МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра ВТ

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Организация ЭВМиС»

Студент гр. 1308 _____ Мельник Д. А.

Студент гр. 1308 _____ Лепов А. В.

Преподаватель _____ Костичев С.В.

Санкт-Петербург

2022

ЗАДАНИЕ

Цель работы:

Изучение работы с видеосистемой в графическом режиме, вывод графика заданной функции, вывод системы координат для выведенного графика (в соответствии с масштабом и расположением точек графика).

Задача: Разработать программу выполняющую следующие действия:

1) Вывод графика заданной функции двумя способами: рисованием пикселей и рисованием линий
2) Вывод разметки системы координат, согласование масштабов графика и системы координат.

ТЗ:

Функция: $f(x) = Sin^3(x) + Cos^2(x)$

Отрезок, на который приходится график: $[3\pi/2; 8\pi]$

РАСПРЕДЕЛЕНИЕ ОБЯЗАННОСТЕЙ В БРИГАДЕ

Мельник Даниил:

1) Написание программы включая функцию рисования графика
2) Отладка программы.

Лепов Алексей:

1) Работа с документацией по теме
2) Формирование теоретической базы для программы
3) Оформление отчёта

КРАТКИЕ СВЕДЕНИЯ О ВИДЕОСИСТЕМАХ ПЭВМ

Использование графики в языке C++ - это многошаговый процесс. Прежде всего необходимо определить тип видеоадаптера. Затем устанавливается подходящий режим его работы и выполняется инициализация графической системы в выбранном режиме. После этого становятся доступными для использования функции графической библиотеки graphicx.h для построения основных графических примитивов: отрезков прямых линий, окружностей, эллипсов, прямоугольников, секторов, дуг и т.д., появляется возможность вывода текста с использованием различных шрифтов.

Использование библиотеки графики намного сокращает объем программирования для вывода основных графических примитивов. C++ "маскирует" многие технические детали управления оборудованием, о которых пользователь должен быть осведомлен при работе с видеоадаптером через порты или BIOS.

Однако достоинство графической модели заключается в ее относительной независимости от различных типов видеоадаптеров и открытости для дальнейших расширений. Появление новых типов видеоадаптеров не потребует большой переработки программ, так как все новые особенности аппаратуры будут учитываться в средствах библиотеки C++.

Весь код библиотеки графики разбивается на две части: немобильную, которая зависит от типа видеоадаптера и мобильную.

Немобильная часть представляет собой так называемый .BGI-драйвер (BGI - Borland Graphics Interface). Драйвер является обработчиком прерывания 10h, который должен дополнить системный обработчик до того, как будут использоваться мобильные функции.

Основные функции, выполняемые .BGI-драйвером, сводятся к установке и обновлению ряда внешних переменных, которые могут изменяться как функциями системного обработчика прерывания 10h.

Графические режимы, поддерживаемые библиотекой графики, задаются символическими константами, описанными в заголовочном файле <graphics.h> в перечислимом типе graphics_modes.

Инициализацию графической модели выполняет функция initgraph(): void far initgraph(int *graphdriver, int *graphmode, char * pathtodriver).

- closegraph() - эта функция освобождает память, распределенную под драйверы графики, файлы шрифтов и промежуточные данные и восстанавливает режим работы адаптера в то состояние, в котором он находился до выполнения инициализации системы.

Только после определения типа адаптера и его максимального режима выполняются установка нужного пользователю режима и загрузка .BGI-драйвера. Далее приводится описание функции detectgraph().

- detectgraph () - определяет тип активного видеоадаптера системы и тип подключенного монитора в персональном компьютере. Информация о подходящем драйвере и максимальном режиме возвращается в точку вызова в двух переменных, на которые указывают graphdriver и graphmode соответственно.
- graphresult() - возвращает значение внутреннего кода ошибки, установленного последним обращением к функциям графической библиотеки.

После того, как проведена инициализация графической системы, может быть установлен другой, не превосходящий максимального, режим видеоадаптера и выбраны цвета для пикселов. Установку режима выполняет функция setgraphmode(). Целая группа функций – getgraphmode(), getmaxmode(), getmodename() , getmoderange() - упрощает работу по определению текущего установленного режима. Две функции позволяют определить ширину и высоту экрана в пикселах для текущего видеорежима: getmaxx() и getmaxy(). Функция restorecrtmode() возвращает видеоадаптер в текстовый режим. Далее следует описание упомянутых функций.

- getgraphmode () - возвращает текущий графический режим, установленный для графической модели функциями initgraph() или setgraphmode().
- getmaxmode()
- getmodename() - возвращает указатель на ASCII-строку символов, содержащую имя символической константы, соответствующей режиму mode_number.
- setgraphmode() - устанавливает видеосистему в режим, заданный значением переменной mode, и сбрасывает значения внутренних переменных системы графики в их значения по умолчанию (стиль линий, маска заполнения, шрифт и т.д.).
- restorecrtmode() - возвращает видеоадаптер в режим, в котором он был до выполнения инициализации системы графики.

После инициализации системы графики и установки нужного видеорежима возможен выбор необходимых цветов пикселов.

- getbkcolor() - возвращает целое число, равное коду цвета фона.
- getmaxcolor() - возвращает максимальное значение кода цвета пиксела минус 1.
- setbkcolor () - устанавливает новый цвет пикселов, имеющих код цвета 0. Новый цвет фона задает значение аргумента color.
- setcolor ()- устанавливает цвет, используемый функциями графического вывода в значение, заданное аргументом color.

Окно экрана в графическом режиме, или графическое окно (viewport), - это прямоугольная область экрана, заданная пиксельными координатами левого верхнего и правого нижнего углов. В графическом окне определены относительные координаты.

- getviewsettings() - заполняет поля структурной переменной по шаблону viewporttype информацией о графическом окне.

Основное применение функции - определение и сохранение

характеристик текущего графического окна перед переопределением текущего окна для последующего восстановления параметров окна.

- setviewport () - описывает новое графическое окно с координатами (столбец, строка) левого верхнего угла left, top, координатами правого нижнего угла right, bottom и значением флага усечения clip.

Графические координаты X и Y измеряются в пикселах экрана относительно координат левого верхнего угла текущего окна. Функции графического вывода изменяют эти координаты в соответствии с объемом выведенной на экран информации.

- getx (), gety () - возвращают текущие координаты X и Y, измеряемые относительно координат левого верхнего угла текущего графического окна.
- moveto () - устанавливает новое значение координат текущей позиции.
- moverel () - устанавливает новое значение координат текущей позиции.

Библиотека графики позволяет выводить на экран текст различными шрифтами. C++ имеет два типа шрифтов: битовый и сегментированный.

Каждому символу битового шрифта (bit-mapped font) ставится в соответствие матрица пикселов фиксированного размера. C++ использует в качестве битового шрифта таблицу знакогенератора для символов размером 8x8.

Другой тип шрифтов, используемый при выводе текста на экран, фактически задает правило "рисования" каждого символа. Он описывается как совокупность отрезков прямых линий, или сегментов. Этим и объясняется название шрифта - сегментированный (stroke font).

В C++ доступны четыре сегментированных шрифта: Triplex, Small, Sans-Serif и Gothic.

Поведение системы графики при выводе текста в графическом режиме задается целой группой значений внутренних переменных. Их текущие установки доступны после вызова функции gettextsettings().

- gettextsettings() - заполняет поля структурной переменной по шаблону

textsettingstype информацией о текущих шрифте, направлении вывода текста, размере знакоместа относительно шрифта по умолчанию и способе "прижатия" (выравнивания) шрифта в пределах знакоместа. Функции передается указатель texttypeinfo на описанную структурную переменную.

- settextstyle() - выбирает шрифт, устанавливает направление и размер знакоместа для последующего вывода текстовой информации через функции библиотеки графики outtext() и outtextxy().
- textheight() - возвращает высоту строки символов в пикселах, на которую указывает textstring.
- textwidth () - возвращает ширину строки символов в пикселах, на которую указывает textstring.
- settextjustify() – задает новую установку выравнивания символов текста в графических режимах работы адаптера.
- outtext () - выводит ASCII-строку текста, на начало которой указывает textstring, используя текущие позицию, цвет и установки направления, типа шрифта и выравнивания строки.
- outtextxy () - выводит ASCII-строку текста, на начало которой указывает textstring, используя текущие цвет, установки направления, типа шрифта и выравнивания строки.

Функции способны выводить только нуль-терминированные строки, и для выполнения форматированного вывода в графических режимах выбранными сегментированными шрифтами поступают следующим образом.

Если инициализация системы графики выполнена успешно, становятся доступными функции графической библиотеки для построения основных графических примитивов - отрезков прямых линий, дуг, окружностей, эллипсов, прямоугольников, секторных и столбцовых диаграмм и т.д. Все функции библиотеки графики, генерирующие вывод информации на экран, работают в пределах текущего графического окна.

- getlinesettings () - возвращает информацию об установленном в текущий момент времени стиле "рисования" отрезков прямых линий и графических примитивов.
- setlinestyle ()- устанавливает стиль "рисования" отрезков прямых линий и графических примитивов.
- setwritemode() - устанавливает режим вывода отрезков прямых линий в значение, определяемое аргументом mode.

Следующий параметр системы графики - так называемый коэффициент сжатия, или коэффициент пропорциональности (aspect ratio). Он задает форму пиксела на экране монитора.

- getaspectratio () - заполняет две переменные, описанные точкой вызова, значениями коэффициента сжатия для текущего видеорежима.
- setaspectratio () - устанавливает новое значение коэффициента сжатия, которое будет использоваться системой графики при выводе геометрических примитивов - прямоугольников, дуг, окружностей, эллипсов.

Рекомендуемое использование функции - корректировка вывода графической информации при использовании нестандартных мониторов, для которых не может автоматически определить корректное значение коэффициента сжатия, а также корректировка графического вывода для мониторов с некорректной линейностью по вертикали и горизонтали.

Последние из параметров графической системы, влияющие на вывод графической информации, это маска заполнения и стиль заполнения. Маска заполнения позволяет задать способ заполнения отдельных областей экрана.

- getfillpattern () - заполняет область памяти из 8 байт, описанную точкой вызова, текущим значением маски заполнения.
- setfillpattern () - задает цвет пикселов и маску для заполнения областей экрана.
- getfillsettings() - заполняет поля структурной переменной по шаблону

struct fillsettingstype информацией о текущей маске и цвете заполнения.

- setfillstyle() - выбирает один из предопределенных стилей заполнения.
- Getpixel() - определяет, лежит ли пиксел с координатами (x, y) в текущем графическом окне, и, если лежит, возвращает код цвета этого пиксела.
- putpixel() - определяет, лежит ли пиксел с координатами (x, y) в текущем графическом окне, и, если лежит, выводит на экран пиксел, код цвета которого равен pixelcolor.

Типичным применением точечного вывода является формирование сложных изображений.

Целая группа функций библиотеки графики предназначена для вывода отрезков прямых линий.

- line() - выводит отрезок прямой линии между двумя явно специфицированными точками (x1, y1) и (x2, y2), используя текущие цвет, стиль, толщину и режим вывода линии.
- linerel() - выводит отрезок прямой линии между текущей позицией (начало отрезка) и точкой, заданной горизонтальным смещением dx и вертикальным смещением dy от текущей позиции (конец отрезка).
- lineto() - выводит отрезок прямой линии между текущей позицией (начало отрезка) и точкой, заданной горизонтальной координатой x и вертикальной координатой y (конец отрезка).

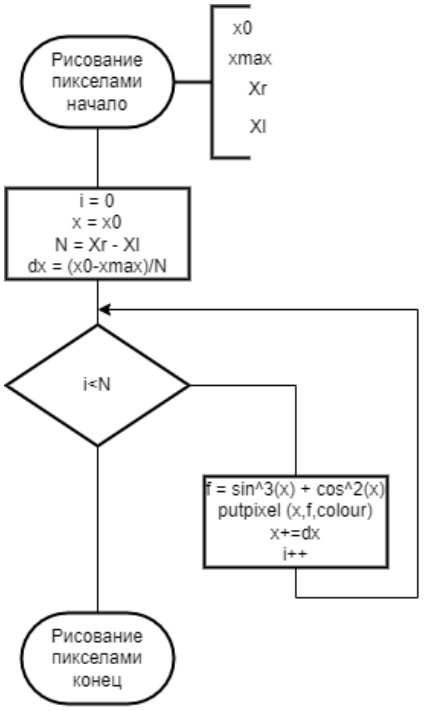
Библиотека графики содержит функции для вывода дуги окружности или целой окружности, эллиптической дуги или целого эллипса, кругового сектора, ломаной линии из нескольких отрезков прямой (полигона), прямоугольника, прямоугольной полосы заданного цвета и стиля заполнения, прямоугольника заданной толщины в аксонометрии.

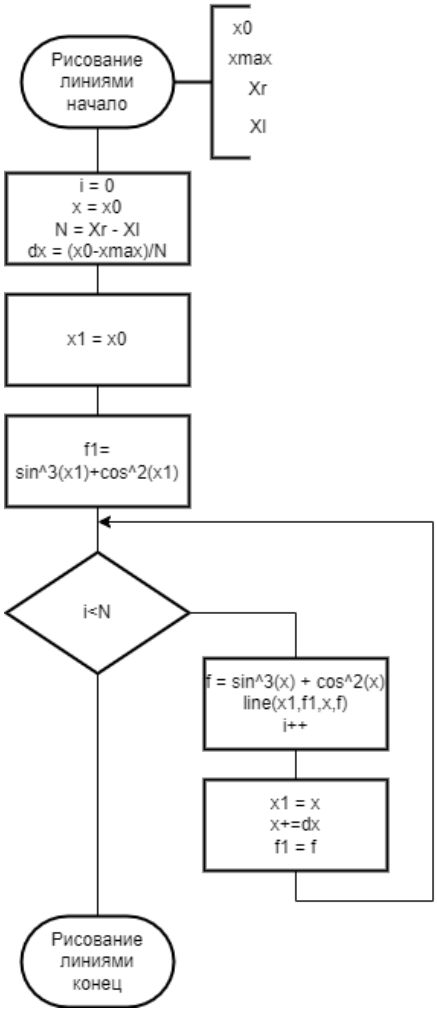
- arc() - выводит дугу окружности радиусом radius.
- bar() - выводит полосу, заполненную текущим цветом с использованием текущей маски заполнения.
- bar3d() - выводит в изометрии "столбик" и заполняет его фронтальную

поверхность текущим цветом с использованием текущей маски заполнения.

- circle() - выводит окружность заданного аргументом radius радиуса с центром, заданным координатами x и y.
- drawpoly() - "соединяет" отрезками прямых линий текущего цвета и стиля точки (полигон), координаты которых заданы парами значений.
- ellipse () - выводит эллиптическую дугу или полный эллипс, используя текущий цвет.
- fillellipse() - выводит эллипс, заполненный текущим стилем
- fillpoly() - выводит контур полигона, заданного numpoints точками
- floodfill () - заполняет текущим стилем область экрана, ограниченную непрерывной линией с цветом border, начиная с точки с координатами (x, y). Функция заполняет область либо внутри замкнутой линии, либо вне ее. Это зависит от положения начальной точки.
- pieslice() - выводит контур кругового сектора и заполняет его внутреннюю область текущим стилем.
- rectangle() - выводит контур прямоугольника, заданного координатами левого верхнего (left, top) и правого нижнего (right, bottom) углов.
- sector() - работает аналогично функции pieslice(), за исключением того, что выводится не круговая, а эллиптическая дуга.

АГОРИТМЫ РИСОВАНИЯ ГРАФИКОВ





ТЕКСТ ОТЛАЖЕННОЙ ПРОГРАММЫ

1) Рисование графика пикселами

```
#include <iostream.h>
#include <conio.h>
#include <stdio.h>
#include <math.h>
#include <dos.h>
#include <graphics.h>
#include <string.h>

void make_graph(int N, int X1, int Y1, int Y3, double x0, double xm,
double dx, float &fmax)
{
    double x;
    float f;
    int i;
    for (x = x0; x <= xm; x += dx)
    {
         f = (float)(pow((sin(x)),3.0)+pow((cos(x)),2.0));
         if (fmax < f)
              fmax = f;
    }
    for (i = 0, x = x0; i < N; i++, x += dx)
    {
         f = (float)(pow((sin(x)),3.0)+pow((cos(x)),2.0));
         putpixel(X1 + i +100 , Y3 - (int)((float)(Y3 - Y1) / (2.6 /
f)), YELLOW);
    }
}

int main(){
 clrscr();
    int driver = DETECT, mode;

    initgraph(&driver, &mode, "c://turboc3//bgi");

    int Xmax = getmaxx(), Ymax = getmaxy();
    int X1 = 40, X2 = Xmax - 40, Y1 = 50, Y2 = Ymax - 70, N = X2 -
X1 - 73 - 26, Y3 = (Y2-Y1)/2+30;
    float F, fmax = 0.;
    double x0 = 3 * M_PI / 2, xm = 8 * M_PI, dx = (xm - x0) / (N);

    setlinestyle(0, 20, 3);
    setcolor(YELLOW);
    rectangle(10, 10, Xmax - 10, Ymax - 10);

    setcolor(WHITE);
    setlinestyle(0, 1, 1);
    line(X1, Y3, X1, Y2 - 320);
    line(X1, Y3, X1, Y2 - 80);
    line(X1, Y3, X2, Y3);
```

```
    setcolor(YELLOW);
    outtextxy(X1 + 8, Y2-327, "f(x)");
    outtextxy(X2 + 8, Y3 - 4, "x");

    setcolor(WHITE);
    int i;
    for (i = -4; i < 5; ++i)
         line(X1 - 5, Y3 - i * 30, X1 + 5, Y3 - i * 30);
    for (i = 1; i < 9; ++i){
         line(X1 + 70 * i, Y3 + 5, X1 + 70 * i, Y3 - 5);
    }
    char xi[14][2];
    for (i = 4; i < 33; i += 4)
    {
         sprintf(xi[i / 4 - 1], "%dpi", i/4);
         outtextxy(X1 + 32.2 * i / 2 + i - 3, Y3 + 7, xi[i / 4 - 1]);
    }

    char yi[14][2];
    int k =0;
    for (i = 0; i < 5; i += 2)
    {
         sprintf(yi[k], "%d", i / 2);
         outtextxy(X1 - 20, Y3 - 30 * i - i, yi[k]);
         k++;
    }
    for (i=-2; i>-5; i-=2){
     sprintf(yi[k], "%d", i/2);
     outtextxy(X1-20, Y3 - 30 * i + i, yi[k]);
     k++;
    }
    k=0;

    make_graph(N, X1, Y1, Y3, x0, xm, dx, fmax);

    char cfmax[10], str[24];
    sprintf(cfmax, "%f", fmax);
    strcpy(str, "fmax = ");
    strcat(str, cfmax);
    setcolor(YELLOW);
    outtextxy(90, Ymax - 135, "f(x) = sin^3(x) + cos^2(x)");
    outtextxy(390, Ymax - 135, str);

    getch();
    closegraph();
    return 0;
}
```

2) Рисование линиями

```
#include <iostream.h>
#include <conio.h>
#include <stdio.h>
#include <math.h>
#include <dos.h>
#include <graphics.h>
#include <string.h>

void make_graph(int N, int Y1, int Y3, double x0, double xm, double
dx, float &fmax)
{
   double x, x1;
   float f, f1;
   int i;
   for (x = x0; x <= xm; x += dx)
   {
        f = (float)(pow((sin(x)),3.0)+pow((cos(x)),2.0));
        if (fmax < f)
             fmax = f;
   }
   setlinestyle(0,1,1);
   setcolor(YELLOW);
   x1 = x0;
   f1 = (float)(pow((sin(x0)),3.0)+pow((cos(x0)),2.0));
   for (i = 0, x = x0; i < N; i++, x += dx)
   {
        f = (float)(pow((sin(x)),3.0)+pow((cos(x)),2.0));
        line( x1+i+100 , Y3-(int)((float)(Y3-Y1)/(2.6/f1)) , x+i+100
, Y3-(int)((float)(Y3-Y1)/(2.6/f)) );
        x1 = x;
        f1 = f;
   }
}

int main(){
 clrscr();
   int driver = DETECT, mode;

   initgraph(&driver, &mode, "c://turboc3//bgi");

   int Xmax = getmaxx(), Ymax = getmaxy();
   int X1 = 40, X2 = Xmax - 40, Y1 = 50, Y2 = Ymax - 70, N = X2 -
X1 - 73 - 26, Y3 = (Y2-Y1)/2+30;
   float F, fmax = 0.;
   double x0 = 3 * M_PI / 2, xm = 8 * M_PI, dx = (xm - x0) / (N);

   setlinestyle(0, 20, 3);
   setcolor(YELLOW);
   rectangle(10, 10, Xmax - 10, Ymax - 10);

   setcolor(WHITE);
```

```
    setlinestyle(0, 1, 1);
    line(X1, Y3, X1, Y2 - 320);
    line(X1, Y3, X1, Y2 - 80);
    line(X1, Y3, X2, Y3);

    setcolor(YELLOW);
    outtextxy(X1 + 8, Y2-327, "f(x)");
    outtextxy(X2 + 8, Y3 - 4, "x");

    setcolor(WHITE);
    int i;
    for (i = -4; i < 5; ++i)
         line(X1 - 5, Y3 - i * 30, X1 + 5, Y3 - i * 30);
    for (i = 1; i < 9; ++i){
         line(X1 + 70 * i, Y3 + 5, X1 + 70 * i, Y3 - 5);
    }
    char xi[14][2];
    for (i = 4; i < 33; i += 4)
    {
         sprintf(xi[i / 4 - 1], "%dpi", i/4);
         outtextxy(X1 + 32.2 * i / 2 + i - 3, Y3 + 7, xi[i / 4 - 1]);
    }

    char yi[14][2];
    int k =0;
    for (i = 0; i < 5; i += 2)
    {
         sprintf(yi[k], "%d", i / 2);
         outtextxy(X1 - 20, Y3 - 30 * i - i, yi[k]);
         k++;
    }
    for (i=-2; i>-5; i-=2){
     sprintf(yi[k], "%d", i/2);
     outtextxy(X1-20, Y3 - 30 * i + i, yi[k]);
     k++;
    }
    k=0;

    make_graph(N, Y1, Y3, x0, xm, dx, fmax);

    char cfmax[10], str[24];
    sprintf(cfmax, "%f", fmax);
    strcpy(str, "fmax = ");
    strcat(str, cfmax);
    setcolor(YELLOW);
    outtextxy(90, Ymax - 135, "f(x) = sin^3(x) + cos^2(x)");
    outtextxy(390, Ymax - 135, str);
    getch();
    closegraph();
    return 0;
}
```

ПРИМЕРЫ ЗАПУСКА ПРОГРАММЫ

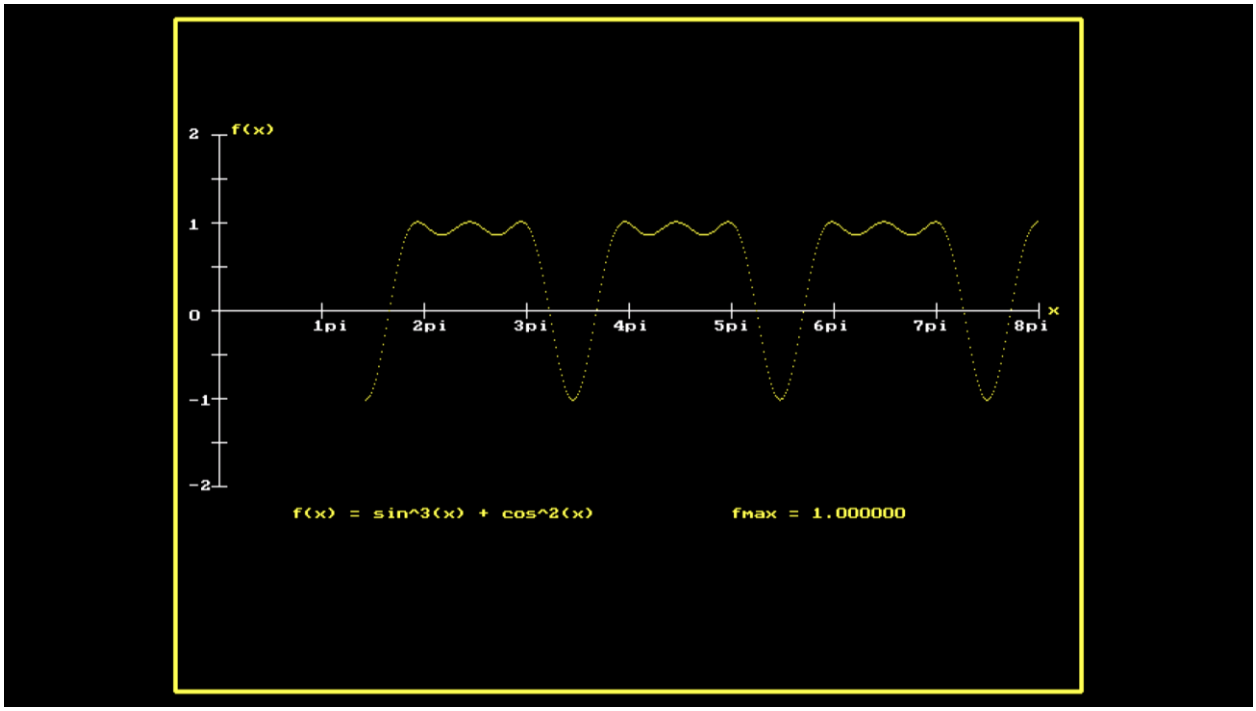


Рис. 1 — Пример работы программы на пикселах.

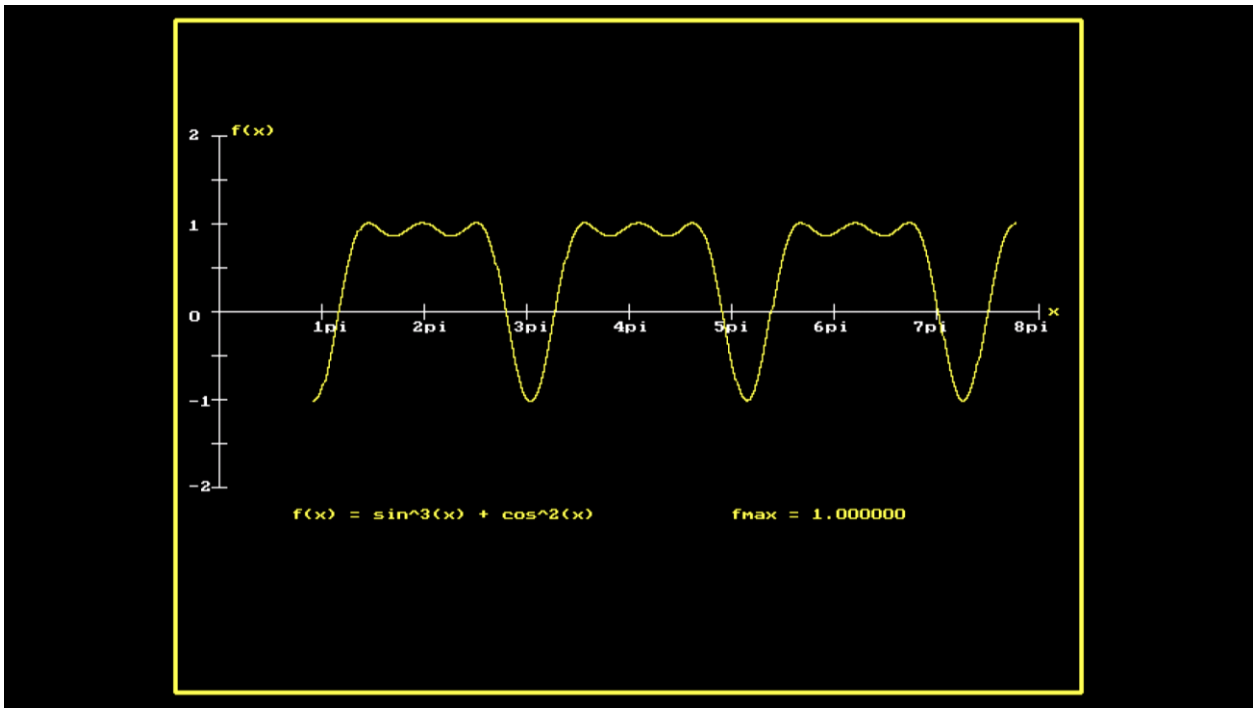


Рис. 2 – Пример работы программы на линиях

ВЫВОД

В процессе выполнения лабораторной работы были получены навыки работы с видеосистемой в графическом режиме, вывод графика заданной функции с масштабированием и разметкой осей. Рисование графика выполнено двумя способами: пикселами и линиями.