованные переменные (отражающие в имени назначения переменной).
5. Вычисляемые координаты точек на экране монитора являются целыми неотрицательными величинами и не должны превосходить максимальных разрешающих значений экрана по X и Y.
6. Следует выбирать наиболее удобную процедуру из числа имеющихся в составе модуля *Graph* (см. приложение 1), использовать глобальные константы, позволяющие облегчить написание программ, улучшить их понимание и восприятие.

Упражнение

1. Построить графический объект с определением опорной точки:

а) квадрат; б) треугольник.

1.2. Однородные координаты точки на плоскости

Пусть M – произвольная точка плоскости с координатами x и y, вычисленными относительно заданной прямолинейной координатной системы. Однородными координатами этой точки называется любая тройка одновременно не равных нулю чисел x_1, x_2, x_3, связанных с заданными числами x и y следующими соотношениями:

$$\frac{x_1}{x_3} = x, \frac{x_2}{x_3} = y. \qquad (1.1)$$

При решении задач компьютерной графики однородные координаты обычно вводятся так: произвольной точке M(x, y) плоскости ставится в соответствие точка $(x, y, 1) \in$ M в пространстве.

Заметим, что произвольная точка на прямой, соединяющей начало координат, точку O(0,0,0), с точкой $(x, y, 1) \in$ M, может быть задана тройкой чисел вида (h_x, h_y, h).

Будем считать, что $h \neq 0$.

Вектор с координатами h_x, h_y, h является направляющим вектором прямой, соединяющей точки O(0,0,0) и $(x, y, 1) \in$ M. Эта прямая пересекает плоскость $z = 1$ в точке $(x, y, 1)$, которая однозначно определяет точку (x, y) координатной плоскости xy.

Тем самым между произвольной точкой с координатами (x, y) и множеством троек чисел вида $(h_x, h_y, h), h \neq 0$, устанавливается (взаимно однозначное) соответствие, позволяющее считать числа h_x, h_y, h новыми координатами этой точки.

1.3. Построение правильного многоугольника

Многоугольник называется правильным, если все углы и длины сторон равны. Для построения правильного треугольника на экране по заданному числу сторон нужно определить центральный угол стороны $\alpha = \frac{2\pi}{n}$ (где n-число сторон). Центр многоугольника совместить с центром экрана. Начальную вершину многоугольника поместить на горизонтальной оси (правее центра). Угол наклона i-й вершины к оси OX составляет $\alpha_i = \alpha \cdot i$, а ее координаты определяются следующим образом:

$$x = round(R \cdot cos(\alpha \cdot i)), \qquad (1.2)$$

$$y = round(R \cdot sin(\alpha \cdot i)),$$

где R – радиус описанной окружности.

Упражнение

1. Построить графический объект:

а) шестиугольник; б) двенадцатиугольник.

2. Изобразить вращение отрезка вокруг своего центра, совпадающего с центром экрана, против часовой стрелки.

1.4. Построение кривых на плоскости

Часто требуется построить плоское изображение кривой по заданному уравнению. Кривая аппроксимируется отрезками прямых или близко расположенными точками. Процесс сводится к нахождению координат точек, принадлежащих кривой. Для этого, изменяя в цикле значение аргумента (абсцисс точек), по уравнению кривой вычисляем соответствующие значения функции (ординат точек), полученные точки соединяем отрезками.

Кривизной линии в точке M называется число $K = \lim\limits_{\Delta s \to 0} \left(\frac{\Delta \varphi}{\Delta s} \right)$, где φ – угол поворота касательной на дуге MN, Δs - длина этой дуги. Наилучший результат

аппроксимации достигается, если при выбранном шаге изменения аргумента будут высвечиваться каждый раз две смежные точки растра, лежащие на кривой. Следует придерживаться правила: при достаточно большом радиусе кривизны кривой две соседние точки должны выбираться таким образом, чтобы величина угла (выраженная в радианах), образованного радиусами в рассматриваемых точках, была равна 1/R, где R – максимальный радиус кривизны кривой в выбранных точках.

Приведем в качестве справочного материала формулы для расчета радиуса кривизны кривой.

1) При задании кривой в прямоугольной системе координат в виде уравнения ***Y=f(X)*** радиус кривизны равен

$$R = \frac{(1 + y'^{2})^{3/2}}{|y''|}. \qquad (1.3)$$

2) При задании кривой параметрическими уравнениями $x=f_1(t)$, $y=f_2(t)$

$$R = \frac{(x'^{2} + y'^{2})^{3/2}}{|x'y'' - x''y'|} \qquad (1.4)$$

(дифференцирование берется по параметру t).

3) При задании кривой в полярной системе координат:

$$R = \frac{(p^{2} + p'^{2})^{3/2}}{|p^{2} + 2p'^{2} - pp''|}, \qquad (1.5)$$

где p – полярный радиус; φ – полярный угол (дифференцирование берется по параметру φ).

Если совместить ось OX декартовых прямоугольных координат и полярную ось OP, то декартовы координаты точки M(x,y) и ее полярные координаты *(φ,p)* будут связаны зависимостью:

$$x=p\cos(\varphi),\ y=p\sin(\varphi). \qquad (1.6)$$

Нужно найти максимальный радиус кривизны (соответствующий ему шаг изменения аргумента будет минимальным и равен 1/R) и провести аппроксимацию на основе этого значения.

Задание: построить кривую с использованием радиуса кривизны. Нужно построить кривую $x=f_1(t)$, $y=f_2(t)$, $0<=t<=2\pi$, заданную в параметрическом виде. Найдем *r* – максимальный радиус кривизны по формуле (1.3). Определим ***dt=1/r*** – шаг изменения аргумента.

Пример программы *построения кривой*

```
Procedure krub;
begin
 t:=0; x:=f1(t); y:=f2(t);
moveto(xscr(x), yscr(y)); { Установить курсор в начальную точку}
  repeat
    t:=t+dt
    x:=f1(t);  y:=f2(t);
    lineto(xscr(x),yscr(y));
  until t>=2*PI;
```

end;

Упражнения

1. Нарисовать сетку, заполняющую прямоугольную область, центр которой совпадает с центром экрана:

а) стороны прямоугольной области параллельны сторонам экрана;

б) прямоугольная область повернута на некоторый угол.

2. АФФИННЫЕ ПРЕОБРАЗОВАНИЯ НА ПЛОСКОСТИ И В ПРОСТРАНСТВЕ

2.1. Аффинные преобразования на плоскости

Все не вырожденные преобразования графических объектов можно выполнить с помощью трех базовых операций, соответствующие трем матрицам аффинных преобразований.

1) ***переносу*** (перемещению) точки на плоскости соответствует матрица переноса с вектором переноса ***(dx,dy),*** соответствующим направлениям по осям координат:

$$\boldsymbol{T} = \begin{pmatrix} \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \boldsymbol{dx} & \boldsymbol{dy} & \mathbf{1} \end{pmatrix} \qquad (2.1)$$

2) ***масштабированию*** точки на плоскости относительно начала координат соответствует матрица масштабирования с коэффициентами $\boldsymbol{k_x}$, $\boldsymbol{k_y}$ (увеличения ***(k>1)*** или уменьшения размеров ***(0<k<1)),*** соответствует направлениям по осям координат:

$$\boldsymbol{D} = \cdot \begin{pmatrix} \boldsymbol{k_x} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{k_y} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} \end{pmatrix} \qquad (2.2.)$$

3) ***повороту*** точки на плоскости (вращение, изменение ориентации) относительно начала координат $\boldsymbol{(x_c=0, y_c=0)}$ на угол $\boldsymbol{\varphi}$, матрица преобразования имеет вид:

$$\boldsymbol{M} = \begin{pmatrix} \boldsymbol{\cos\varphi} & \boldsymbol{\sin\varphi} & \mathbf{0} \\ \boldsymbol{-\sin\varphi} & \boldsymbol{\cos\varphi} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} \end{pmatrix}. \qquad (2.3)$$

Для реализации перечисленных операций используется аппарат линейных преобразований. 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 преобразуются в параллельные и всегда имеется обратное преобразование.

Для того, чтобы аффинное преобразование можно было представить суперпозицией трех преобразований: переноса, масштабирования, поворота — необходимо перейти к описанию произвольной точки плоскости уже не парой чисел, а тройкой.

2.2. Перенос точки на плоскости

Для переноса точки из позиции с координатами ***(x,y)*** в позицию с координатами ***(x_1,y_1)*** надо к координате ***x*** добавить величину ***dx***, а к координате ***y – dy***, причем ***dx = x_1-x, dy = y_1-y***, вектор переноса в этом случае равен ***(dx,dy).*** Для эффективного использования аффинных преобразований изображений более удобна их матричная запись:

$$(x_1, y_1, 1) = (x, y, 1) \cdot \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ dx & dy & 1 \end{pmatrix} = \begin{pmatrix} x + dx \\ y + dy \\ 1 \end{pmatrix}. \qquad (2.5)$$

Пример процедуры *переноса точки на плоскости m(x,y) с вектром перемещения (dx, dy).*

```
procedure perenos(x,y:integer;var x1,y1:integer);
 begin
   x1:=x+dx;
   y1:=y+dy;
end;
```

Перенос изображения – аффинное преобразование, позволяющее перемещать графический объект из одного места экрана в другую часть экрана на заданный вектор переноса. Если параметры графического объекта определены через опорную точку, тогда достаточно определить вектор переноса только для опорной точки.

2.3. Масштабирование расстояния точки от центра на плоскости

Наряду с коэффициентом масштабирования для выполнения операции масштабирования надо указать центр – некоторую произвольную центральную точку, относительно которой выполняется масштабирование.

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

Матрица преобразования для выполнения операции масштабирования относительно начала координат:

$$(x_1, y_1,) = (x, y, 1) \cdot \begin{pmatrix} k_x & 0 & 0 \\ 0 & k_y & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} x \cdot k_x \\ y \cdot k_y \\ 1 \end{pmatrix}, \qquad (2.5)$$

где k_x – коэффициент масштабирования по оси абсцисс; k_y – коэффициент масштабирования по оси ординат.

Применяя преобразование (2.4) ко всем точкам рисунка, получим рисунок, масштабированный относительно начала координат. При этом, если $k_x > 1$ и $k_y > 1$, то рисунок увеличивается в размере и удаляется от начала координат; если $0<k_x < 1$ и $0<k_y < 1$, то рисунок уменьшается в размерах и приближается к началу координат.

Алгоритм получения для точки ***(x,y)*** координат масштабированной точки ***(x_1,y_1)*** относительно центра масштабирования **($x_м$,$y_м$)** состоит из трех шагов:

1 шаг. Вектор переноса определяется из условия совмещения центра масштабирования и начала координат, на вектор ***(-$x_м$,-$y_м$),*** см. (2.1).

2 шаг. Масштабирование на заданный коэффициент относительно начала координат, см. (2.2).

3 шаг. Вектор перенос определяется из условия возвращения центра масштабирования в прежнее положение, на вектор ***($x_м$,$y_м$),*** см. (2.1).

Координаты масштабированной точки определяются из следующих выражений:

$$x_1 = x \cdot k_x + (1 - k_x) \cdot x_m$$
$$y_1 = y \cdot k_y + (1 - k_y) \cdot y_m , \qquad (2.6)$$

где ***x,y*** - координаты исходной точки;

x_1,y_1 - координаты масштабированной точки;

x_m,y_m - координаты центра масштабирования;

kx,ky - коэффициенты масштабирования.

Задание. Изобразить прямоугольник с координатами левого верхнего угла (1,1) и правого нижнего угла (1000, 2000), центр масштабирования – точка (100, 200), в центре прямоугольной области экрана размером (400,200).

k_x – равно отношению разности ***X_{max} и X_{min}*** экрана, отведенного для изображения объекта, к максимальному значению по оси абсцисс (***k_x***=400/1000)

k_y – равно отношению разности ***Y_{max} и Y_{min}*** экрана, отведенного для изображения объекта, к максимальному значению по оси ординат (***k_y***=200/2000).

Пример процедуры *масштабирования графического объекта*

```
procedure macstab(x,y,kx,ky,xm,ym:integer; var x1,y1:integer);
 begin
x1=kx* (x-xm)+xm;
y1=y*ky+(1-ky)* ym;
 end;
```

Размер рисунка можно изменить, если умножить все расстояния между точками на некоторую постоянную положительную величину (коэффициент масштабирования).

2.4. Поворот точки относительно центра на плоскости

Для выполнения поворота графического объекта на заданный угол относительно центра поворота. Центр поворота может быть расположен в любом месте экрана, а также за пределами его границ.

Матрица преобразования для выполнения операции поворота точки на плоскости относительно начала координат на угол φ :

$$(x_1, y_1,) = (x, y, 1) \cdot \begin{pmatrix} \cos\varphi & \sin\varphi & 0 \\ -\sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} x \cdot \cos\varphi - y \cdot \sin\varphi \\ x \cdot \sin\varphi + y \cdot \cos\varphi \\ 1 \end{pmatrix} \quad (2.7)$$

Алгоритм *поворота произвольной точки относительно заданного центра поворота состоит из следующих шагов* :

1 шаг. Вектор переноса определяется из условия совмещения центра поворота с началом координат, на вектор $(-x_м,-y_м)$, см. (2.1).

2 шаг. Использовать матрицу поворота на заданный угол относительно начала координат, см. (2.3).

3 шаг. Вектор переноса определяется из условия возвращения центра поворота в прежнее положение, на вектор $(x_м,y_м)$, см. (2.1).

Если исходную точку A с координатами (x,y) по дуге окружности с центром в точке C с координатами (x_c,y_c) поворачивают на угол ϕ, то координаты (x_1,y_1) повернутой точки могут быть записаны в следующем виде:

$$\begin{aligned} x_1 &= x_c + (x - x_c)\cos\varphi - (y - y_c)\sin\varphi; \\ y_1 &= y_c + (y - y_c)\cos\varphi + (x - x_c)\sin\varphi; \end{aligned} \qquad (2.8)$$

Пример процедуры *определяет координаты точки (x,y) после поворота заданной точки (x_n,y_n) относительно центра (x_c,y_c) на угол φ:*

```
procedure rotor(xn,yn:integer;var x,y:integer);
  begin
x1 = xc + (x-xc)·cosϕ- (y-yc)·sinϕ;
y1 = yc + (y-yc)·cosϕ+ (x-xc)·sinϕ
  end;
```

2.5. Аффинные преобразования составных графических объектов. Композиция преобразований

Перенос, масштабирование и поворот графических объектов эквивалентен переносу, масштабированию и повороту всех точек рисунка и последующему высвечиванию соединяющих их линий.

Для фигур, обладающих симметрией, или с границами, вычисляемыми с помощью уравнений, при перечисленных аффинных преобразованиях изображений нет необходимости осуществлять их для всех точек. Например, для переноса окружности достаточно перенести центр и вычертить ее, зная радиус.

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. Например, Перенос Поворот, Масштабирование (однородное $k_x=k_y$).

2.6. Аффинные преобразования в пространстве

В трехмерном случае (3D) (3-dimension) для введения однородных координат поступим аналогично тому, как это было сделано в 2D. Заменим координатную тройку (x, y, z), задающую точку в пространстве, на четверку чисел $(x, y, z, 1)$, в общем виде на четверку $(h_x, h_y, h_z, h), \quad h \neq 0$.

Каждая точка пространства (кроме начальной точки О) может быть задана четверкой одновременно не равных нулю чисел; эта четверка чисел определена однозначно с точность до общего множителя. Предложенный переход к новому способу задания точек дает возможность воспользоваться матричной записью и в более сложных, трехмерных задачах.

Любое аффинное преобразование в трехмерном пространстве может быть представлено в виде суперпозиции вращений, растяжений, отражений и переносов. Поэтому вполне уместно сначала подробно описать матрицы именно этих преобразований (ясно, что в данном случае порядок матриц должен быть равен четырем).

2.6.1. Матрицы вращения в пространстве

Матрица вращения вокруг оси абсцисс на угол φ относительно начала координат:

$$[R_x] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\varphi & \sin\varphi & 0 \\ 0 & -\sin\varphi & \cos\varphi & \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (2.9)$$

Матрица вращения вокруг оси ординат на угол ψ относительно начала координат:

$$[R_y] = \begin{bmatrix} \cos\psi & 0 & -\sin\psi & 0 \\ 0 & 1 & 0 & 0 \\ \sin\psi & 0 & \cos\psi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (2.10)$$

Матрица вращения вокруг оси аппликат на угол χ относительно начала координат:

$$[R_z] = \begin{bmatrix} \cos\chi & \sin\chi & 0 & 0 \\ -\sin\chi & \cos\chi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (2.11)$$

2.6.2.Матрица растяжения (сжатия):

Матрица растяжения (сжатия) относительно начала координат:

$$[D]=\begin{bmatrix}\alpha & 0 & 0 & 0\\ 0 & \beta & 0 & 0\\ 0 & 0 & \gamma & 0\\ 0 & 0 & 0 & 1\end{bmatrix} \qquad (2.12)$$

где $\alpha > 0$ – коэффициент растяжения (сжатия) вдоль оси абсцисс; $\beta > 0$ – коэффициент растяжения (сжатия) вдоль оси ординат; $\gamma > 0$ – коэффициент растяжения (сжатия) вдоль оси аппликат.

2.6.3. Матрицы отражения точки относительно координатных плоскостей

Матрица отражения относительно плоскости ***xy***:

$$[M_z]=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & -1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix} \qquad (2.13)$$

Матрица отражения относительно плоскости *yz*:

$$[M_x]=\begin{bmatrix}-1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix} \qquad (2.14)$$

Матрица отражения относительно плоскости *zx*:

$$[M_y]=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & -1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix} \qquad (2.15)$$

2.6.4. Матрица переноса точки в пространстве

Матрица переноса (здесь (λ, μ, ν) – вектор переноса):

$$[T]=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ \lambda & \mu & \nu & 1\end{bmatrix} \qquad (2.16)$$

Пример процедуры *поворота точки м(x,y,z) в пространстве относительно начала координат(0,0,0) на угол f вокруг оси абсцисс и вокруг оси ординат на угол s:*

```
procedure rtr(xn,yn,zn:integer;var x,y,z:integer);
  begin
  x:=round(xn*cos(f)-zn*sin(f)*cos(s)+yn*sin(f)*sin(s));
  y:=round(yn*cos(s)+zn*sin(s));
  z:=round(xn*sin(f)-yn*sin*cos(f)+zn*cos(s)*cos(f));
  end;
```

3. ДЕЛОВАЯ ГРАФИКА

Деловая графика использует основное достоинство машинной графики — наглядное представление информации, применяется в инженерной практике при выдаче результатов расчетов, экспериментов.

3.1. Построение графика функции

Для построения графика функции необходимо иметь таблицу (массив) значений аргумента и таблицу соответствующих значений функции. Каждой паре чисел из этих таблиц на графике соответствует точка на экране. Высветив эти точки, получим совокупность точек точечный график. Соединив полученные точки отрезками прямых или кривых, получим непрерывную кривую.

При отображении точек графика на экране необходимо выполнить операцию масштабирования, то есть сопоставить каждой паре значений аргумента и функции точку на экране.

Масштаб показывает, скольким точкам растра на экране соответствует единица физической величины. Масштабные коэффициенты для координатных осей вычисляются из следующих выражений:

$$cx = \frac{X_k - X_n}{X_{max} - X_{min}},$$
$$cy = \frac{Y_k - Y_n}{Y_{max} - Y_{min}}, \qquad (3.1)$$

где cx – масштабный коэффициент по оси абсцисс X;

cy – масштабный коэффициент по оси ординат Y;

X_n, Y_n – координаты верхнего левого угла поля вывода графика;

X_k, Y_k – координаты нижнего правого угла поля вывода графика;

X_{min}, X_{max} – минимальное и максимальное значения аргумента;

Y_{min}, Y_{max} – минимальное и максимальное значения функции.

При определении позиции экрана для очередной точки графика следует воспользоваться следующими формулами:

а) на оси абсцисс:

$$kx = round((x - X_{min}) \cdot cx) + X_n, \qquad (3.2)$$

где kx - номер позиции в строке (вдоль оси абсцисс);

x – текущее значение аргумента функции;

б) на оси ординат:

$$ky = round((Y_{max} - y) \cdot cy) + Y_n, \qquad (3.3)$$

Соотношение (3.2) получено из очевидной пропорции: всему диапазону X_{max} -X_{min} соответствует диапазон на экране величиной (X_k - X_n), а диапазону (x - X_{min}) — подлежащий определению диапазон координат устройства. К полученному интервалу координат устройства необходимо прибавить координату точки начала отсчета, т.е. X_n.

Помимо собственно кривой обычно на поле вывода наносят координатную сетку, представляющую собой вертикальные и горизонтальные линии,

проводимые с определенным шагом. Координатные оси оцифровывают, причем оцифровку наносят рядом с линией сетки.

3.2. Построение гистограммы, круговой диаграммы

Гистограмма является часто встречающейся формой представления информации. Гистограмма представляет собой совокупность смежных прямоугольников (столбиков), построенных на одной прямой линии. Высота каждого прямоугольника пропорциональна величине отображаемой функции (рис.1.). Соответствующие значения экранных координат по заданным значениям функции можно вычислить с помощью формул (3.1), (3.2).

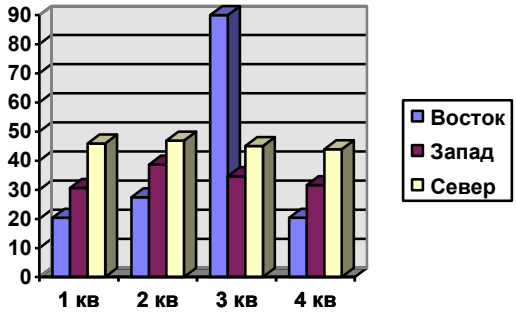

Рис.1. Сравнительная гистограмма, ориентированная вертикально

При построении гистограммы в качестве исходных данных обычно задается массив значений некоторой величины (функции), который надо представить в виде столбчатой диаграммы.

В круговой диаграмме в отличие от гистограммы изображаемые величины отображаются не в виде столбиков, а в виде круговых секторов (рис.2).

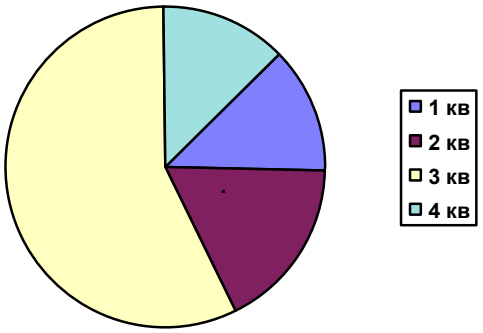

Рис.2. Круговая диаграмма

В качестве исходных данных при построении диаграммы задается массив значений рассматриваемой величины (функции). Местоположение и размер диаграммы удобно задать координатами центра диаграммы и величиной ее радиуса. При построении круговой диаграммы надо иметь в виду, что графические системы обычно позволяют вычертить дугу или сектор круга с начальным и конечным углами, задаваемыми целыми значениями градусов. Поэтому вычертить сектор с центральным углом меньше одного градуса

нельзя, а дробные значения градусов надо округлять до целого, что вносит некоторую погрешность.

Круговые диаграммы наиболее удобны для представления информации в процентном соотношении. В этом случае сумма всех значений принимается за 100 %, чему соответствует сектор с углом в 360^0 (т.е. полная дуга). Для вычисления величины центрального угла сектора, соответствующего очередному значению функции, следует использовать следующее выражение:

$$\alpha(i) = \frac{f(i)}{\sum_{j=1}^{n} f(j)} 360, \qquad (3.4)$$

где $\alpha(i)$ – величина центрального угла ***i***-го сектора, ***i=1,...,n,***

f(i) – i-е значение функции;

n – общее количество значений функции.

Начальный угол сектора представляет собой сумму центральных углов всех предшествующих ему секторов, может быть определен из выражения (3.3.1):

$$\alpha_n(i) = \sum_{j=1}^{i-1} \alpha(j), \qquad (3.5)$$

где $\alpha_n(i)$ – начальный угол ***i-го*** сектора.

Конечный угол сектора представляет собой сумму центральных углов всех предшествующих ему секторов, а также текущего сектора, поэтому он определяется следующим образом:

$$\alpha_k(i) = \sum_{j=1}^{i} \alpha(j), \qquad (3.6)$$

где $\alpha_k(i)$ – конечный угол ***i-го*** сектора.

И для последнего сектора:

$$\alpha(n) = 360 - \sum_{j=1}^{n-1} \alpha(j). \qquad (3.7)$$

4. ВИДЫ ПРОЕКТИРОВАНИЯ

Реальные объекты существуют в трехмерном пространстве, поэтому кроме длины и высоты они имеют и толщину. Когда изображение объекта воспроизводится на плоском (двумерном) экране, можно либо игнорировать его толщину, либо спроецировать объекты на экран так, чтобы ощущалась глубина изображения (рис.3).

Значения координаты ***z*** равны нулю на поверхности экрана, положительны в глубине экрана и отрицательны перед экраном. Положения точки в пространстве определяются тремя координатами ***(x,y,z)*** – положение на экране определяется координатами ***x*** и ***y***, а координата ***z*** указывает глубину расположения точки относительно плоского экрана. Координатой ***z*** можно пользоваться для получения различных видов предмета, для определения невидимых линий и поверхностей и для построения изображения с использованием перспективы.

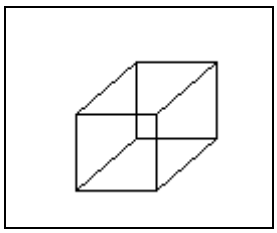

Рис.3. Изображение трехмерного
объекта на плоскости

Для получения проекции объекта на картинную плоскость надо провести через каждую точку объекта прямую из заданного проектирующего пучка и найти точку пересечения этой прямой с плоскостью изображения.

В компьютерной графике используются два основных вида проектирования.

1) При параллельном центр (несобственного) пучка находится в бесконечности (рис. 4).

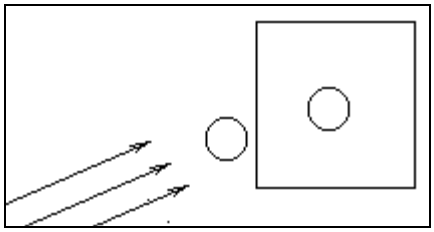

Рис. 4. Параллельное проектирование

2) При центральном проектировании все прямые исходят из одной точки – центра собственного пучка (рис.5).

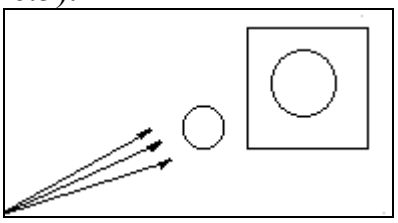

Рис. 5. Перспективное проектирование

4.1. Параллельные проекции

Ортографическая проекция — картинная плоскость совпадает с одной из координатных плоскостей или параллельна ей.

Матрица проектирования вдоль оси X на плоскость YZ:

$$\boldsymbol{p}_x = \begin{pmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{pmatrix}, \qquad (4.1)$$

аналогично – вдоль осей Y и Z.

Перемножив исходные координаты объекта на матрицу P_x и соединив их, получим вид объекта сбоку (справа или слева).

Если плоскость проектирования параллельна координатной плоскости, необходимо умножить матрицу P_x на матрицу сдвига. Получаем:

$$p_x = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ p & 0 & 0 & 1 \end{pmatrix}. \qquad (4.2)$$

Аналогично записываются матрицы проектирования вдоль других координатных осей.

Аксонометрическая проекция — проектирующие прямые перпендикулярны картинной плоскости. В соответствии со взаимным расположением картинной плоскости и координатных осей различают 3 вида: триметрия, диметрия и изометрия. Каждый из трех видов проекции получается комбинацией поворотов, а затем ортографическое проектирование:

•***триметрия*** – нормальный вектор картинной плоскости образует с координатными осями попарно различные углы. При повороте на угол f относительно оси ординат, на угол s вокруг оси абсцисс и последующего ортогонального проектирования вдоль оси аппликат получим матрицу:

$$M = \begin{pmatrix} \cos f & 0 & -\sin f & 0 \\ 0 & 1 & 0 & 0 \\ \sin f & 0 & \cos f & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos s & \sin s & 0 \\ 0 & -\sin s & \cos s & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} =$$

$$= \begin{pmatrix} \cos(f) & \sin(s)\sin(f) & 0 & 0 \\ 0 & \cos(s) & 0 & 0 \\ \sin(f) & -\sin(f)\cos(s) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad (4.3)$$

•***диметрия*** – два угла между нормалью картинной плоскости и координатными осями равны, матрицу преобразования можно получить, подставив соотношение $\sin^2(f) = tg^2(s)$ в (4.3).

•***изометрия*** – все три угла между нормалью картинной плоскости и координатными осями равны, матрицу преобразования можно получить, подставив соотношения $\sin^2(f) = \frac{1}{3}$ и $\sin^2(f) = \frac{1}{2}$ в (4.3).

4.2. Перспективные проекции

Предположим, что центр проектирования лежит на оси ***Z*** в точке ***C(0,0,c)*** и плоскость проектирования совпадает с плоскостью XY, тогда для произвольной точки ***M(x,y,z)*** координаты ***M*(x*,y*,z*)***, полученные после перспективного проектирования, находятся следующим образом:

$$X^* = \frac{1}{1-\frac{z}{c}}x, \quad Y^* = \frac{1}{1-\frac{z}{c}}y, \quad Z^* = 0. \qquad (4.4)$$

Проектирующие параллельные линии сходятся в одной точке, которая называется точкой схода. Перспективная проекция различается по количеству точек схода: одноточечная, двухточечная, трехточечная (рис.6).

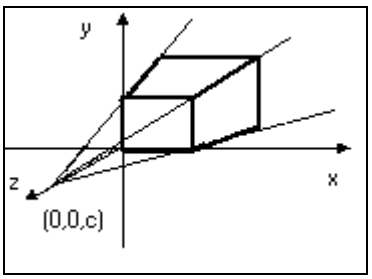

Рис.6. Перспективная проекция

Матрица перспективного преобразования Q с центром проектирования в точке C(0,0,c):

$$Q = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1/c \\ 0 & 0 & 0 & 1 \end{pmatrix}. \qquad (4.5)$$

В общем случае, когда координатные оси не параллельны картинной плоскости, таких точек три, матрица соответствующего преобразования выглядит так:

$$Q = \begin{pmatrix} 1 & 0 & 0 & -1/a \\ 0 & 1 & 0 & -1/b \\ 0 & 0 & 1 & -1/c \\ 0 & 0 & 0 & 1 \end{pmatrix}. \qquad (4.6)$$

Точки ***(-a,0,0) , (0,-b,0) и (0,0,-c)*** есть главные ***точки схода***.

Матрица перспективного преобразования является невырожденной, поэтому для получения перспективной проекции трехмерного объекта на плоскости нужно матрицу Q перемножить с матрицей параллельного проектирования P_z .

Пример процедуры *одноточечного перспективного преобразования точки M(x,y,z) в точку* $M(x_1,y_1)$ *с проектированием на картинную плоскость, совпадающую с плоскостью OXY, с точкой схода (0,0,-c):*

```
procedure perspct(x,y,z:integer;var x,y,z:integer);
  begin
  x1:=round(x/(1-z/c))+getmaxx div 2;
  y1:=round(y/(1-z/c))+getmaxy div 2;
  end;
```

5. РАСТРОВЫЕ АЛГОРИТМЫ

5.1.Основные понятия

Растр – двумерный целочисленный массив точек, упорядоченный в строки и столбцы на экране дисплея.

Алгоритм считается растровым, если он использует тот факт, что экран представляет растровую плоскость. Основным достоинством растровых алгоритмов является то, что при вычислениях используются только целые числа, на вычисление чисел с плавающей точкой затрачивается гораздо больше времени, следовательно, растровые алгоритмы являются более быстродействующими.

Растровые данные — представление изображения в виде двумерной целочисленной матрицы дискретных точек (пикселов), каждая из которых имеет свою яркость и цвет.

Связность определяет путь по целочисленной решетке, возможность соединения двух пикселов растровой линии, то есть кривую можно описать последовательным набором пикселов на всем ее протяжении от начальной точки до конечной, вычисляя на каждом шаге направление элементарного движения от текущей точки к следующей за ней. Это дает возможность увеличить быстродействие алгоритма за счет использования операций сложения и вычитания, исключая другие операции: умножения, деления и т.д.

Введем понятия 4- и 8-связности:

а) 4-связность, сильносвязный путь, когда пиксель считается соседним, если либо их x-координаты, либо y-координаты отличаются на единицу (движения в четырех направлениях) (рис.7).

$$|x_1 - x_2| + |y_1 - y_2| <= 1, \qquad (5.1)$$

б) 8-связность, слабосвязный путь на плоскости, если x и y-координаты отличаются не более чем на единицу (движение в восьми направлениях) (рис.8):

$$|x_1 - x_2| <= 1, \ |y_1 - y_2| <= 1. \qquad (5.2)$$

Рис.7. Направления шагов по 4-связному пути

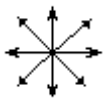

Рис.8. Направления шагов по 8-связному пути

Простой кривой на плоскости называется множество, у которого все точки имеют ровно двух соседей и две точки имеют ровно по одному соседу. Простой замкнутой кривой на плоскости называется множество, у которого все точки имеют ровно двух соседей. В качестве линии на растровой сетке выступает набор пикселов $p_1, p_2, \ldots, p_n$, где любые два пиксела p_i, p_{i+1} являются соседними.

5.2. Построение отрезка

Растровое представление отрезка не является единственным, возможны различные способы. В качестве примера приведем алгоритм построения 4-связного отрезка.

Даны точки $A(x_1,y_1)$ и $B(x_2,y_2)$, где $0<=x_1<x_2$, $0<=y_1<y_2$. Построить 4-связную линию. На каждом шаге элементарного движения от точки к точке нужно определять направления шага. Введем переменную ***factor***, учитывающую отклонения по направлениям x и y.

Алгоритм *построения 4-связной линии*

Начало
factor=0
Если factor > 0,
  тогда factor := factor - dx
  {двигаемся вдоль направления оси OY} y=y+1,

иначе

factor:=factor+dy

{двигаемся вдоль направления оси OX} x=x+1.

Для построения 8-связной линии на каждом этапе элементарного движения нужно изменять значения x-координаты ***x:=x+1*** и в зависимости от переменной ***factor*** изменять значения y-координаты.

Можно получить общую версию алгоритма, для этого необходимо убрать ограничения $0<=x_1<=x_2$, $0<=y_1<=y_2$, т.е. учесть ориентацию отрезка относительно положительных направлений осей координат.

5.3. Заполнение сплошных областей

Среди алгоритмов растровой графики о заполнении внутренности сплошной области можно выделить два типа:

1) Заполнение внутренности области, ограниченной замкнутым контуром (по признаку связности).

2) Заполнение внутренности многоугольника, заданного своими вершинами или ребрами (по признаку четности).

Рассмотрим алгоритм заполнения области по критерию связности. Пусть дана область, ограниченная набором пикселов заданного цвета, и точка ***(x,y),*** принадлежащая внутренности этой области.

Вводим переменные:

bordercolor – в которой хранится значение цвета контура (бордюра),

color – цвет области,

c:=getpixel(x,y) – функция возвращает цвет пиксела в точке с координатами x и y.

1 шаг. Для заданной точки ***(x,y)*** определяем и заполняем максимальный отрезок (x_1,x_2), содержащий эту точку и лежащий внутри области.

2 шаг. Проверяем отрезки, лежащие выше и ниже граничных точек x_1 и x_2. Если такие пикселя находятся, то функция рекурсивно вызывается для их обработки.

Алгоритм *заполняет любую область, в том числе и такую, в которой присутствует отверстие.*

Начало.

1)Инициализируем стек, помещая в него затравочный пиксель.

2) Пока стек не пуст:

а) Извлекаем пиксель ***(x,y)*** из стека.

б) Заполняем максимально возможный интервал, в котором находится пиксель, вправо и влево вплоть до достижения граничных пикселов.

с) Запоминаем крайнюю левую и крайнюю правую абсциссы заполненного интервала.

д) В соседних строках под и над интервалом находим не заполненные к настоящему моменту внутренние пиксели области, которые объединены в интервалы, а в каждом из этих интервалов находим крайние правые и левые пиксели. Каждый из этих пикселов вносим в стек в качестве затравочного.

Конец.

Алгоритм *заполнения области по критерию четности, для этого определим тест принадлежности точки многоугольнику*.

Пусть задан многоугольник, ограниченный несамопересекающейся замкнутой ломаной $p_1,p_2,...,p_n$, и некоторая точка ***(x,y)*** и требуется определить, содержится ли эта точка внутри многоугольника или нет.

Начало. Рассмотрим все отрезки ломаной, кроме горизонтальных, на пересечения с горизонтальным лучом, выходящим из заданной точки. Если число пересечений четно, следовательно, точка находится вне многоугольника, если число пересечений нечетно, точка принадлежит многоугольнику. При попадании луча в вершину пересечение засчитывается только с теми отрезками ломаной, для которых она является верхней.

Введем переменную ***S*** – для подсчета количества пересечений горизонтального луча с ребрами многоугольника.

S=0;

Для каждого ребра выполним проверку:

если { не горизонтально },
то если { ребро пересекается с горизонтальным лучом или координата точки пересечения равна максимальной точке ребра},
то ***S= S+1***;

После проверки всех ребер многоугольника на пересечения необходимо проверить четность переменной ***S***.

Если ***S*** – четно, то точка не принадлежит многоугольнику.

Таким образом, использование вышеописанного теста определяет принадлежность точки многоугольнику.

Конец.

Для получения более эффективных алгоритмов можно ограничить область прямоугольником, описанным около многоугольника. Все стороны рассортировываются в соответствии со значением y_i и x_i в порядке убывания. Поочередно обрабатываются парные стороны с одинаковым значением y. Используя критерий четности, определяем отрезки сканирующей прямой, принадлежащей многоугольнику. При перемещении сканирующей прямой снизу вверх на пересечения проверяются только те ребра, значения максимальных ординат которых меньше ординаты сканирующей прямой. Т. о. поддерживается информация о множестве "активных" ребер, пересекающихся со сканирующей прямой, что сокращает число операций вычисления и влияет на быстродействие алгоритма.

6. ИНТЕРПОЛЯЦИЯ СПЛАЙНАМИ

Простым сплайном называется кусочно-полиномиальная функция (1, 2, 3 степени и выше), используемая для представления сложных гладких кривых. Сплайны используются для решения задачи интерполяции: по заданному набору точек на плоскости или в пространстве построить кривую, либо проходящую через все точки, либо проходящую вблизи от этих точек (задача сглаживания).

6.1. Сплайн – функции

Типичная задача интерполяции состоит в восстановлении с той или иной точностью функции $f(x)$ на заданном отрезке $[a,b]$ по таблице чисел (x_i, y_i), $i = 0,\ldots N$, где $y_i = f(x_i)$ и точки x_i образуют упорядоченную последовательность $a = x_0 < x_1 < \ldots < x_N = b$. Весьма эффективным методом решения этой задачи является построение интерполяционного кубического сплайна (рис. 9).

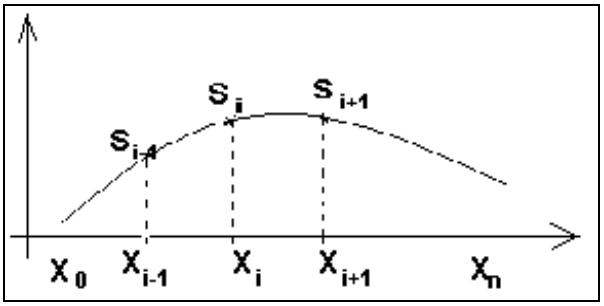

Рис. 9. Сплайн-функция: $x_0,\ldots,x_n$ - узлы интерполяции

$S_i(x)$ – интерполяционный кубический сплайн

Определение 1. Функция $S(x) \in C^2[a,b]$ называется интерполяционным кубическим сплайном, если:

1) на каждом отрезке $[x_i, x_{i+1}]$ функция $S(x)$ является кубическим многочленом

$$S(x) \equiv S_i(x) = a_{i,0} + a_{i,1}(x - x_i) + a_{i,2}(x - x_i)^2 + a_{i,3}(x - x_i)^3$$ для $x \in [x_i, x_{i+1}]$, $i = 0,\ldots, N-1$;

2) соседние многочлены гладко состыкованы между собой

$$S_{i-1}^{(r)}(x_i - 0) = S_i^{(r)}(x_i + 0),\ i = 1,\ldots, N-1,\ r = 0, 1, 2;$$

3) выполняются условия интерполяции

$$S(x_i) = y_i,\ i = 0,\ldots, N.$$

Согласно определению интерполяционный кубический сплайн $S(x)$ является упорядоченным набором кубических многочленов, гладко состыкованных между собой так, что они образуют дважды непрерывно дифференцируемую функцию. Точки стыковки соседних многочленов x_i, $i = 1,\ldots, N-1$ называются узлами сплайна. Они могут не совпадать с точками интерполяции. Узлы сплайна могут также иметь различную кратность в зависимости от числа сопрягаемых производных. В частности, узел x_i, имеет кратность k_i $(0 \le k_i \le 3)$, если производные соседних многочленов сопрягаются в этом узле до порядка $3 - k_i$. Мы будем рассматривать только простые узлы (кратности 1).

Каждый из составляющих сплайн N многочленов имеет 4 коэффициента, что дает в совокупности $4N$ параметров. Из этого числа следует вычесть $3(N-1)$ ограничений гладкости и $N+1$ интерполяционных условий. Остающиеся свободными два параметра $(4N - 3(N-1) - N - 1 = 2)$ обычно определяются с помощью ограничений на значениях сплайна и его

производных на концах отрезка $[a,b]$ (или вблизи концов). Эти ограничения называются краевыми условиями. Существует несколько различных видов краевых условий, из которых наиболее употребительными являются следующие типы:

1. Ограничения на значения первой производной сплайна:
$$S'(x_0) = y'_0 \text{ и } S'(x_N) = y'_N;$$
2. Ограничения на значения второй производной сплайна:
$$S''(x_0) = y''_0 \text{ и } S''(x_N) = y''_N;$$
3. Периодические краевые условия:
$$S^{(r)}(x_0) = S^{(r)}(x_N),\ r = 0, 1, 2;$$
4. Совпадение ближайших к концам отрезка $[a,b]$ соседних многочленов:

$S_0(x) \equiv S_1(x)$ и $S_{N-1}(x) \equiv S_N(x)$, т.е. $S'''(x_i - 0) = S'''(x_i + 0)$, $i = 1, N-1$.

Выполнения периодических краевых условий естественно требовать в предположении, что интерполируемая функция $f(x)$ — периодическая с периодом $b-a$.

Полученную систему уравнений относительно коэффициентов можно решить методом трехточечной прогонки, затем построить сплайн-функцию, как обычную кривую.

Метод трехточечной прогонки

Рассмотрим эффективный метод решения систем линейных уравнений, имеющих трехдиагональные матрицы с диагональным преобладанием. Приводимый ниже алгоритм носит название метода трехточечной прогонки и является специальным случаем метода исключения Гаусса. Имея в виду системы линейных уравнений, возникающие при построении интерполяционного кубического сплайна с краевыми условиями типов 1, 2 или 4, рассмотрим следующую линейную систему:

$$\begin{bmatrix} b_1 & c_1 & 0 & 0 & \dots & 0 \\ a_2 & b_2 & c_2 & 0 & \dots & 0 \\ 0 & a_3 & b_3 & c_3 & \dots & 0 \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & a_{n-1} & b_{n-1} & c_{n-1} \\ 0 & \dots & 0 & 0 & a_n & b_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_{n-1} \\ d_n \end{bmatrix}. \quad (6.1)$$

Чтобы начать исключение, разделим первое уравнение этой системы на диагональный элемент b_1 и используем обозначения $p_1 = \frac{c_1}{b_1}$ и $q_1 = \frac{d_1}{b_1}$. Предположим, что мы исключили все нулевые поддиагональные элементы в первых $i-1$ строках. В этом случае система преобразуется к виду.

$$\begin{bmatrix} 1 & p_1 & 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 1 & p_2 & 0 & 0 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & & & & \vdots \\ 0 & \dots & 0 & 1 & p_{i-1} & 0 & \dots & 0 \\ 0 & \dots & 0 & a_i & b_i & c_i & \dots & 0 \\ \vdots & & & & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & 0 & 0 & a_{n-1} & b_{n-1} & c_{n-1} \\ 0 & \dots & 0 & 0 & 0 & 0 & a_n & b_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{i-1} \\ x_i \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_{i-1} \\ d_i \\ \vdots \\ d_{n-1} \\ d_n \end{bmatrix}. \quad (6.2)$$

Теперь, чтобы исключить поддиагональный элемент a_i в i-й строке, умножим $(i-1)$- ю строку на a_i и вычтем ее из i-й строки. В результате i-я строка нашей системы преобразуется к виду

$$(b_i - a_i p_{i-1})x_i + c_i x_{i+1} = d_i - a_i q_{i-1}.$$

Чтобы получить единицу на главной диагонали матрицы, разделим i-ю строку на коэффициент $(b_i - a_i p_{i-1})$. В результате для элементов p_i и q_i в окончательном виде i-й строки получаем следующие формулы:

$$\begin{aligned} p_i &= \frac{c_i}{b_i - a_i p_{i-1}},\ i = 2,\dots,n-1, \quad p_1 = \frac{c_1}{b_1}, \\ q_i &= \frac{d_i - a_i q_{i-1}}{b_i - a_i p_{i-1}},\ i = 2,\dots,n, \qquad q_1 = \frac{d_1}{b_1}. \end{aligned} \quad (6.3)$$

Продолжая исключение, получаем систему уравнений с двухдиагональной матрицей вида

$$\begin{bmatrix} 1 & p_1 & 0 & 0 & \dots & 0 \\ 0 & 1 & p_2 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & & \vdots \\ 0 & \dots & 0 & 1 & p_{n-2} & 0 \\ 0 & \dots & 0 & 0 & 1 & p_{n-1} \\ 0 & \dots & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-2} \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_{n-2} \\ q_{n-1} \\ q_n \end{bmatrix}. \quad (6.4)$$

Это позволяет записать рекуррентные формулы для вычисления неизвестных x_i

$$\begin{aligned} x_n &= q_n, \\ x_i &= p_i x_{i+1} + q_i,\ i = n-1,\dots,1. \end{aligned} \quad (6.5)$$

Аналогичными свойствами обладает интерполяционная бикубическая сплайн-функция в пространстве.

Достоинства предложенного способа :

- для решения линейных систем, при построении сплайн-функций, существует много эффективных методов, к тому же эти системы достаточно просты;
- графики построенных сплайн-функций проходят через все заданные точки, полностью сохраняя первоначальную информацию.

Недостатки:

•изменение лишь одной точки, при описанном подходе, заставляет пересчитывать заново все коэффициенты;

•во многих задачах исходный набор точек задается приближенно и требование прохождения графика искомой функции через каждую точку этого набора оказывается излишним.

6.2 Сплайновые кривые

Кривой Безье, определяемой массивом $V=\{V_0,V_1,...,V_m\}$, называется кривая, определяемая векторным уравнением

$$r(t)=\sum_{i=0}^{m} C_m^i t^i(1-t)^{m-i}V_i, \quad 0\le t\le 1, \qquad (6.6)$$

где $C_m^i=\dfrac{m!}{i!(m-i)!}$ – коэффициенты в разложении бинома Ньютона (число сочетаний из m элементов по i). На рис. 10 представлены контрольная ломаная и кривая Безье, проходящие через четыре заданные вершины V_0, V_1, V_2, V_3.

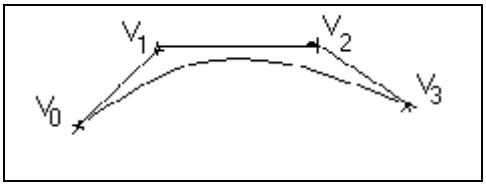

Рис.10. Кривая Безье

Кривая Безье обладает замечательными свойствами:

1) она является гладкой;

2) начинается в точке V_0 и заканчивается в точке V_m, касаясь при этом отрезков $V_0V_1, ..., V_{m-1}V_m$;

3) функциональные коэффициенты $C_m^i t^i(1-t)^{m-i}$ при вершинах $Vi, i=0,1,..., m$ являются универсальными многочленами Бернштейна; они неотрицательны, и их сумма равна единице:

$$\sum_{i=0}^{m} C_m^i t^i(1-t)^{m-i}=1.$$

Наряду с отмеченными достоинствами *кривые Безье обладают и определенными недостатками*. Основные недостатки элементарных кривых Безье:

1) степень функциональных коэффициентов напрямую связана с количеством точек в заданном наборе (на единицу меньше);

2) при добавлении хотя бы одной точки в заданный набор необходимо провести полный пересчет функциональных коэффициентов в параметрическом уравнении кривой;

3) изменение хотя бы одной точки приводит к заметному изменению вида всей кривой.

Составная кубическая кривая Безье представляет объединение элементарных кубических кривых Безье $\gamma_1,...,\gamma_m$, такая что $r_i(1)=r_{i+1}(0), i=0,..., m-1,$ где $r=r_i(t), 0<t<1,$ – параметрическое уравнение кривой γ_i. Т.о., конец предыдущей γi совпадал с началом последующей γ_{i+1}.

B-сплайновая кривая обладает всеми достоинствами кривой Безье, но лишена ее недостатков, представлена на рис 11.

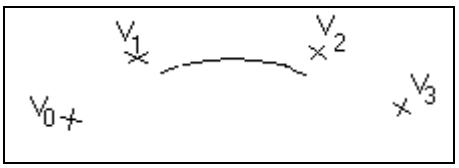

Рис.11. Элементарная B-сплайновая кривая

По заданному набору точек V_0,V_1,V_2,V_3 элементарная кубическая B-сплайновая кривая определяется при помощи векторного параметрического уравнения следующего вида:

$$r(t)=\frac{(1-t)^3}{6}V_0+\frac{3t^3-6t^2+4}{6}V_1+\frac{-3t^3+3t^2+3t+1}{6}V_2+\frac{t^3}{6}V_3 \quad (6.7)$$

где $0\le t\le 1$.

Элементарная B-сплайновая кривая, определенная на V_0,V_1,V_2,V_3 набору точек, обладает свойствами:

1) она является гладкой;

2) начинается вблизи точки V_1 и заканчивается вблизи V_2, касаясь при этом отрезка V_1V_2;

3) функциональные коэффициенты в уравнении, определяющем элементарную B-сплайновую кубическую кривую, неотрицательны, в сумме составляют единицу, универсальны (не зависят от конкретного расположения точек в заданной четверке);

4) степень функциональных коэффициентов не связана с количеством точек в заданном наборе;

5) при добавлении хотя бы одной точки в заданный набор нет необходимости проводить полный пересчет функциональных коэффициентов в параметрическом уравнении кривой;

6) изменение хотя бы одной точки не приводит к изменению вида всей кривой.

Составная кубическая B-сплайновая кривая, задаваемая параметрическим уравнением $r=r(t),\ 0\le t\le m-2$ и определяемая набором точек $V_0,V_1,...,V_m$ $(m>3)$, представляет собой объединение $m-2$ элементарных кубических B-сплайновых кривых $\gamma_3,...,\gamma_m$, описываемых уравнениями вида $r=r_i(t),\ i=1,...,m-2;\ i-1\le t\le i$, – параметрическое уравнение кривой γ_i.

Составная B-сплайновая кубическая кривая является гладкой кривой и лежит в объединении $m-2$ выпуклых оболочек, порожденных последовательными четверками точек заданного набора. Добавляя в исходный набор дополнительные точки, можно получить составную B-сплайновую кубическую кривую с разными свойствами.

7. ГЕОМЕТРИЧЕСКОЕ ИЗОБРАЖЕНИЕ ГРАФИКА ФУНКЦИИ ДВУХ ПЕРЕМЕННЫХ

Рассмотрим сначала задачу построения графика функции двух переменных $z=f(x,y)$ в виде сетки координатных линий $x=const$ и $y=const$.

Будем рассматривать параллельное проектирование, при котором проекцией вертикальной линии на картинной плоскости (экране) является

вертикальная линия. Легко убедиться в том, что в этом случае точка $p(x,y,z)$ переходит в точку $((p,e_1),(p,e_2))$ на картинной плоскости, где

$$\begin{aligned} e_1 &= (\cos\varphi, \sin\varphi, 0), \\ e_2 &= (\sin\varphi\sin\psi, -\cos\varphi\sin\psi, \cos\psi), \end{aligned} \tag{7.1}$$

а направление проектирования имеет вид

$$e_3 = (\sin\varphi\cos\psi, -\cos\varphi\cos\psi, -\sin\psi), \tag{7.2}$$

где $\varphi \in [0, 2\pi]$, $\psi \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$.

Рассмотрим сначала построение графика функции в виде набора линий, соответствующих постоянным значениям y, считая, что углы φ и ψ подобраны таким образом, что при $y_1 < y_2$ плоскость $y = y_1$ расположена ближе к картинной плоскости, чем плоскость $y = y_2$.

Заметим, что каждая линия семейства $z = f(x, y_i)$ лежит в своей плоскости $y = y_i$, причем все эти плоскости параллельны и, следовательно, не могут пересекаться. Из этого следует, что при $y_j > y_i$ линия $z = f(x, y_j)$ не может закрывать линию $z = f(x, y_i)$.

Тогда возможен следующий алгоритм построения графика функции $z = f(x, y)$: линии рисуются в порядке удаления (возрастания по y) и при рисовании очередной линии рисуется только та ее часть, которая не закрывается ранее нарисованными линиями.

Такой алгоритм называется методом плавающего горизонта.

Для определения частей линии $z = f(x, y_k)$, которые не закрывают ранее нарисованные линии, вводятся так называемые линии горизонта, или контурные линии.

Пусть проекцией линии $z = f(x, y_k)$ на картинную плоскость является линия $Y = Y_k(X)$, где (X, Y) — координаты на картинной плоскости, причем Y соответствует вертикальной координате. Контурные линии $Y^k_{max}(X)$ и $Y^k_{min}(X)$ определяется следующими соотношениями:

$$\begin{aligned} Y^k_{max}(X) &= \max_{1 \le i \le k-1} Y_i(X), \\ Y^k_{min}(X) &= \min_{1 \le i \le k-1} Y_i(X). \end{aligned} \tag{7.3}$$

На экране рисуются только те части линии $Y = Y_k(X)$, которые находятся выше линии $Y^k_{max}(X)$ или ниже линии $Y^k_{min}(X)$.

Одной из наиболее простых и эффективных реализаций данного метода является растровая реализация, при которой в области задания исходной функции вводится сетка:

$$\{(x_i, y_j),\ i = 1,\ldots,n_1,\ j = 1,\ldots,n_2\}$$

и каждая из линий $Y = Y_k(X)$ представляется в виде ломаной. Для рисования сегментов этой ломаной используется модифицированный алгоритм Брезенхейма, который перед выводом очередного пикселя сравнивает его

ординату с верхней и нижней контурными линиями, представляющими из себя в этом случае массивы значений ординат.

Рассмотрим теперь задачу построения полутонового изображения графика функции $z = f(x, y)$.

Как и ранее, введем сетку:

$$\{(x_i, y_j),\ i = 1,...,n_1,\ j = 1,...,n_2\}$$

и затем приблизим график функции набором треугольных граней с вершинами в точках $(x_i, y_j, f(x_i, y_j))$.

Для удаления невидимых граней воспользуемся аналогичным методом – упорядоченным выводом граней. Только в данном случае треугольники будем выводить не по мере удаления от картинной плоскости, а по мере их приближения, начиная с дальних и заканчивая ближними: треугольники, расположенные ближе к плоскости экрана, выводятся позже и закрывают собой невидимые части более дальних треугольных граней.

Для определения порядка, в котором должны выводиться грани, воспользуемся тем, что треугольники, лежащие в полосе

$$\{(x, y), y_i \le y \le y_{i+1}\},$$

не могут закрывать треугольники из полосы

$$\{(x, y), y_{i-1} \le y \le y_i\}.$$

Уравнение ***z = F(x,y)*** определяет некоторую поверхность. Пара значений ***x*** и ***y*** определяет на плоскости OXY точку ***P(x,y), z=F(x,y)*** - аппликату соответствующей точки ***M(x,y,z).***

Алгоритм *построения параллельных кривых с проектированием на плоскость OXY:*

1 шаг. Найти масштабные коэффициенты для того, чтобы изображение занимало экран полностью.

2 шаг. Для построения трехмерного объекта на плоскости необходимо получить его параллельную или перспективную проекцию.

3 шаг. Строим график функции одной переменной относительно x с фиксированным шагом от y_{min} к y_{max}. Т.о., получим изображение поверхности из параллельных линий.

Пример программы *построения изображения функции двух переменных, с помощью координационных линий (y=const):*

```
uses crt, graph;
const
otstup=30;
dy=4;
dx=2;
w=3.1459/180;
ph=45*w;
ps=45*w;
e:array [1..2,1..3]  of real =(cos(ph),sin(ps),0,sin(ps)*sin(ph),-
sin(ps)*cos(ph),cos(ps));
var
```

```
z,xl,x,y:real;
h,hx,xc,yc,x1,x2,y1,y2:integer;
  z2,z1:integer;
begin
{ инициализация графического режима}
xc:=getmaxx div 2;
yc:=getmaxy div 2 + 2*otstup;
{определение масштабных коэффициентов}
h:=yc div 3;
 y:=-h;
 xl:=xc div 2;
 repeat
 {получение параллельных прямых при фиксировании yi  }
 x:=-xl;
 z:=h*sin(w*sqrt(x*x+y*y));
 {проектирование на OXY}
 x1:=round(x*e[1,1]+y*e[1,2]+z*e[1,3]);
 z1:=round(x*e[2,1]+z*e[2,3]-y*e[2,2]);
 x:=x+dx;
 repeat
 z:=h*sin(w*sqrt(x*x+y*y));
 x2:=round(x+y);
 z2:=round(z-y+x);
 line(x1+xc,yc-z1,x2+xc,yc-z2);
 x1:=x2;
 z1:=z2;
 x:=x+dx;
 until x>xl;
 y:=y+dy;
 until y>h;
 readln;
 closegraph;
 end.
```

Результаты работы программы, поверхность, состоящая из координатных линий при фиксированных значения y_k функции $f(x,y)=sin(\sqrt{x^2+y^2})$ представлены на рис.12.

Аналогично строятся параллельные кривые при фиксированной координате x.

Алгоритм построения графика функции двух переменных, называемый методом плавающего горизонта, включает в себя анализ видимых и невидимых линий:

Пусть проекцией семейства линий $z=f(x,y_k)$ на картинную плоскость является $y=y_k(x)$, соответствующих постоянным значениям y, причем все эти плоскости параллельны и не пересекаются.

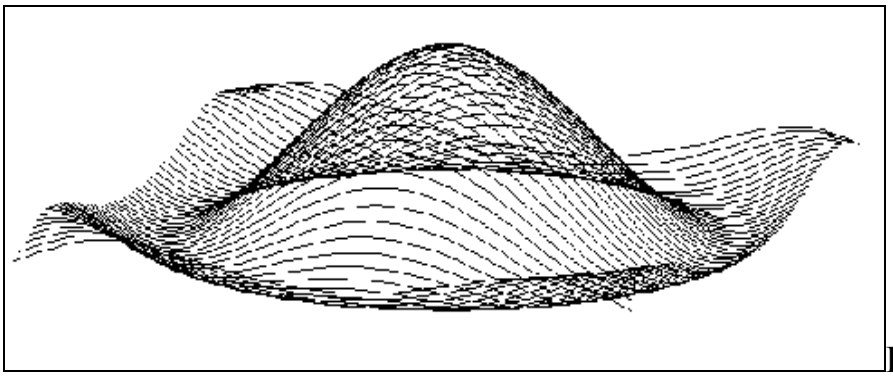

Рис.12. Геометрическое изображение функции двух переменых.

Пусть $y_1 < y_2 < ... < y_n$, тогда линии $z=f(x,y_k)$ рисуются в порядке удаления и при рисовании очередной линии рисуется только та часть, которая не закрывается ранее нарисованными контурными линиями.

Пусть проекцией линз $z=f(x,y_k)$ на картинную плоскость является линия $Y = Y^k(x)$ — соответствующая постоянным значениям $y=y_k$, где (X, Y) - координаты картинной плоскости.

Контурные линии определяются соотношением:

$$Y^k_{max}(x) = \max_{1<i<k-1} Y_i(x) \qquad (7.4)$$

$$Y^k_{min}(x) = \min_{1<i<k-1} Y_i(x).$$

На экране рисуются только те линии $Y=Y_k(x)$, которые находятся выше линии $Y^k_{max}(x)$ и ниже линии $Y^k_{min}(x)$, (рис. 12).

8. РАБОТА С ОСНОВНЫМИ ГРАФИЧЕСКИМИ УСТРОЙСТВАМИ

8.1. Основные операции работы с мышью

Если посмотреть на любую хорошо сделанную прикладную программу, то придется признать, что не менее половины всего объема программы служит именно для организации интерфейса ввод/вывод информации, работа с мышью, организация меню, реакция на ошибки и все, что относится к способу взаимодействия между программой и человеком.

Наиболее распространенным устройством ввода графической информации в ПЭВМ является мышь. Каждая мышь поставляется со своим драйвером - специальной программой - предоставляющей некоторый интерфейс прикладным программам. Работа с мышью реализуется через механизм прерываний. Приведем набор функций для работы с мышью в соответствии со стандартом фирмы Microsoft:

*а) **Функция** инициализации и проверки наличия мыши.*

```
function Resetmouse:boolean;
  var r:registers;
    begin
      r.ax:=0;
      intr($33,r);
      resetmouse:=r.ax=$ffff;
    end;
```

*б) **Процедура** высвечивания на экране курсора мыши.*

```
procedure showmouse;
    var r:registers;
     begin
       r.ax:=1;
         intr($33,r);
      end;
```

в) ***Процедура*** *удаления с экрана курсора мыши.*

```
procedure nidemouse;
   var r:registers;
    begin
     r.ax:=2;
       intr($33,r);
    end;
```

При работе с мышью следует иметь в виду, что выводить изображения поверх курсора мыши нельзя, необходимо убрать курсор с экрана, выполнить требуемый вывод и затем снова вывести курсор мыши на экран.

г) ***Процедура*** *чтения состояния мыши (ее координаты и нажатие кнопок).*

```
procedure readmouse(var x,y:integer; var l:boolean);
    var r:registers;
       begin
         r.ax:=3;
           intr($33,r);
             x:=r.cx;
               y:=r.dx;
                 l:=(r.bx and 1)<>0;
{                  p:=(r.bx and 2)<>0;}
{                  k:=(r.bx and 3)<>0;}
        end;
```

д) ***Процедура*** *передвижения курсора мыши в точку с заданными координатами.*

```
 procedure movemouse(x,y:integer);
   var r:registers;
    begin
      r.ax:=4;
      r.cx:=x;
      r.dx:=y;
      intr($33,r);
    end;
```

8.2. Организация простейшего меню

Организовать меню можно с помощью вышеизложенных функций и процедур.

1. Выполнить функцию инициализации и проверки наличия мыши.
2. Организовать бесконечный цикл с выходом по нажатию правой кнопки мыши.

3. Определить границы действия каждого пункта меню.

4. Выполнить процедуру чтения состояния мыши (ее координаты и нажатие кнопок). Если курсор мыши находится в определенной границе подменю и нажата левая кнопка, то выполняется действие выбранного подменю.

В основе организации интерфейса лежат несколько общих принципов:

- *графический режим;
- *представления ряда объектов пиктограммами;
- *использование указывающего устройства – мыши;
- *наглядность;
- *стандартизация всех основных действий и элементов, которые могут использоваться при конструировании прикладных программ, делая их похожими в обращении и облегчая процесс их написания.

Пример программы *организации простейшего меню*

```
USES crt,graph;
  var
  pr:integer;
  l,m,r:boolean;
  x,y:integer;
 procedure {инициализация графики}
{процедуры работы с устройством ввода мышью}
BEGIN
 {инициализация графики}
  pr:=0;   {организация бесконечного цикла}
while pr=0 do
begin
   rectangle(10,50,100,70);
   outtextxy (20,60,'AAAAA');          {пункты меню}
   rectangle(300,50,390,70);
    outtextxy(310,60,'Выход');
showmouse;
readmouse(x,y,l,m,r);
if  l and  (x<=100) and (x>=10) and (y<=70) and (y>=50)
 then begin
  cleardevice;
  outtextxy(200,200,'AAAAA-Нажмите <Enter>');
  readln;
end;
if r or l and  (x<=390) and (x>=300) and (y<=70) and (y>=50)
 then begin
 outtextxy(200,200,'Выход');
pr:=1;
end;  end;
{закрытия графического режима}
```

```
closegraph;
END.
```

9. ЗАДАНИЯ НА ЛАБОРАТОРНЫЙ ПРАКТИКУМ

ЛАБОРАТОРНАЯ РАБОТА № 1

Создание изображений плоских геометрических объектов

Цель работы – научиться составлять программы, осуществляющие аффинные преобразования плоских геометрических фигур.

Необходимые сведения из теории *(см. 1.1, 1.2, 1.3, 2.1, 2.2, 2.3,2.4,2.5, ПРИЛОЖЕНИЕ 1):*

1) Формулы преобразования из прямоугольной системы координат в экранные координаты.
2) Понятие однородных координат.
3) Базовые операции аффинных преобразований точки на плоскости.
4) Алгоритмы смещения, поворота на заданный угол, масштабирования изображения относительно выбранного центра.

Задание

Построить геометрические фигуры с использованием базовых операций аффинных преобразований.

Порядок выполнения работы

1. В центре экрана, выбрав опорную точку, построить плоское изображение фигуры из таблицы 1 по заданным параметрам (A, B – стороны, D1,D2 – диагонали, R – радиус описанной окружности).
2. Перенести полученное изображение в центр одной из четвертей экрана (правое - верхнее, левое - верхнее, левое - нижнее, правое - нижнее).
3. Относительно центра экрана произвести масштабирование изображения (четные варианты – kx=2, ky=2, нечетные варианты – kx=0.5, ky=0.5).
4. Относительно центра экрана произвести поворот изображения на заданный угол из таблицы 1 по варианту.

Таблица 1. Параметры плоских геометрических фигур по вариантам

Вариант	Фигура	Длина стороны	Угол поворота
1.	Квадрат	A=100	30
2	Ромб	D1=100, D2=150	45
3	Треугольник	A=100	60
4	Прямоугольник	A=100, B=150	30
5	Квадрат	A=75	45
6	Ромб	D1=150, D2=100	60
7	Прям	A=150,	30

	оугольник	B=100	
8	Треуг ольник	A=150	45
9	Ромб	D1=150, D2=100	60
10	Квадр ат	A=150	30
11	Прави льный 5-угольник	R=100	60
12	Прави льный 6-угольник	R=100	30
13	Прави льный 7-угольник	R=100	60
14	Прави льный 8-угольник	R=100	60

ЛАБОРАТОРНАЯ РАБОТА № 2

Создание изображений плоских кривых

Цель работы – научиться составлять программы, осуществляющие аффинные преобразования плоских кривых.

Необходимые сведения из теории *(см. 1.1, 1.2, 1.4, 2.1, 2.2, 2.3,2.4,2.5, ПРИЛОЖЕНИЕ 1):*

1) Представление уравнения кривой в различных формах в параметрическом виде, в полярной системе координат.

2) Алгоритм построения кривых на плоскости с использованием радиуса кривизны.

3) Базовые операции аффинных преобразований точки на плоскости.

4) Алгоритмы смещения, поворота на заданный угол, масштабирования изображения относительно выбранного центра.

Задание

Построить кривые с использованием базовых операций аффинных преобразований.

Порядок выполнения работы

1. Определить аналитически радиус кривизны.
2. В центре экрана построить изображение плоской кривой, заданной уравнением (из таблицы 2) по варианту.
3. Выполнить аффинные преобразования для заданной кривой, пункты 2, 3, 4 лабораторной работы N 1.

Таблица 2. Уравнения кривых по вариантам

Вар иант	Уравнение кривой

1	$X=b\cdot\cos^3(t)$, $y=b\cdot\sin^3(t)$
2	$x = t^3 + 3t + 1$, $y = t^3 - 3t + 1$
3	$x = 2a\cdot\cos(t) - a\cdot\cos(2t)$, $y=2a\cdot\sin(t) - a\cdot\sin(2t)$ (картоида)
4	$x = a\cdot(t^2 - 1) / (t^2 + 1)$, $y = a\cdot t\cdot(t^2 - 1) / (t^2 + 1)$ (строфоида)
5	$x = a\cdot(2)^{1/2}(t^3 + t) / (t^4 + 1)$, $y = a\cdot(2)^{1/2}(t - t^3) / (t^4 + 1)$ (лемниската Бернулли)
6	$x =3t / (t^3 + 1)$, $y = 3t^2 / (t^3 + 1)$ (декартов мост)
7	$r = -a\cdot\cos(2\varphi)/\cos(\varphi)$, полярная система координат (строфоида)
8	$r^2 = 2(a^2)\cos(\varphi)$ (лемниската)
9	$ay^2=x(x-a)^2$ (петлевая парабола)
10	$r = - a\cdot\sin^2(\varphi)/ \cos(\varphi)$ (циссоида)
11	$r = -a\cdot\cos(\varphi) + b$ (улитка Паскаля)
12	$r = -\varphi k$ (спираль Архимеда)
13	$r = a\cdot e^{\varphi k}$ (логарифмическая спираль)
14	$X = -pi\cdot r + \alpha\cdot r - r\cdot\sin(\alpha)$, $y = r-r\cdot\cos(\alpha)$ (циклоида)

ЛАБОРАТОРНАЯ РАБОТА № 3

Элементы деловой графики

Цель работы – научиться составлять программы, осуществляющие графическое представление информации в виде графиков, диаграмм, гистограмм.

Необходимые сведения из теории *(см. 3.1, 3.2,ПРИЛОЖЕНИЕ 1):*

1) Алгоритм построения графика функции.
2) Алгоритм построения гистограммы.
3) Алгоритм построения круговой диаграммы.

Задание

Построить график, гистограмму, круговую диаграмму по статистическим данным.

Порядок выполнения работы

1) Собрать статистические данные по самостоятельно определенной теме.
2) Построить график по заданным значениям функции с использованием коэффициентов масштабирования.
3) Построить гистограмму по статистическим данным.
4) Построить круговую диаграмму по статистическим данным.

ЛАБОРАТОРНАЯ РАБОТА № 4

Использование окон в компьютерной графике

Цель работы – научиться составлять программы, осуществляющие операции с отдельными частями изображения.

Необходимые сведения из теории *(см. 1.1, 1.2, 2.1, 2.2, 2.3,2.4,2.5, ПРИЛОЖЕНИЕ 1):*

1) Алгоритм нахождения точки пересечения двух прямых, заданных координатами двух точек, принадлежащих этим прямым.
2) Условие определения положения точки относительно прямой.
3) Алгоритмы аффинных преобразований на плоскости.
4) Алгоритмы операций выделения, отсечения и стирания в выделенной экранной области.

Задание

Построить геометрическую фигуру и выделить часть изображения, попавшего в окно.

Порядок выполнения работы

1) Построить геометрическую фигуру с произвольными параметрами (из таблицы 1).
2) Определить координаты окна так, чтобы часть геометрической фигура попала в область окна.
3) Найти точки пересечения окна и сторон геометрической фигуры.
4) Выделить линии, попавшие в окно, цветом или изменением толщины линии.
5) Выделенный фрагмент перенести в центр любой экранной четверти, используя коэффициент масштабирования.

6) используя процедуры (imagesize, getmem, getimage, putimage), запомнить выделенный фрагмент изображения и вывести во все три оставшиеся экранные четверти.

ЛАБОРАТОРНАЯ РАБОТА № 5

Виды проектирования

Цель работы – научиться составлять программы, осуществляющие операции проектирования трехмерных объектов на плоскость.

Необходимые сведения из теории *(см. 2.4, 2.5, 2.6, 4.1, 4.2, ПРИЛОЖЕНИЕ 1):*

1) Алгоритмы построения Платоновых тел
2) Виды параллельного проектирования. Матрицы параллельного проектирования.
3) Виды перспективного проектирования. Матрицы перспективного проектирования
4) Алгоритмы аффинных преобразований в пространстве.
5) Алгоритмы удаления невидимых ребер Платоновых тел.

Задание

Построить перспективную проекцию Платонова тела с удалением невидимых ребер.

Порядок выполнения работы

1) Определить входные данные: координаты Платонова тела, координаты вершин, координаты ребер из таблицы 3 по варианту.
2) Реализовать алгоритм поворота заданного Платонова тела, повернуть в пространстве с помощью аффинных преобразований на заданные углы из таблицы 3.
3) Получить изображение на экране с помощью матриц перспективного и параллельного проектирования.
4) Применить алгоритм удаления невидимых ребер.

Таблица 3. Платонова тела по вариантам

Вариант	Тело	Поворот на α относительно оси OX, на β – оси OY	Коэффициент
1	Тетраэдр	40 – 25	2
2	Октаэдр	10 – 20	0,5
3	Куб	20 – 70	0,5
4	Тетраэдр	10 – 40	4
5	Октаэдр	60 – 20	0,5
6	Куб	30 –30	5
7	Тетраэдр	45 – 20	0,5

8	Октаэдр	60 – 15	1,5
9	Куб	20 –40	0,5
10	Тетраэдр	10 – 20	1,5
11	Октаэдр	30 – 50	5
12	Куб	60 – 40	0,5
13	Тетраэдр	75 – 50	0,5
14	Куб	35 –60	0,5

ЛАБОРАТОРНАЯ РАБОТА № 6

Растровые алгоритмы

Цель работы – научиться составлять программы с использованием растровых алгоритмов.

Необходимые сведения из теории *(см. 5.1,5.2,5.3):*

1) Основные понятия: растр, связность, растровый алгоритм, растровые данные.
2) Алгоритмы построения отрезка.
3) Тест на принадлежность точки многоугольнику.
4) Алгоритм закраски многоугольника.
5) Алгоритмы заполнения сплошной области по критерию связности.

Задание

Построить изображение многоугольника, стороны которого представлены растровыми отрезками, закрасить его. Внутри многоугольника нарисовать несколько окружностей, заполнить их другим цветом.

Порядок выполнения работы

1) Написать процедуру рисования растрового отрезка по заданным координатам концов отрезка.
2) Построить многоугольник с помощью процедуры из п.1).
3) Закрасить многоугольник, используя тест на принадлежность точки многоугольнику.
4) Построить несколько окружностей внутри многоугольника. Заполнить их с использованием алгоритма заполнения областей по критерию связности.

ЛАБОРАТОРНАЯ РАБОТА № 7

Интерполяция сплайнами

Цель работы – научиться составлять программы, осуществляющие построение сплайновых кривых .

Необходимые сведения из теории *(см. 6.1,6.2,8.1,8.2):*

1) Основные свойства сплайн-функции.
2) Основные свойства кривой Безье.
3) Основные свойства B-сплайновой кривой.

4) Основные свойства составной B-сплайновой кривой.

5) Работа с устройства ввода графической информации (мышки).

Задание

Построить изображения сплайновых кривых, ввод данных осуществлять при помощи устройства ввода графической информации (мышка).

Порядок выполнения работы

1) Написать процедуру генерации элементарной кривой Безье.
2) Написать процедуру генерации элементарной B-сплайновой кривой.
3) Организовать ввод данных с помощью мыши.
4) Построить изображения кривой Безье с помощью процедуры п.1.
5) Построить изображения составной B-сплайновой кривой с помощью процедуры п.2.
6) Построить замкнутую составную B-сплайновую кривую.

ЛАБОРАТОРНАЯ РАБОТА № 8

Построение графика функции двух переменных

Цель работы – научиться составлять программы, реализующие построения графика функции двух переменных.

Необходимые сведения из теории *(см. 4.1, 4.2, 7.0):*

1) Виды проектирования.
2) Алгоритм построения графика функции двух переменных.
3) Растровая версия алгоритма построения графика функции двух переменных.
4) Алгоритм полутонового изображения графика функции двух переменных.
5) Алгоритм удаления невидимых линий.

Таблица 4. Уравнения по вариантам

Вариант	Уравнение
1	$z=\sin(\sqrt{(x^2+y^2)})$
2	$z=\cos(x\ y)$
3	$z=\frac{xy(x-y)(x-y)}{\sqrt{x^2+y^2}}$
4	$z=\frac{4}{(x^2+y^2)}$
5	$z=\frac{4}{(x+y)}$
6	$z=\sqrt{xy}$
7	$z=\frac{xy}{y-x}$
8	$z=\sqrt{a^2-x^2-y^2}$
9	$z=\frac{1}{\sqrt{1-x^2-y^2}}$
10	$z=\sqrt{1-x^2/4-y}$

11	$z = x^2 - y$
12	$z = a \cdot \sin(\pi / b \cdot \sqrt{x^2 + y^2})$
13	$z = x + \sqrt{x^2 - y^2}$
14	$z = a \cdot \cos(xy \cdot \pi / b)$

Задание

Построить график функции двух переменных.

Порядок выполнения работы

1) Определить аналитически области изменений x и y, для которых функция имеет вещественное значение.

2) Построить изображения функции двух переменных.

3) По заданному уравнению (из таблицы 4) построить функцию двух переменных по алгоритму.

4) Использовать алгоритм удаления невидимых линий.

ПРИЛОЖЕНИЕ 1

Основные группы операций графической библиотеки

1) Графическая библиотека – модуль Graph Turbo Pascal 6.0.

uses Graph – для подключения графической библиотеки Graph.

2) ***initgraph (****var driver, mode: integer; driverpath: stritg****)*** - процедура инициализации графического режима.

driver: определен ряд констант, задающих набор стандартных драйверов: CGA,EGA,VGA и

mode : определяет режим GAHi, EGALo, EGAHi,VGALo,VGAHi

driverpath: указывается имя каталога, где находятся драйверы адаптера - файлы типа BGI (cga.bgi, egavga.bgi).

3) ***Detect*** - выбор режима наибольшего разрешения.

4) функция ***GraphResult*** - возвращает код завершения предыдущей графической операции, успешному выполнению соответствует значение grOk.

closegraph - процедура окончания работы с режимом

5) ***putpixel****(x,y:integer;color: word);* - процедура ставит пиксел заданного цвета Color в точку с координатами (x,y).

getpixel*(x,y:integer):word*; - функция возвращает цвет пиксела с координатами (x,y).

6) ***setcolor****(color:word);* - устанавливает цвет пера;

7) ***setLineStile****(Style,Pattern,Width:word);* - задает шаблон и толщину линии;

8) ***line****(x_1,y_1,x_2,y_2:integer);* - процедура рисует отрезок, соединяющий точки *(x_1,y_1) и (x_2,y_2);*

9) ***circle****(x,y,r:integer);* - процедура рисует окружность радиуса r с центром в точке *(x,y).*

10) С помощью процедуры ***Bar***$(x_1,y_1,x_2,y_2$*:integer);* можно построить прямоугольник, стороны которого параллельны осям координат.

11) Для построения изображения объекта со сторонами, не параллельными осям координат, используются процедуры ***Line, LineTo, LineReal***, которые отличаются друг от друга только набором задаваемых параметров.

12) Функция ***Ellipse*** рисует эллипс ($x=a\cdot\cos\varphi$, $y=b\cdot\sin\varphi$, $0\le\varphi\le2\pi$), полуоси которого параллельны осям координат, для произвольного эллипса используется по пиксельное изображение (***Putpixel***).

13) При обращении к процедурам ***Arc, Ellipse, Sector, PieSlice***, угловые величины задаются в градусной мере.

14) Аргументы тригонометрических функций *(Sin,Cos)* задаются в радианной мере угла.

Коэффициенты перехода:

(***1 градус=π/180 радиан***; ***1 радиан=180/π градусов***.

Результат вычисления выражений, в которых имеются обращения к функциям ***Sin(φ), Cos(φ),*** должен округляться, так как эти функции возвращают в качестве результата действительную величину.

15) ***setwritemode****(mode:integer)* – режима вывода

mode принимает следующие значения:

*normalput - происходит простой вывод;

*notput - происходит вывод инверсного изображения;

*orput - используется побитовая операция ИЛИ;

*andput - используется побитовая операция И.

16) Работа с изображениями, поддерживается возможность запоминания прямоугольного фрагмента изображения в оперативной памяти и вывод его на экран.

imagesize*(x_1,y_1,x_2,y_2:integer):word*; - функция определяет объем памяти, требуемый для запоминания изображения

getimage(x_1,y_1,x_2,y_2*:integer;var image);* - процедура для запоминания прямоугольного фрагмента изображения

putimage*(x,y:integer;var image;method:word);* – процедура вывода прямоугольного фрагмента изображения – чтобы точка *(x,y)* была в верхнем левом углу изображения. Параметр *method* определяет режим вывода (см. выше).

17) Дополнительные возможности по управлению цветом

setpalette*(color:word,colorvalue:shortint);* - процедура устанавливает для логического цвета Color физический цвет Colorvalue, формируемый путем наложения трех базовых цветов - красного, синего, зеленого.

18) Использование видеостраниц.

setvisualpage*(page:word);* – установка видимой страницы, где *page* – номер страницы, которая станет видимой на экране после вызова этой процедуры.

setactivepage*(page:word);* - установка активной страницы, где *page* – номер страницы, на которую происходит весь вывод.

19) Учесть фактор различающихся разрешающих способностей в горизонтальном и вертикальном направлениях. Процедура ***GetAspectRatio*** позволяет получить такие два числа x_a, y_a, отношение которых равно отношению сторон полученного прямоугольника при рисовании квадрата. Отношение этих величин обратно пропорционально отношению разрешающих способностей $\frac{x_a}{y_a} = \frac{r_y}{r_x}$, где *rx, ry* – разрешающие способности вдоль осей X и Y соответственно.

Например, при вычерчивании квадрата со стороной a пикселей по оси X отложим номинальное значение $r_x=a$, тогда по оси Y следует задать длину в $r_y = a\frac{x_a}{y_a}$ пикселей.

СПИСОК ЛИТЕРАТУРЫ

1. Дудник Е.А. Лабораторный практикум: Методическое пособие для студентов специальности 073002 «Прикладная математика» /Рубцовский индустриальный институт, РИО, 1999, 57 с.

2. Шикин Е., Боресков А. Компьютерная графика.-М.:Диалог-МИФИ, 1996. -287 с.

3. Хирн Д., Бейкер М. Микрокомпьютерная графика.-М.:Мир, 1987. -352 с.

4. Павлидиус Т. Алгоритмы машинной графики и обработки изображений. - М.:Радио и связь,1986.-400 с.

5. Аммерал Л. Машинная графика на языке Си.-М.:Сол Систем, 1992.

Том.1: Принципы программирования в машинной графике.-224с.

Том 2: Машинная графика на персональных компьютерах.-232 с.

Том 3: Интерактивная трехмерная машинная графика.-317 с.

Том 4: Программирование графики на Turbo Си.-221с.

6. 6.Эгрон Ж. Синтез изображений. Базовые алгоритмы. -М.: Радио и связь, 1993.-216 с.