Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет ИТ

Кафедра ИиВД

Специальность: 1-98 01 03 – «Программное обеспечение информационной безопасности мобильных систем»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

КУРСОВОЙ РАБОТЫ

по дисциплине Компьютерная геометрия и графика

Тема Приложение Windows «Фигуры Лиссажу»

Исполнитель:

студент 3 курса 7 группы

Якубенко Ксения Дмитриевна

Руководитель:

Преподаватель А. А. Дятко

Курсовой проект защищен

с оценкой:

Руководитель: А. А. Дятко

Минск 2020

Содержание

[Введение 3](#_Toc57939643)

[Математическое выражение для кривой Лиссажу 4](#_Toc57939644)

[Примеры Фигур Лиссажу 5](#_Toc57939645)

[Применение в технике 6](#_Toc57939646)

[1. Постановка задачи 7](#_Toc57939647)

[2. Разработка алгоритма 8](#_Toc57939648)

[3. Вычисления для построения фигуры Лиссажу 9](#_Toc57939649)

[4. Программная реализация 15](#_Toc57939650)

[Заключение 17](#_Toc57939651)

[Список используемой литературы 18](#_Toc57939652)

[Приложение А 19](#_Toc57939653)

# Введение

**Фигуры Лиссажу** —  замкнутые траектории, прочерчиваемые точкой, совершающей одновременно два гармонических колебания в двух взаимно перпендикулярных направлениях.

Впервые изучены французским учёным Жюлем Антуаном Лиссажу. Вид фигур (Рис. 1) зависит от соотношения между периодами (частотами), фазами и амплитудами обоих колебаний. В простейшем случае равенства обоих периодов фигуры представляют собой эллипсы, которые при разности фаз или вырождаются в отрезки прямых, а при разности фаз  и равенстве амплитуд превращаются в окружность. Если периоды обоих колебаний неточно совпадают, то разность фаз всё время меняется, вследствие чего эллипс всё время деформируется. При существенно различных периодах фигуры Лиссажу не наблюдаются. Однако, если периоды относятся как целые числа, то через промежуток времени, равный наименьшему кратному обоих периодов, движущаяся точка снова возвращается в то же положение — получаются фигуры Лиссажу более сложной формы. Фигуры Лиссажу вписываются в прямоугольник, центр которого совпадает с началом координат, а стороны параллельны осям координат и расположены по обе стороны от них на расстояниях, равных амплитудам колебаний.

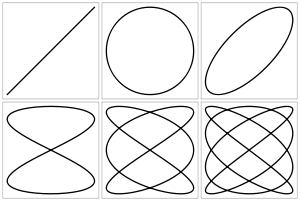


Рисунок 1. Фигуры Лиссажу.

## Математическое выражение для кривой Лиссажу

где *A*, *B* — амплитуды колебаний, *a*, *b* — частоты, *f* — сдвиг фаз.

Вид кривой сильно зависит от соотношения *a*/*b*. Когда соотношение равно 1, фигура Лиссажу имеет вид эллипса, при определённых условиях она имеет вид прямой (*A* = *B*, *δ*= π/2 радиан) и отрезка прямой (*δ* = 0). Ещё один пример фигуры Лиссажу — парабола (*a*/*b* = 2, *δ* = π/2). При других соотношениях фигуры Лиссажу представляют собой более сложные фигуры, которые являются замкнутыми при условии *a*/*b* — рациональное число.

## Примеры Фигур Лиссажу

Примеры фигур Лиссажу ниже с *δ* = *π*/2, нечётным натуральным числом *a*, и также натуральным числом *b*, и |*a* − *b*| = 1 (таблица 1).

Таблица 1. Фигуры Лиссажу.

|  |  |  |
| --- | --- | --- |
| Описание: D:\Data Dump\120px-Lissajous_curve_1by2.svg.png | Описание: D:\Data Dump\120px-Lissajous_curve_3by2.svg.png | Описание: D:\Data Dump\120px-Lissajous_curve_3by4.svg.png |
| *a* = 1, *b* = 2 | *a* = 3, *b* = 2 | *a* = 3, *b* = 4 |
| Описание: D:\Data Dump\120px-Lissajous_curve_5by4.svg.png | Описание: D:\Data Dump\120px-Lissajous_curve_5by6.svg.png | Описание: D:\Data Dump\120px-Lissajous_curve_9by8.svg.png |
| *a* = 5, *b* = 4 | *a* = 5, *b* = 6 | *a* = 9, *b* = 8 |

## Применение в технике

Если подать на входы «**X**» и «**Y**» осциллографа сигналы близких частот, то на экране можно увидеть фигуры Лиссажу. Этот метод широко используется для сравнения частот двух источников сигналов и для подстройки одного источника под частоту другого. Когда частоты близки, но не равны друг другу, фигура на экране вращается, причем период цикла вращения является величиной, обратной разности частот, например, период оборота равен 2 с — разница в частотах сигналов равна 0,5 Гц. При равенстве частот фигура застывает неподвижно, в любой фазе, однако на практике, за счет кратковременных нестабильностей сигналов, фигура на экране осциллографа обычно чуть-чуть подрагивает. Использовать для сравнения можно не только одинаковые частоты, но и находящиеся в кратном отношении, например, если образцовый источник может выдавать частоту только 5 МГц, а настраиваемый источник — 2,5 МГц.

# Постановка задачи

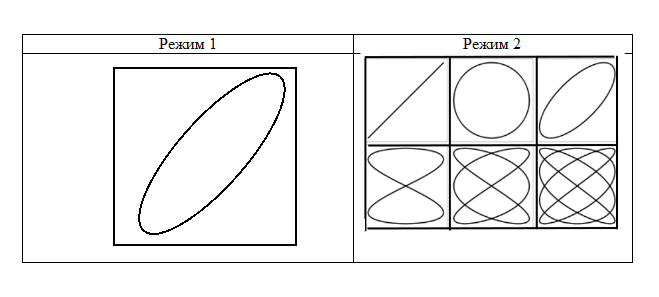
Создать приложение Windows для построения фигур Лиссажу на плоскости X0Y (Рис. 5.1)

Фигуры Лиссажу формируются колебаниями:

Изменяемые параметры:

* амплитуда – ;
* частота [Гц]
* коэффициент ;
* фазовый сдвиг [Град.];

1. Создать изображение фигуры Лиссажу в режиме анимации при изменении значения фазового сдвига с шагом при постоянном значении параметров (Режим 1, рис. 5.1).
2. Обеспечить отображение на экране монитора сразу нескольких (6) объектов, имеющих свои параметры (Режим 2, рис. 5.1).
3. Для установки параметров рисунка и фигуры (фигур) Лиссажу использовать окно диалога. Предусмотреть параметры по умолчанию, которые должны появляться в окне диалога при его открытии.
4. Для выбора режима использовать меню.
5. Обеспечить запись изображения в графический файл \*.bmp



# 2. Разработка алгоритма

В разрабатываемом приложение необходимо реализовать взаимодействие с пользователем. Для этих целей используются диалоговые окна.

Для вычисления координат x и y,а она также рассчитывает количество точек для полного отображение фигуры, используется функция, grapg1().

Также должен быть реализован блок Draw для создания графика в памяти компьютера и переносе его на дисплей монитора.

Блок схема алгоритма приведена на рис.1.



Рис. 1. Блок-схема алгоритма

# 3. Вычисления для построения фигуры Лиссажу

Для примера вычислений возьмем параметры:

* А=1
* В=1
* а=3
* в=3
* f=40о

И область в которой мы хотим построить нашу фигуру имеет координаты в ОСК:

* RW(40,40,825,415) - прямоугольная область,

где (40,40)-координаты левого верхнего угла, а (825,415)-координаты правого нижнего угла областей.

Или более простым: RW(left, top, right, bottom), где

RW.left =40

RW.top = 40

RW.right = 825

RW.bottom= 415

Перейдем к построению:

**Во первых**, необходимо перевести координаты фигуры Лиссажу из мировой системы координат(МСК) в оконные(ОСК).

У нас есть область, в которой должна отображаться фигура, после построения, но для того, чтобы ее отобразить на экране, необходимо ее координаты перевести из МСК в ОСК.

Найдем эту область в МСК:

Т.к. все наши вариации фигуры Лиссажу им разные размеры и формы, то самым простым способом, для определения области, в которой будет отображаться наша фигура, будет – нахождение наименьших и наибольших значений по Х и по У фигуры.

Это будет выглядеть так:

RS(xmin, ymax, xmax, ymin) – прямоугольная область, в которой будет находиться наша фигура, в МСК, где

xmin – наименьшее значение функции по Х

ymax – наибольшее значение функции по У

xmax – наибольшее значение функции по Х

ymin – наименьшее значение функции по У,

таким образом (xmin, ymax)- координаты верхнего левого угла области, а (xmax, ymin) – координаты нижнего правого угла области.

Или более простым: RS(left, top, right, bottom), т.е.

RS.left = xmin

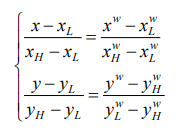
RS.top = ymax

RS.right = xmax

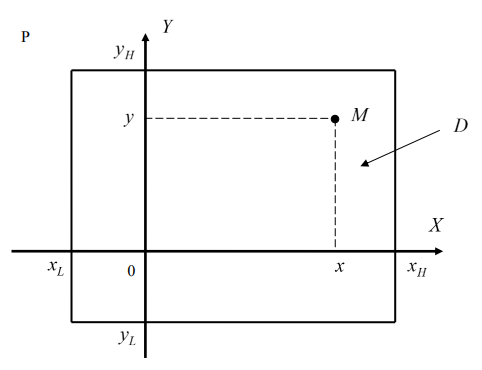
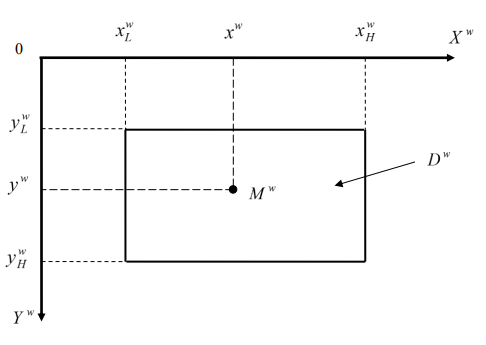
RS.bottom= ymin

Рассмотрим перевод координат из мировой системы координат в оконную на примере координат точки. Пусть есть M=m(x,y)- точка графика в МСК, а Mw=Mw(xw,yw)- ее образ в ОСК.

Тогда справедлива следующая система:



Которая выводится из следующих рисунков:

Тогда для наших областей:

xlw = RW.left

ylw =RW.top

xhw =RW.right

yhw =RW.bottom

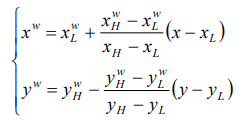
xl = RS.left

yl =RS.top

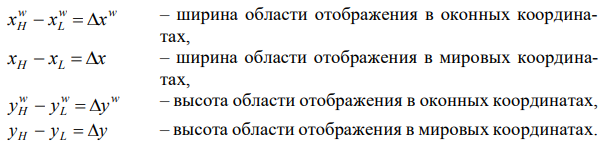
xh =RS.right

yh = RS.bottom

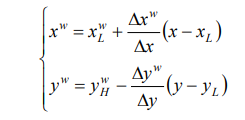
Выразим координаты в ОСК:



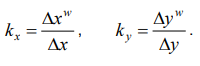
Введем обозначения:



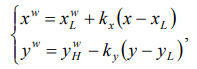
Тогда можно упростить систему:



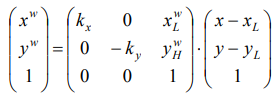
Произведем замену:



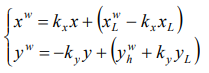
Тогда система примет вид:



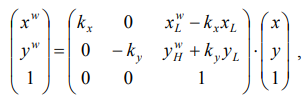
Представив в виде матрицы получим:



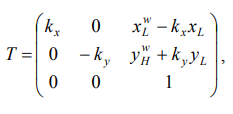
Перепишем систему в другом виде, для того, чтобы выразить x и y:



Тогда в матричном виде:



Т.е. для того, чтобы перевести координаты из МСК в ОСК необходимо умножить координаты в МСК на матрицу пересчета T:



**Тогда в нашем случае, зная координаты в МСК(RS) и ОСК(RW) можно найти матрицу пересчета. Она примет вид:**

M=

Тогда, для того чтобы изобразить нашу фигуру необходимо:

\*M

Тогда найдем минимальное и максимальное значение по X и Y для фигуры Лиссажу:

Для того, чтобы найти минимальное и максимальное значение необходимо:

1. Находим минимальное и максимальное значение функции по Х:
2. Необходимо найти первую производную:

x(t)=

1. Приравнять найденное выражение к 0:

t1=0,291

t2=3,432

Тогда:

x(0,291)=0,027

x(3,432)=0,19

1. Находим минимальное и максимальное значение функции по У:
2. Находим первую производную:

y(t)=

2)Приравниваем найденное значение к 0:

t1=0,524

t2=3,665

Тогда:

y(0,524)=0,027

y(3,665)=0,19

Таким образом:

xmin=0,027

xmax=0,19

ymin=0,027

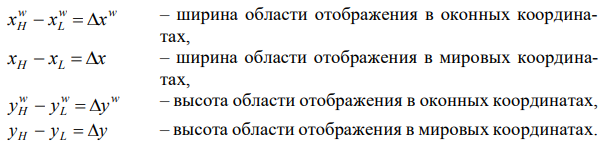
ymax=0,19

RS(0,027; 0,19; 0,19; 0,027)

RW(40,40,825,415)

Зная координаты, находим коэффициенты для матрицы пересчетов:

kx= ky=



kx=(825-40)/(0,19-0,027)=4815,95

ky=(415-40)/(0,19-0,027)=2300,6

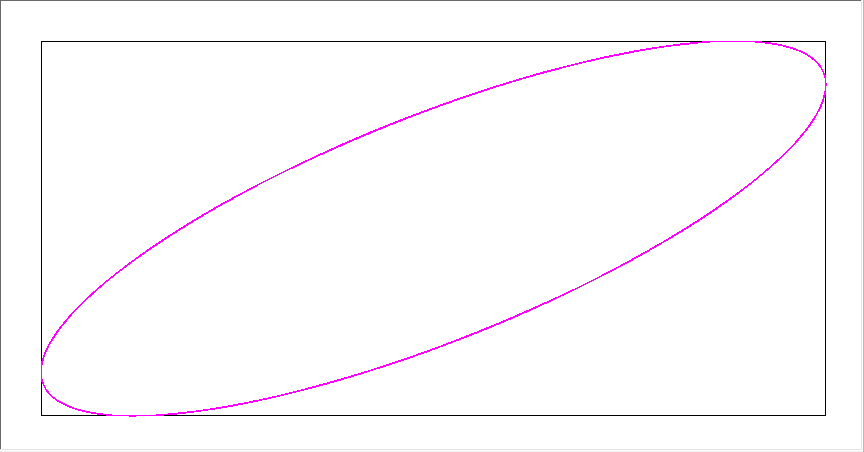
Тогда, матрица для данной задачи примет вид:

M=

M=

Тогда координаты для построения фигуры Лиссажу:

Теперь можно построить график в ОСК, подставляя различные значения переменной t:



# 4. Программная реализация

В данной курсовой работе было реализовано программное средство для отображения фигур Лиссажу с различными параметрами (см. «Постановка задачи»).

Начальное окно представлено на рис. 2.

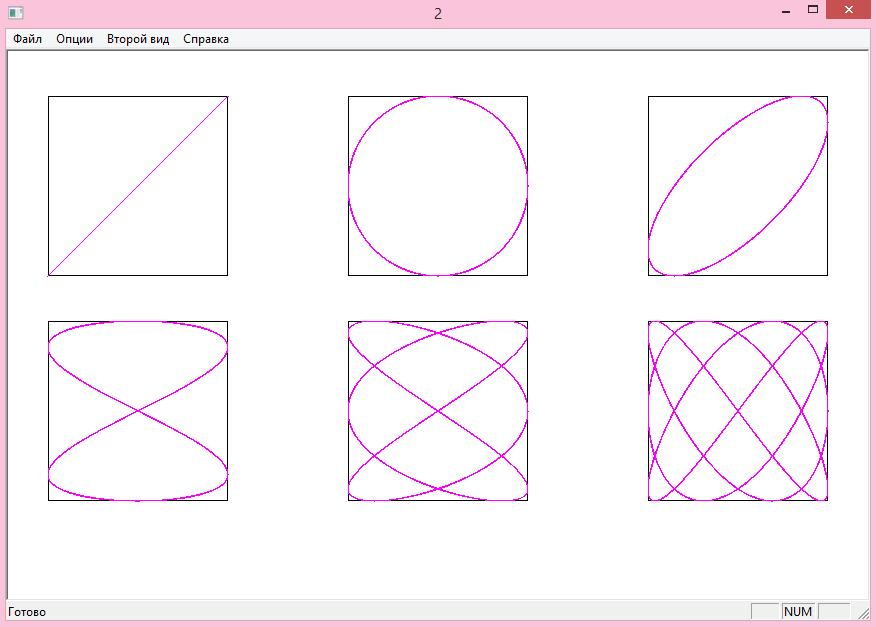


Рис. 2. Начальное окно

Диалоговое окно с заданием параметров отображения представлено на рис. 3.

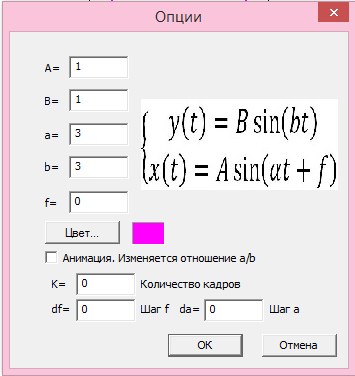


Рис. 3. Диалоговое окно “Опции”

Также в реализованном приложении была предусмотрена возможность анимированного отображения фигур. При этом пользователь может выбрать по каким параметрам проводить анимацию.

Реализована возможность сохранить текущую фигуру в формате bmp (рис. 4).

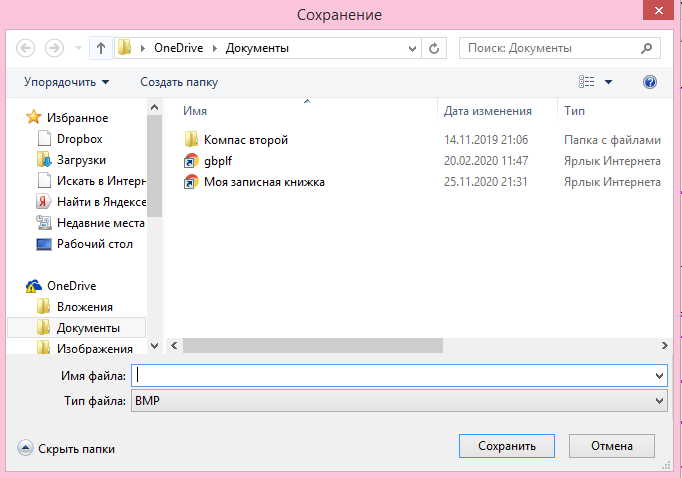


Рис. 4. Диалоговое окно сохранения.

Исходный код реализованного приложения приведен в приложении А.

# Заключение

В рамках курсовой работы было разработано программное средство для построения фигур Лиссажу, в котором есть возможность изменения параметров фигуры, а также цвета пера.

Разработка приложения осуществлялась в среде разработки MS Visual Studio 2019 с использованием языка программирования C++.

Применяются фигуры Лиссажу в технике, для сравнения частот двух источников сигналов и для подстройки одного источника под частоту другого.

# Список используемой литературы

1. Порев В.Н. Компьютерная графика. – СПб.: БХВ-Петербург, 2002. – 432с.: ил.
2. Секунов Н.Ю. Самоучитель Visual C++ .NET. – СПб.: БХВ-Петербург, 2002. – 736 с.: ил.
3. Давыдов В.Г. Visual C++. Разработка Windows-приложений с помощью MFC и API-функций. – СПб.: БХВ-Петербург, 2008. – 576 с.: ил.
4. Поляков А.Ю., Брусенцев В.А. Методы и алгоритмы компьютерной графики в примерах на Visual C++, 2-е изд., перераб. и доп. – СПб.: БХВ-Петербург, 2003. – 560 с.: ил.
5. Сайт http://ru.wikipedia.org.

# Приложение А

**MyImage.h**

#pragma once

int showBitMap(HWND hwnd, HANDLE hBit, int x, int y);

int ClientToBmp (HWND hWnd, CString filename, RECT r);

void MyLine(CDC\* dc, int x1, int x2, int y1, int y2, COLORREF color);

void LineX(CDC \* dc, int x1, int y1, int dx, int dy, int dir, COLORREF color);

void LineY(CDC \* dc, int x1, int y1, int dx, int dy, int dir, COLORREF color);

void MyCircle(CDC& dc, CPoint p1, int r, COLORREF color);

void CirclePoint(CDC &dc, int x, int y, CPoint &p, COLORREF color);

**MyImage.cpp**

#include "stdafx.h"

#include "2.h"

#include "ChildView.h"

#include "myImage.h"

#include "math.h"

int showBitMap(HWND hwnd, HANDLE hBit, int x, int y)

{

BITMAP BitMap;

GetObject(hBit, sizeof(BITMAP), &BitMap);

//размеры для окон

int height = BitMap.bmHeight;

int width = BitMap.bmWidth;

//получаем контекст изображения

HDC hdc = GetDC(hwnd);

//создаём контекст памяти

HDC hdcMem = CreateCompatibleDC(hdc);

//в данный контекст памяти заносим дескриптор битовой картыы

HBITMAP oldBitMap = (HBITMAP)SelectObject(hdcMem, hBit);

BitBlt(hdc, x, y, width, height, hdcMem, 0, 0, SRCCOPY);

SelectObject(hdcMem, oldBitMap);

ReleaseDC(hwnd, hdc);

DeleteDC(hdcMem);

return 0;

}

int ClientToBmp (HWND hWnd, CString filename, RECT r)

{

HANDLE fh = CreateFile (filename, GENERIC\_WRITE, 0, NULL, CREATE\_ALWAYS, FILE\_ATTRIBUTE\_NORMAL | FILE\_FLAG\_SEQUENTIAL\_SCAN, NULL);

if (fh == INVALID\_HANDLE\_VALUE)

return 2;

BITMAPINFOHEADER bi;

//заполняем заголовок нулями

ZeroMemory (&bi, sizeof (BITMAPINFOHEADER));

bi.biSize = sizeof (BITMAPINFOHEADER);

bi.biWidth = r.right - r.left; //

bi.biHeight = r.bottom - r.top;//

bi.biPlanes = 1;

bi.biBitCount = 32; //16;

bi.biSizeImage = (bi.biWidth \* bi.biBitCount + 31)/32\*4\*bi.biHeight;

BITMAPFILEHEADER bmfHdr;

ZeroMemory (&bmfHdr, sizeof (BITMAPFILEHEADER));

bmfHdr.bfType = ('M'<<8)|'B'; //заполняем дисковый заголовок

bmfHdr.bfSize = bi.biSizeImage + sizeof (BITMAPFILEHEADER) + bi.biSize;//размер файла

bmfHdr.bfReserved1 = bmfHdr.bfReserved2 = 0;

bmfHdr.bfOffBits = (DWORD)sizeof(BITMAPFILEHEADER) + (DWORD)bi.biSize; //смещение до начала данных

HDC hDC = GetDC (hWnd);

HDC hDCMem = CreateCompatibleDC (hDC);

HBITMAP hBitmap = CreateCompatibleBitmap (hDC, bi.biWidth, bi.biHeight);

HBITMAP oldBitmap = (HBITMAP)SelectObject (hDCMem, hBitmap);

BitBlt (hDCMem, 0,0, bi.biWidth, bi.biHeight, hDC, r.left, r.top, SRCCOPY);

hBitmap = (HBITMAP)SelectObject (hDCMem, oldBitmap);

HANDLE hDIB = GlobalAlloc (GHND /\*GMEM\_FIXED\*/, bi.biSizeImage);

char\* lp = (char\*)GlobalLock (hDIB);//выделяем место

GetDIBits (hDC, hBitmap, 0, bi.biHeight, lp, (LPBITMAPINFO)&bi, DIB\_RGB\_COLORS);

DWORD dwWritten = sizeof (BITMAPFILEHEADER);

WriteFile(fh, (LPSTR)&bmfHdr, sizeof(BITMAPFILEHEADER), &dwWritten, NULL);

dwWritten = sizeof (BITMAPINFOHEADER);

WriteFile (fh, (LPSTR)&bi, sizeof (BITMAPINFOHEADER), &dwWritten, NULL);

dwWritten = bi.biSizeImage;

WriteFile (fh, lp, bi.biSizeImage, &dwWritten, NULL);

GlobalUnlock(hDIB);

GlobalFree(hDIB);

DeleteObject(hBitmap);

lp = NULL;

CloseHandle (fh);

ReleaseDC (hWnd, hDCMem);

ReleaseDC (hWnd, hDC);

DeleteDC (hDCMem);

DeleteDC (hDC);

if (dwWritten == 2) return 2;

return 0;

}

void MyLine(CDC\* dc, int x1, int x2, int y1, int y2, COLORREF color)

{

//если рисуется линия снизу вверх

if(y1 > y2)

{

//меняем координаты местами

int temp;

temp = x1;

x1 = x2;

x2 = temp;

temp = y1;

y1 = y2;

y2 = temp;

}

int dx = x2 - x1;

int dy = y2 - y1;

if(dx > 0)

{

if(dx > dy)//рисуем слева направо

{

LineX(dc, x1, y1, dx, dy, 1, color );

}

else

{

LineY(dc, x1, y1, dx, dy, 1, color);

}

}

else //рисуем справа налево

{

dx = -dx;

if(dx > dy)

{

LineX(dc, x1, y1, dx, dy,-1, color);

}

else

{

LineY(dc, x2, y2, dx, dy,-1, color);

}

}

}

void LineX(CDC \* dc, int x1, int y1, int dx, int dy, int dir, COLORREF color) //рисуем по х

{

int d1 = 2 \* dy;

int d2 = 2 \* (dy - dx);

int d = 2 \* dy - dx;

dc->SetPixel(x1,y1,color);

while(dx--)

{

if(d < 0)

d += d1;

else

{

y1++;

d += d2;

}

x1 += dir;

dc->SetPixel(x1,y1,color);

}

}

void LineY(CDC \* dc, int x1, int y1, int dx, int dy, int dir,COLORREF color)

//рисуем по у

{

int d1 = 2 \* dx;

int d2 = 2 \* (dx - dy);

int d = 2 \* dx - dy;

dc->SetPixel(x1,y1,color);

while(dy--)

{

if(d < 0)

d += d1;

else

{

x1++;

d += d2;

}

y1 += dir;

dc->SetPixel(x1,y1,color);

}

}

void CirclePoints(CDC &dc, int x, int y, CPoint &p, COLORREF color)

{

dc.SetPixel(p.x + x, p.y - y, color);

dc.SetPixel(p.x + x, p.y + y, color);

dc.SetPixel(p.x - x, p.y - y, color);

dc.SetPixel(p.x - x, p.y + y, color);

dc.SetPixel(p.x + y, p.y - x, color);

dc.SetPixel(p.x + y, p.y + x, color);

dc.SetPixel(p.x - y, p.y - x, color);

dc.SetPixel(p.x - y, p.y + x, color);

}

void MyCircle(CDC& dc, CPoint p, int r, COLORREF color)

{

double d = 5/4 - r;

int p1 = 3,

q = p1 - 2 \* r + 2;

int x0 = 0, y0 = r;

CirclePoints(dc, 0, r, p, color);

while(x0 <= r/sqrt((double)2))

{

if(d < 0)

{

CirclePoints(dc, ++x0, y0, p, color);

d = d + p1;

p1 = p1 + 2;

q = q + 2;

}

else

{

CirclePoints(dc, ++x0, --y0, p, color);

d = d + q;

p1 = p1 + 2;

q = q + 4;

}

}

}

**CPlot2D.h**

#ifndef LIBGRAPH

#define LIBGRAPH 1

const double pi=3.14159;

struct CSizeD

{

double cx;

double cy;

};

struct CRectD

{

double left;

double top;

double right;

double bottom;

CRectD()

{

left=top=right=bottom=0;

};

CRectD(double l,double t,double r,double b);

void SetRectD(double l,double t,double r,double b);

CSizeD SizeD(); // Возвращает размеры(ширина, высота) прямоугольника

};

struct CMyPen //перо

{

int PenStyle; //стиль

int PenWidth; //толщина

COLORREF PenColor; //цвет

CMyPen()

{

PenStyle = PS\_SOLID;

PenWidth = 1;

PenColor = RGB(0,0,0);

}

void Set(int PS, int PW, COLORREF PC)

{

PenStyle = PC;

PenWidth = PW;

PenColor = PC;

}

};

//--------------------------

CMatrix SpaceToWindow(CRectD& rs, CRect& rw); //матрица пересчёта

void SetMyMode(CDC& dc,CRectD& rs, CRect& rw); //// Устанавливает режим отображения MM\_ANISOTROPIC и его параметры

//--------------------------

class CPlot2D

{

public:

CMatrix x; //аргумент

CMatrix y; //функция

CMatrix k; //матрица пересчёта координат

CRect RW; //прямоуголник в оконных

CRectD RS; // прямоугольная область в мировых

CMyPen PenLine; //перо для линии

CMyPen PenAxis; //перо для осей

public:

CPlot2D()

{

k.RedimMatrix(3,3);

}

void SetParams(CMatrix& XX, CMatrix& YY, CRect& RWX);

void SetWindowRect(CRect& RWX);

void GetWindowCoords(double xs, double ys, int &xw, int &yw);

void SetPenLine(CMyPen& PLine); //перо для рисования графика

void SetPenAxis(CMyPen& PAxis); //перо для рисования осей

void Draw (CDC& dc, int Ind1, int Ind2); //рисование с самостоятельным пересчётом координат

void Draw1(CDC& dc, int Ind1, int Ind2); //рисование без самостоятельного пересчёта

void GetRs(CRectD& RS); //возвращает область графика в мировых координатах

};

#endif

**CPlot2D.cpp**

#include "stdafx.h"

#include <iostream>

#include <iostream>

CRectD::CRectD(double l,double t,double r,double b)

{

left = l;

top = t;

right = r;

bottom = b;

}

//------------------------------------------------------------------------------

void CRectD::SetRectD(double l,double t,double r,double b)

{

left = l;

top = t;

right = r;

bottom = b;

}

//------------------------------------------------------------------------------

CSizeD CRectD::SizeD()

{

CSizeD cz;

cz.cx=fabs(right-left); // Ширина прямоугольной области

cz.cy=fabs(top-bottom); // Высота прямоугольной области

return cz;

}

//-------------------------------------------------------------------------------

CMatrix SpaceToWindow(CRectD& rs, CRect& rw)

{

CMatrix M(3,3);

CSize s = rw.Size(); //размер в оконных координатах

int wx = s.cx; //в оконных

int wy = s.cy;

CSizeD sd = rs.SizeD(); //размер в мировых координатах

double dx = sd.cx;

double dy = sd.cy;

double kx = (double)wx/dx;

double ky = (double)wy/dy;

M(0,0) = kx;

M(0,1) = 0;

M(0,2) = (double)rw.left-kx\*rs.left;

M(1,0) = 0;

M(1,1) = -ky;

M(1,2) = (double)rw.bottom+ky\*rs.bottom;

M(2,0) = 0;

M(2,1) = 0;

M(2,2) = 1;

return M;

}

//-------------------------------------------------------------------------------

void SetMyMode(CDC& dc,CRectD& rs, CRect& rw)

{

double dsx = rs.right - rs.left;

double dsy = rs.top - rs.bottom;

double xsL = rs.left;

double ysL = rs.bottom;

int dwx = rw.right - rw.left;

int dwy = rw.bottom - rw.top;

int xwL = rw.left;

int ywL = rw.bottom;

dc.SetMapMode(MM\_ANISOTROPIC);

dc.SetWindowExt(dsx,dsy);

dc.SetViewportExt(dwx,-dwy);

dc.SetWindowOrg(xsL,ysL);

dc.SetViewportOrg(xwL,ywL);

}

//---------------------------------------------------------------------------------

void CPlot2D::SetParams(CMatrix& XX, CMatrix& YY, CRect& RWX)

{

//XX - вектор данных по х

//YY - вектор данных по y

//RWX - область в окне

int nRowsX = XX.rows();

int nRowsY = YY.rows();

if (nRowsX!=nRowsY)

{

char\* error = "SetParams - неправильный размер массива данных";

MessageBox(NULL,error,"Ошибка",MB\_ICONSTOP);

exit(0);

}

x.RedimMatrix(nRowsX);

y.RedimMatrix(nRowsY);

x = XX;

y = YY;

double x\_max = x.MaxElement();

double x\_min = x.MinElement();

double y\_max = y.MaxElement();

double y\_min = y.MinElement();

RS.SetRectD(x\_min,y\_max,x\_max,y\_min); //область в мировых

RW.SetRect(RWX.left,RWX.top,RWX.right,RWX.bottom); //область в оконных

k = SpaceToWindow(RS,RW); //матрица пересчёта координат

}

//------------------------------------------------------------------------------------

void CPlot2D::SetWindowRect(CRect& RWX)

{

RW.SetRect(RWX.left,RWX.top,RWX.right,RWX.bottom); //область в оконных

k = SpaceToWindow(RS,RW);

}

//------------------------------------------------------------------------------------------

//установка параметров пера

//для линии

void CPlot2D::SetPenLine(CMyPen& PLine)

{

PenLine.PenStyle = PLine.PenStyle;

PenLine.PenWidth = PLine.PenWidth;

PenLine.PenColor = PLine.PenColor;

}

//для осей

void CPlot2D::SetPenAxis(CMyPen& PAxis)

{

PenAxis.PenStyle = PAxis.PenStyle;

PenAxis.PenWidth = PAxis.PenWidth;

PenAxis.PenColor = PAxis.PenColor;

}

//-------------------------------------------------------------------------------------------------

void CPlot2D::GetRs(CRectD& RS)

{

//RS - структура, куда записывается параметры области графика

RS.left = (this->RS).left;

RS.top = (this->RS).top;

RS.right = (this->RS).right;

RS.bottom = (this->RS).bottom;

}

//---------------------------------------------------------------------------------------------------------

void CPlot2D::GetWindowCoords(double xs, double ys, int &xw, int &yw)

{

CMatrix V(3),W(3);

V(2) = 1;

V(0) = xs;

V(1) = ys;

W = k\*V;

xw = (int)W(0);

yw = (int)W(1);

}

//--------------------------------------------------------------------------------------------------------

void CPlot2D::Draw(CDC& dc, int Ind1, int Ind2)

{

CMatrix V(3), W(3);

V(2) = 1;

if(Ind1==1)

dc.Rectangle(RW);

V(0) = x(0);

V(1) = y(0);

W = k\*V;

CPen MyPen(PenLine.PenStyle, PenLine.PenWidth, PenLine.PenColor);

CPen\* pOldPen = dc.SelectObject(&MyPen);

dc.MoveTo((int)W(0),(int)W(1)); //начальная точка для рисования графика

for(int i = 1; i<x.rows();i++)

{

V(0) = x(i);

V(1) = y(i);

W = k\*V;

dc.LineTo((int)W(0),(int)W(1));

}

dc.SelectObject(pOldPen);

}

//-------------------------------------------------------------------------------

void CPlot2D::Draw1(CDC& dc, int Ind1, int Ind2)

{

CRect IRS (RS.left,RS.top,RS.right,RS.bottom);

if (Ind1 == 1)

dc.Rectangle(IRS);

if(Ind2 == 1) //если нужны оси

{

CPen MyPen(PenAxis.PenStyle, PenAxis.PenWidth, PenAxis.PenColor);

CPen\* pOldPen = dc.SelectObject(&MyPen);

if(RS.left\*RS.right<0) //нужна ось у

{

dc.MoveTo(0,(int)RS.top);

dc.LineTo(0,(int)RS.bottom);

}

if(RS.top\*RS.bottom<0) //нужна ось x

{

dc.MoveTo((int)RS.left,0);

dc.LineTo((int)RS.right,0);

}

dc.SelectObject(pOldPen);

}

CPen MyPen(PenLine.PenStyle,PenLine.PenWidth,PenLine.PenColor);

CPen\* pOldPen= dc.SelectObject(&MyPen);

dc.MoveTo((int)x(0),(int)y(0));

for(int i=1; i<x.rows(); i++)

dc.LineTo((int)x(i),(int)y(i));

dc.SelectObject(pOldPen);

}

/////////------------

**CDialogOptions.h**

#pragma once

#include "afxcmn.h"

// диалоговое окно CDialogOptions

class CDialogOptions : public CDialog

{

DECLARE\_DYNAMIC(CDialogOptions)

public:

CDialogOptions(CWnd\* pParent = NULL); // стандартный конструктор

virtual ~CDialogOptions();

BOOL OnInitDialog();

// Данные диалогового окна

enum { IDD = IDD\_DIALOG1 };

COLORREF COLORLine;

protected:

virtual void DoDataExchange(CDataExchange\* pDX); // поддержка DDX/DDV

DECLARE\_MESSAGE\_MAP()

afx\_msg void ButtonClickedOK();

public:

afx\_msg void OnBnClickedMfcbutton1();

CProgressCtrl LabelColor;

double InputA;

double InputB;

double Inputa;

double Inputb;

int InputQ;

BOOL IfAnim;

int kcouts;

afx\_msg void OnBnClickedCheck1();

double df;

double da;

};

**CDialogOptions.cpp**

// DialogOptions.cpp: файл реализации

//

#include "stdafx.h"

#include "2.h"

#include "DialogOptions.h"

#include "afxdialogex.h"

#include "MainFrm.h"

#include "ChildView.h"

// диалоговое окно CDialogOptions

IMPLEMENT\_DYNAMIC(CDialogOptions, CDialog)

CDialogOptions::CDialogOptions(CWnd\* pParent): CDialog(CDialogOptions::IDD, pParent)

, InputA(0)

, InputB(0)

, Inputa(0)

, Inputb(0)

, InputQ(0)

, IfAnim(FALSE)

, kcouts(0)

, df(0)

, da(0)

{

}

BOOL CDialogOptions::OnInitDialog()

{

CDialog::OnInitDialog();

CProgressCtrl \*prg = (CProgressCtrl\*) GetDlgItem(IDC\_PROGRESS1);

prg->SetBkColor(COLORLine);

CEdit \*ed=(CEdit\*)GetDlgItem(IDC\_EDIT6);

ed->SetReadOnly(false);

//LabelColor.SetBkColor(RGB(255,0,255));

return true;

}

CDialogOptions::~CDialogOptions()

{

}

void CDialogOptions::DoDataExchange(CDataExchange\* pDX)

{

CDialog::DoDataExchange(pDX);

DDX\_Control(pDX, IDC\_PROGRESS1, LabelColor);

DDX\_Text(pDX, IDC\_EDIT1, InputA);

DDX\_Text(pDX, IDC\_EDIT2, InputB);

DDX\_Text(pDX, IDC\_EDIT3, Inputa);

DDX\_Text(pDX, IDC\_EDIT4, Inputb);

DDX\_Text(pDX, IDC\_EDIT5, InputQ);

DDX\_Check(pDX, IDC\_CHECK1, IfAnim);

DDX\_Text(pDX, IDC\_EDIT6, kcouts);

DDX\_Text(pDX, IDC\_EDIT7, df);

DDX\_Text(pDX, IDC\_EDIT8, da);

}

BEGIN\_MESSAGE\_MAP(CDialogOptions, CDialog)

ON\_BN\_CLICKED(IDOK,ButtonClickedOK)

ON\_BN\_CLICKED(IDC\_MFCBUTTON1, &CDialogOptions::OnBnClickedMfcbutton1)

ON\_BN\_CLICKED(IDC\_CHECK1, &CDialogOptions::OnBnClickedCheck1)

END\_MESSAGE\_MAP()

void CDialogOptions::ButtonClickedOK()

{

OnOK();

}

// обработчики сообщений CDialogOptions

void CDialogOptions::OnBnClickedMfcbutton1()

{

CColorDialog GetColorDlg;

if(GetColorDlg.DoModal()==IDOK)

{

COLORLine = GetColorDlg.GetColor();

LabelColor.SetBkColor(COLORLine);

}

}

void CDialogOptions::OnBnClickedCheck1()

{

CEdit \*ed=(CEdit\*)GetDlgItem(IDC\_EDIT6);

if(IfAnim)

{

ed->SetReadOnly(true);

}

else

{

ed->SetReadOnly(false);

}

}

**ChildView.h**

#pragma once

// окно CChildView

class CChildView : public CWnd

{

// Создание

public:

CChildView();

// Атрибуты

public:

// Операции

public:

bool m\_IsMousePressed;

CRect \*m\_SelectedRect;

bool m\_IsImageToPaste;

bool m\_IsGraphSelect1;

bool m\_IsGraphSelect2;

bool m\_IsGraphSelect3;

bool m\_IsGraphSelect4;

bool m\_IsGraphSelect5;

bool m\_IsGraphSelect6;

bool m\_IsGraphSelect7;

CPen pen,bpen;

CBrush brush,bbrush;

int m\_height;

int m\_width;

HANDLE opened\_file;

bool m\_File\_Load;

bool m\_File\_Save;

char\* m\_saved\_file;

int i;

int i1;

int i2;

int i3;

int i4;

int i5;

int i6;

int beginX;

int beginY;

int endX;

int endY;

bool flag;

CPoint p1, p2, pl;

//-------------------------------

int NN;

int iterflag;

int iterflag1;

int iterflag2;

int iterflag3;

int iterflag4;

int iterflag5;

int iterflag6;

double qradtorad(int grad);

double qradtorad1(int grad1);

double qradtorad2(int grad2);

double qradtorad3(int grad3);

double qradtorad4(int grad4);

double qradtorad5(int grad5);

double qradtorad6(int grad6);

afx\_msg void OnRButtonDown(UINT var, CPoint point);

afx\_msg void OnRButtonDown1(UINT var, CPoint point);

afx\_msg void OnRButtonDown2(UINT var, CPoint point);

afx\_msg void OnRButtonDown3(UINT var, CPoint point);

afx\_msg void OnRButtonDown4(UINT var, CPoint point);

afx\_msg void OnRButtonDown5(UINT var, CPoint point);

afx\_msg void OnRButtonDown6(UINT var, CPoint point);

afx\_msg void OnLButtonUp(UINT var, CPoint point);

afx\_msg void OnMouseMove(UINT var, CPoint point);

afx\_msg void OnOpen();

afx\_msg void OnSave();

afx\_msg void Graph1();

afx\_msg void Graph2();

afx\_msg void Graph3();

afx\_msg void Graph4();

afx\_msg void Graph5();

afx\_msg void Graph6();

afx\_msg void Graph7();

afx\_msg void OnTimer( UINT );

void Animate();

void Animate1();

void Animate2();

void Animate3();

void Animate4();

void Animate5();

void Animate6();

void OpenSix();

//---------------------------

afx\_msg void Prop();

double a,b,A,B,df,da;

int grad;

bool IfFirst;

bool IfAnim;

bool pause;

int kcouts;

//----------------------------

afx\_msg void Prop1();

double a1, b1, A1, B1, df1, da1;

int grad1;

bool IfFirst1;

bool IfAnim1;

bool pause1;

int kcouts1;

//----------------------------

afx\_msg void Prop2();

double a2, b2, A2, B2, df2, da2;

int grad2;

bool IfFirst2;

bool IfAnim2;

bool pause2;

int kcouts2;

//----------------------------

afx\_msg void Prop3();

double a3, b3, A3, B3, df3, da3;

int grad3;

bool IfFirst3;

bool IfAnim3;

bool pause3;

int kcouts3;

//----------------------------

afx\_msg void Prop4();

double a4, b4, A4, B4, df4, da4;

int grad4;

bool IfFirst4;

bool IfAnim4;

bool pause4;

int kcouts4;

//----------------------------

afx\_msg void Prop5();

double a5, b5, A5, B5, df5, da5;

int grad5;

bool IfFirst5;

bool IfAnim5;

bool pause5;

int kcouts5;

//----------------------------

afx\_msg void Prop6();

double a6, b6, A6, B6,df6, da6;

int grad6;

bool IfFirst6;

bool IfAnim6;

bool pause6;

int kcouts6;

//-----------------------------

CRectD RS;

CMatrix XX,YY;

CRect RWX, RW;

CPlot2D CPlot1;

CPlot2D CPlot2;

CPlot2D CPlot3;

CPlot2D CPlot4;

CPlot2D CPlot5;

CPlot2D CPlot6;

CMyPen PenLine,PenAxis;

//-----------------------------

// Переопределение

protected:

virtual BOOL PreCreateWindow(CREATESTRUCT& cs);

// Реализация

public:

virtual ~CChildView();

// Созданные функции схемы сообщений

protected:

double pi;

afx\_msg BOOL OnEraseBkgnd(CDC\* pDC);

afx\_msg void OnPaint();

DECLARE\_MESSAGE\_MAP()

};

**ChildView.cpp**

*// ChildView.cpp : реализация класса CChildView*

*#include "stdafx.h"*

*#include "2.h"*

*#include "ChildView.h"*

*#include "MainFrm.h"*

*#include "DialogOptions.h"*

*#include <math.h>*

*#ifdef \_DEBUG*

*#define new DEBUG\_NEW*

*#endif*

*// CChildView*

*CChildView::CChildView()*

*{*

*m\_IsMousePressed = 0;*

*m\_IsImageToPaste = 0;*

*m\_IsGraphSelect1 = 0;*

*m\_IsGraphSelect2 = 0;*

*m\_IsGraphSelect3 = 0;*

*m\_IsGraphSelect4 = 0;*

*m\_IsGraphSelect5 = 0;*

*m\_IsGraphSelect6 = 0;*

*m\_IsGraphSelect7 = 0;*

*m\_SelectedRect = new CRect();*

*pen.CreatePen(PS\_GEOMETRIC,1,RGB(0,255,255));*

*bpen.CreatePen(0,1,RGB(255,255,255));*

*bbrush.CreateSolidBrush(RGB(255,255,255));*

*PenLine.Set(PS\_SOLID,1,RGB(255,0,255));*

*m\_File\_Load = 0;*

*m\_height = 0;*

*m\_width = 0;*

*m\_File\_Save = 0;*

*i1=0;*

*i2 = 0;*

*i3 = 0;*

*i4 = 0;*

*i5 = 0;*

*i6 = 0;*

*i = 0;*

*beginX = 0;*

*beginY = 0;*

*endX = 0;*

*endY = 0;*

*flag=0;*

*//параметры лисажу*

*pi=3.1415926535898;*

*A=1;*

*B=1;*

*a=3;*

*b=3;*

*IfFirst=false;*

*IfAnim=false;*

*pause = false;*

*kcouts=0;*

*grad=45;*

*iterflag=0;*

*df=0.0;*

*da=0.0;*

*//1*

*A1 = 1;*

*B1 = 1;*

*a1 = 3;*

*b1 = 3;*

*IfFirst1 = true;*

*IfAnim1 = false;*

*pause1 = false;*

*kcouts1 = 0;*

*grad1 = 0;*

*df1 = 0.0;*

*da1 = 0.0;*

*iterflag1 = 0;*

*//2*

*A2 = 1;*

*B2 = 1;*

*a2 = 3;*

*b2 = 3;*

*IfFirst2 = true;*

*IfAnim2 = false;*

*pause2 = false;*

*kcouts2 = 0;*

*grad2 = 90;*

*df2 = 0.0;*

*da2 = 0.0;*

*iterflag2 = 0;*

*//3*

*A3 = 1;*

*B3 = 1;*

*a3 = 3;*

*b3 = 3;*

*IfFirst3 = true;*

*IfAnim3 = false;*

*pause3 = false;*

*kcouts3 = 0;*

*grad3 = 45;*

*df3 = 0.0;*

*da3 = 0.0;*

*iterflag3 = 0;*

*//4*

*A4 = 1;*

*B4 = 1;*

*a4 = 6;*

*b4 = 3;*

*IfFirst4 = true;*

*IfAnim4 = false;*

*pause4 = false;*

*kcouts4 = 0;*

*grad4 = 0;*

*df4 = 0.0;*

*da4 = 0.0;*

*iterflag4 = 0;*

*//5*

*A5 = 1;*

*B5 = 1;*

*a5 = 3;*

*b5 = 2;*

*IfFirst5 = true;*

*IfAnim5 = false;*

*pause5 = false;*

*kcouts5 = 0;*

*grad5 = 90;*

*df5 = 0.0;*

*da5 = 0.0;*

*iterflag5 = 0;*

*//6*

*A6 = 1;*

*B6 = 1;*

*a6 = 3;*

*b6 = 4;*

*IfFirst6 = true;*

*IfAnim6 = false;*

*pause6 = false;*

*kcouts6 = 0;*

*grad6 = 90;*

*df6 = 0.0;*

*da6 = 0.0;*

*iterflag6 = 0;*

*}*

*CChildView::~CChildView()*

*{*

*}*

*BEGIN\_MESSAGE\_MAP(CChildView, CWnd)*

*ON\_WM\_TIMER()*

*ON\_WM\_PAINT()*

*ON\_WM\_RBUTTONDOWN()*

*// ON\_WM\_ERASEBKGND()*

*ON\_COMMAND(ID\_32771,OnOpen)*

*ON\_COMMAND(ID\_OPEN\_SIX\_IMG,OpenSix)*

*ON\_COMMAND(ID\_OPTIONS\_SET\_FIRST,Prop1)*

*ON\_COMMAND(ID\_OPTIONS\_SET\_SECOND, Prop2)*

*ON\_COMMAND(ID\_OPTIONS\_SET\_THERD, Prop3)*

*ON\_COMMAND(ID\_OPTIONS\_SET\_FOURTH, Prop4)*

*ON\_COMMAND(ID\_OPTIONS\_SET\_FIFTH, Prop5)*

*ON\_COMMAND(ID\_OPTIONS\_SET\_SIXTH, Prop6)*

*ON\_COMMAND(ID\_32772,OnSave)*

*ON\_COMMAND(ID\_OPTIONS\_SET,Prop)*

*END\_MESSAGE\_MAP()*

*// обработчики сообщений CChildView*

*BOOL CChildView::PreCreateWindow(CREATESTRUCT& cs)*

*{*

*if (!CWnd::PreCreateWindow(cs))*

*return FALSE;*

*cs.dwExStyle |= WS\_EX\_CLIENTEDGE;*

*cs.style &= ~WS\_BORDER;*

*cs.lpszClass = AfxRegisterWndClass(CS\_HREDRAW|CS\_VREDRAW|CS\_DBLCLKS,*

*::LoadCursor(NULL, IDC\_ARROW), reinterpret\_cast<HBRUSH>(COLOR\_WINDOW+1), NULL);*

*return TRUE;*

*}*

*void CChildView::OnRButtonDown( UINT, CPoint cp)*

*{*

*IfAnim=false;*

*KillTimer(1);*

*}*

*void CChildView::OnRButtonDown1(UINT, CPoint cp)*

*{*

*IfAnim1 = false;*

*KillTimer(2);*

*}*

*void CChildView::OnRButtonDown2(UINT, CPoint cp)*

*{*

*IfAnim2 = false;*

*KillTimer(3);*

*}*

*void CChildView::OnRButtonDown3(UINT, CPoint cp)*

*{*

*IfAnim3 = false;*

*KillTimer(4);*

*}*

*void CChildView::OnRButtonDown4(UINT, CPoint cp)*

*{*

*IfAnim4 = false;*

*KillTimer(5);*

*}*

*void CChildView::OnRButtonDown5(UINT, CPoint cp)*

*{*

*IfAnim5 = false;*

*KillTimer(6);*

*}*

*void CChildView::OnRButtonDown6(UINT, CPoint cp)*

*{*

*IfAnim6 = false;*

*KillTimer(7);*

*}*

*BOOL CChildView::OnEraseBkgnd(CDC\* pDC)*

*{*

*return FALSE;*

*}*

*void CChildView::OnTimer( UINT uTime)*

*{*

*iterflag++;*

*iterflag1++;*

*iterflag2++;*

*iterflag3++;*

*iterflag4++;*

*iterflag5++;*

*iterflag6++;*

*if (!IfFirst)*

*{*

*if (iterflag < kcouts)*

*{*

*grad += df;*

*a += da;*

*Graph1();*

*Invalidate();*

*}*

*if (iterflag >= kcouts)*

*{*

*IfAnim = false;*

*KillTimer(1);*

*iterflag = 0;*

*Invalidate();*

*}*

*}*

*if (!IfFirst1)*

*{*

*if (iterflag1 < kcouts1)*

*{*

*grad1 += df1;*

*a1 += da1;*

*Graph2();*

*Invalidate();*

*}*

*if (iterflag1 >= kcouts1)*

*{*

*IfAnim1 = false;*

*KillTimer(2);*

*iterflag1 = 0;*

*Invalidate();*

*}*

*}*

*if (!IfFirst2)*

*{*

*if (iterflag2 < kcouts2)*

*{*

*grad2 += df2;*

*a2 += da2;*

*Graph3();*

*Invalidate();*

*}*

*if (iterflag2 >= kcouts2)*

*{*

*IfAnim2 = false;*

*KillTimer(3);*

*iterflag2 = 0;*

*Invalidate();*

*}*

*}*

*if (!IfFirst3)*

*{*

*if (iterflag3 < kcouts3)*

*{*

*grad3 += df3;*

*a3 += da3;*

*Graph4();*

*Invalidate();*

*}*

*if (iterflag3 >= kcouts3)*

*{*

*IfAnim3 = false;*

*KillTimer(4);*

*iterflag3 = 0;*

*Invalidate();*

*}*

*}*

*if (!IfFirst4)*

*{*

*if (iterflag4 < kcouts4)*

*{*

*grad4 += df4;*

*a4 += da4;*

*Graph5();*

*Invalidate();*

*}*

*if (iterflag4 >= kcouts4)*

*{*

*IfAnim4 = false;*

*KillTimer(5);*

*iterflag4 = 0;*

*Invalidate();*

*}*

*}*

*if (!IfFirst5)*

*{*

*if (iterflag5 < kcouts5)*

*{*

*grad5 += df5;*

*a5 += da5;*

*Graph6();*

*Invalidate();*

*}*

*if (iterflag5 >= kcouts5)*

*{*

*IfAnim5 = false;*

*KillTimer(6);*

*iterflag5 = 0;*

*Invalidate();*

*}*

*}*

*if (!IfFirst6)*

*{*

*if (iterflag6 < kcouts6)*

*{*

*grad6 += df6;*

*a6 += da6;*

*Graph7();*

*Invalidate();*

*}*

*if (iterflag6 >= kcouts6)*

*{*

*IfAnim6 = false;*

*KillTimer(7);*

*iterflag6 = 0;*

*Invalidate();*

*}*

*}*

*}*

*void CChildView::OnPaint()*

*{*

*CPaintDC dc(this);*

*if(m\_File\_Load)*

*{*

*showBitMap(dc.GetWindow()->m\_hWnd,opened\_file,0,0);*

*}*

*if(IfFirst)*

*{*

*Graph1();*

*}*

*if (IfFirst1)*

*{*

*Graph2();*

*}*

*if (IfFirst2)*

*{*

*Graph3();*

*}*

*if (IfFirst3)*

*{*

*Graph4();*

*}*

*if (IfFirst4)*

*{*

*Graph5();*

*}*

*if (IfFirst5)*

*{*

*Graph6();*

*}*

*if (IfFirst6)*

*{*

*Graph7();*

*}*

*if(m\_IsGraphSelect1)*

*{*

*CPlot1.Draw(dc,1,1);*

*}*

*if (m\_IsGraphSelect2)*

*{*

*CPlot1.Draw(dc, 1, 1);*

*}*

*if (m\_IsGraphSelect3)*

*{*

*CPlot2.Draw(dc, 1, 1);*

*}*

*if (m\_IsGraphSelect4)*

*{*

*CPlot3.Draw(dc, 1, 1);*

*}*

*if (m\_IsGraphSelect5)*

*{*

*CPlot4.Draw(dc, 1, 1);*

*}*

*if (m\_IsGraphSelect6)*

*{*

*CPlot5.Draw(dc, 1, 1);*

*}*

*if (m\_IsGraphSelect7)*

*{*

*CPlot6.Draw(dc, 1, 1);*

*}*

*}*

*void CChildView::OnOpen()*

*{*

*CFileDialog fd(TRUE,NULL,"");*

*fd.m\_ofn.lpstrFilter = "BMP\0 \*.bmp\0\0";*

*if(fd.DoModal()==IDOK)*

*{*

*CWnd \*cwnd = AfxGetMainWnd();*

*HWND hwnd = cwnd->GetSafeHwnd();*

*HBITMAP hBit = (HBITMAP)LoadImage(NULL, fd.GetPathName().GetBuffer(), IMAGE\_BITMAP, 0, 0, LR\_LOADFROMFILE | LR\_CREATEDIBSECTION);*

*opened\_file=hBit;*

*showBitMap(hwnd,hBit,0,0);*

*BITMAP BitMap;*

*GetObject(hBit, sizeof(BITMAP), &BitMap);*

*m\_File\_Load = 1;*

*m\_height = BitMap.bmHeight;*

*m\_width = BitMap.bmWidth;*

*if(i>0)*

*{*

*OnPaint();*

*}*

*i++;*

*}*

*Invalidate();*

*}*

*void CChildView::OnSave()*

*{*

*CFileDialog fileDialog(FALSE, \_T("fif"), 0, 6UL,NULL , this);*

*fileDialog.m\_ofn.lpstrFilter = "BMP\0 \*.bmp\0\0";*

*int result = fileDialog.DoModal();*

*if (result==IDOK)*

*{*

*CWnd \*cwnd = AfxGetMainWnd();*

*HWND hwnd = cwnd->GetSafeHwnd();*

*RECT r;*

*r.bottom= RW.bottom;*

*r.left=RW.left;*

*r.right=RW.right;*

*r.top=RW.top;*

*ClientToBmp(hwnd,fileDialog.GetPathName(),r );*

*}*

*}*

*void CChildView::Graph1()*

*{*

*IfFirst=false;*

*m\_IsGraphSelect1 = true;*

*m\_IsGraphSelect2 = false;*

*m\_IsGraphSelect3 = false;*

*m\_IsGraphSelect4 = false;*

*m\_IsGraphSelect5 = false;*

*m\_IsGraphSelect6 = false;*

*m\_IsGraphSelect7 = false;*

*int N=0.0;*

*if(a==0.0 || b==0.0)*

*{*

*N=360\*100;*

*}*

*else*

*{*

*N=ceil(360/a)\*100;*

*}*

*if(N>100000)*

*N=100000;*

*XX.RedimMatrix(N+1);*

*YY.RedimMatrix(N+1);*

*double f=0.0;*

*double trololo,trol;*

*for(int i =0; i <=N; i++,f+=0.01)*

*{*

*trololo=A\*sin(a\*f + qradtorad(grad));*

*trol=B\*sin(b\*f);*

*XX(i)=trololo;*

*YY(i)=trol;*

*}*

*PenAxis.Set(PS\_SOLID,1,RGB(0,0,0));*

*RW.SetRect(90,45,530,500);*

*CPlot1.SetParams(XX,YY,RW);*

*CPlot1.SetPenLine(PenLine);*

*CPlot1.SetPenAxis(PenAxis);*

*}*

*void CChildView::Animate()*

*{*

*Graph1();*

*Invalidate();*

*}*

*void CChildView::Graph2()*

*{*

*IfFirst1 = false;*

*m\_IsGraphSelect1 = false;*

*m\_IsGraphSelect2 = true;*

*m\_IsGraphSelect3 = true;*

*m\_IsGraphSelect4 = true;*

*m\_IsGraphSelect5 = true;*

*m\_IsGraphSelect6 = true;*

*m\_IsGraphSelect7 = true;*

*int N = 0.0;*

*if (a1 == 0.0 || b1 == 0.0)*

*{*

*N = 360 \* 100;*

*}*

*else*

*{*

*N = ceil(360 / a1) \* 100;*

*}*

*if (N > 100000)*

*N = 100000;*

*XX.RedimMatrix(N + 1);*

*YY.RedimMatrix(N + 1);*

*double f = 0.0;*

*double trololo, trol;*

*for (int i1 = 0; i1 <= N; i1++, f += 0.01)*

*{*

*trololo = A1 \* sin(a1 \* f + qradtorad1(grad1));*

*trol = B1 \* sin(b1 \* f);*

*XX(i1) = trololo;*

*YY(i1) = trol;*

*}*

*PenAxis.Set(PS\_SOLID, 1, RGB(0, 0, 0));*

*RW.SetRect(40, 45, 220, 225);*

*CPlot1.SetParams(XX, YY, RW);*

*CPlot1.SetPenLine(PenLine);*

*CPlot1.SetPenAxis(PenAxis);*

*}*

*void CChildView::Animate1()*

*{*

*Graph2();*

*Invalidate();*

*}*

*void CChildView::Graph3()*

*{*

*IfFirst2 = false;*

*m\_IsGraphSelect1 = false;*

*m\_IsGraphSelect2 = true;*

*m\_IsGraphSelect3 = true;*

*m\_IsGraphSelect4 = true;*

*m\_IsGraphSelect5 = true;*

*m\_IsGraphSelect6 = true;*

*m\_IsGraphSelect7 = true;*

*int N = 0.0;*

*if (a2 == 0.0 || b2 == 0.0)*

*{*

*N = 360 \* 100;*

*}*

*else*

*{*

*N = ceil(360 / a2) \* 100;*

*}*

*if (N > 100000)*

*N = 100000;*

*XX.RedimMatrix(N + 1);*

*YY.RedimMatrix(N + 1);*

*double f = 0.0;*

*double trololo, trol;*

*for (int i2 = 0; i2 <= N; i2++, f += 0.01)*

*{*

*trololo = A2 \* sin(a2 \* f + qradtorad2(grad2));*

*trol = B2 \* sin(b2 \* f);*

*XX(i2) = trololo;*

*YY(i2) = trol;*

*}*

*PenAxis.Set(PS\_SOLID, 1, RGB(0, 0, 0));*

*RW.SetRect(340, 45, 520, 225);*

*CPlot2.SetParams(XX, YY, RW);*

*CPlot2.SetPenLine(PenLine);*

*CPlot2.SetPenAxis(PenAxis);*

*}*

*void CChildView::Animate2()*

*{*

*Graph3();*

*Invalidate();*

*}*

*void CChildView::Graph4()*

*{*

*IfFirst3 = false;*

*m\_IsGraphSelect1 = false;*

*m\_IsGraphSelect2 = true;*

*m\_IsGraphSelect3 = true;*

*m\_IsGraphSelect4 = true;*

*m\_IsGraphSelect5 = true;*

*m\_IsGraphSelect6 = true;*

*m\_IsGraphSelect7 = true;*

*int N = 0.0;*

*if (a3 == 0.0 || b3 == 0.0)*

*{*

*N = 360 \* 100;*

*}*

*else*

*{*

*N = ceil(360 / a3) \* 100;*

*}*

*if (N > 100000)*

*N = 100000;*

*XX.RedimMatrix(N + 1);*

*YY.RedimMatrix(N + 1);*

*double f = 0.0;*

*double trololo, trol;*

*for (int i3 = 0; i3 <= N; i3++, f += 0.01)*

*{*

*trololo = A3 \* sin(a3 \* f + qradtorad3(grad3));*

*trol = B3 \* sin(b3 \* f);*

*XX(i3) = trololo;*

*YY(i3) = trol;*

*}*

*PenAxis.Set(PS\_SOLID, 1, RGB(0, 0, 0));*

*RW.SetRect(640, 45, 820, 225);*

*CPlot3.SetParams(XX, YY, RW);*

*CPlot3.SetPenLine(PenLine);*

*CPlot3.SetPenAxis(PenAxis);*

*}*

*void CChildView::Animate3()*

*{*

*Graph4();*

*Invalidate();*

*}*

*void CChildView::Graph5()*

*{*

*IfFirst4 = false;*

*m\_IsGraphSelect1 = false;*

*m\_IsGraphSelect2 = true;*

*m\_IsGraphSelect3 = true;*

*m\_IsGraphSelect4 = true;*

*m\_IsGraphSelect5 = true;*

*m\_IsGraphSelect6 = true;*

*m\_IsGraphSelect7 = true;*

*int N = 0.0;*

*if (a4 == 0.0 || b4 == 0.0)*

*{*

*N = 360 \* 100;*

*}*

*else*

*{*

*N = ceil(360 / a4) \* 100;*

*}*

*if (N > 100000)*

*N = 100000;*

*XX.RedimMatrix(N + 1);*

*YY.RedimMatrix(N + 1);*

*double f = 0.0;*

*double trololo, trol;*

*for (int i4 = 0; i4 <= N; i4++, f += 0.01)*

*{*

*trololo = A4 \* sin(a4 \* f + qradtorad4(grad4));*

*trol = B4 \* sin(b4 \* f);*

*XX(i4) = trololo;*

*YY(i4) = trol;*

*}*

*PenAxis.Set(PS\_SOLID, 1, RGB(0, 0, 0));*

*RW.SetRect(40, 270, 220, 450);*

*CPlot4.SetParams(XX, YY, RW);*

*CPlot4.SetPenLine(PenLine);*

*CPlot4.SetPenAxis(PenAxis);*

*}*

*void CChildView::Animate4()*

*{*

*Graph5();*

*Invalidate();*

*}*

*void CChildView::Graph6()*

*{*

*IfFirst5 = false;*

*m\_IsGraphSelect1 = false;*

*m\_IsGraphSelect2 = true;*

*m\_IsGraphSelect3 = true;*

*m\_IsGraphSelect4 = true;*

*m\_IsGraphSelect5 = true;*

*m\_IsGraphSelect6 = true;*

*m\_IsGraphSelect7 = true;*

*int N = 0.0;*

*if (a5 == 0.0 || b5 == 0.0)*

*{*

*N = 360 \* 100;*

*}*

*else*

*{*

*N = ceil(360 / a5) \* 100;*

*}*

*if (N > 100000)*

*N = 100000;*

*XX.RedimMatrix(N + 1);*

*YY.RedimMatrix(N + 1);*

*double f = 0.0;*

*double trololo, trol;*

*for (int i5 = 0; i5 <= N; i5++, f += 0.01)*

*{*

*trololo = A5 \* sin(a5 \* f + qradtorad5(grad5));*

*trol = B5 \* sin(b5 \* f);*

*XX(i5) = trololo;*

*YY(i5) = trol;*

*}*

*PenAxis.Set(PS\_SOLID, 1, RGB(0, 0, 0));*

*RW.SetRect(340, 270, 520, 450);*

*CPlot5.SetParams(XX, YY, RW);*

*CPlot5.SetPenLine(PenLine);*

*CPlot5.SetPenAxis(PenAxis);*

*}*

*void CChildView::Animate5()*

*{*

*Graph6();*

*Invalidate();*

*}*

*void CChildView::Graph7()*

*{*

*IfFirst6 = false;*

*m\_IsGraphSelect1 = false;*

*m\_IsGraphSelect2 = true;*

*m\_IsGraphSelect3 = true;*

*m\_IsGraphSelect4 = true;*

*m\_IsGraphSelect5 = true;*

*m\_IsGraphSelect6 = true;*

*m\_IsGraphSelect7 = true;*

*int N = 0.0;*

*if (a6 == 0.0 || b6 == 0.0)*

*{*

*N = 360 \* 100;*

*}*

*else*

*{*

*N = ceil(360 / a6) \* 100;*

*}*

*if (N > 100000)*

*N = 100000;*

*XX.RedimMatrix(N + 1);*

*YY.RedimMatrix(N + 1);*

*double f = 0.0;*

*double trololo, trol;*

*for (int i6 = 0; i6 <= N; i6++, f += 0.01)*

*{*

*trololo = A6 \* sin(a6 \* f + qradtorad6(grad6));*

*trol = B6 \* sin(b6 \* f);*

*XX(i6) = trololo;*

*YY(i6) = trol;*

*}*

*PenAxis.Set(PS\_SOLID, 1, RGB(0, 0, 0));*

*RW.SetRect(640, 270, 820, 450);*

*CPlot6.SetParams(XX, YY, RW);*

*CPlot6.SetPenLine(PenLine);*

*CPlot6.SetPenAxis(PenAxis);*

*}*

*void CChildView::Animate6()*

*{*

*Graph7();*

*Invalidate();*

*}*

*void CChildView::Prop()*

*{*

*CDialogOptions DialOpt;*

*DialOpt.COLORLine=PenLine.PenColor;*

*DialOpt.InputA=A;*

*DialOpt.InputB=B;*

*DialOpt.Inputa=a;*

*DialOpt.Inputb=b;*

*DialOpt.kcouts=kcouts;*

*DialOpt.InputQ=grad;*

*DialOpt.df=df;*

*DialOpt.da=da;*

*if(DialOpt.DoModal()==IDOK)*

*{*

*PenLine.PenColor=DialOpt.COLORLine;*

*A=DialOpt.InputA;*

*B=DialOpt.InputB;*

*a=DialOpt.Inputa;*

*b=DialOpt.Inputb;*

*grad=DialOpt.InputQ;*

*IfAnim=DialOpt.IfAnim;*

*kcouts=DialOpt.kcouts;*

*df=DialOpt.df;*

*da=DialOpt.da;*

*if(IfAnim)*

*{*

*SetTimer(1,100,NULL);*

*}*

*Animate();*

*}*

*}*

*void CChildView::Prop1()*

*{*

*CDialogOptions DialOpt1;*

*DialOpt1.COLORLine = PenLine.PenColor;*

*DialOpt1.InputA = A1;*

*DialOpt1.InputB = B1;*

*DialOpt1.Inputa = a1;*

*DialOpt1.Inputb = b1;*

*DialOpt1.kcouts = kcouts1;*

*DialOpt1.InputQ = grad1;*

*DialOpt1.df = df1;*

*DialOpt1.da = da1;*

*if (DialOpt1.DoModal() == IDOK)*

*{*

*PenLine.PenColor = DialOpt1.COLORLine;*

*A1 = DialOpt1.InputA;*

*B1 = DialOpt1.InputB;*

*a1 = DialOpt1.Inputa;*

*b1 = DialOpt1.Inputb;*

*grad1 = DialOpt1.InputQ;*

*IfAnim1 = DialOpt1.IfAnim;*

*kcouts1 = DialOpt1.kcouts;*

*df1 = DialOpt1.df;*

*da1 = DialOpt1.da;*

*if (IfAnim1)*

*{*

*SetTimer(2, 0, NULL);*

*}*

*Animate1();*

*}*

*}*

*void CChildView::Prop2()*

*{*

*CDialogOptions DialOpt2;*

*DialOpt2.COLORLine = PenLine.PenColor;*

*DialOpt2.InputA = A2;*

*DialOpt2.InputB = B2;*

*DialOpt2.Inputa = a2;*

*DialOpt2.Inputb = b2;*

*DialOpt2.kcouts = kcouts2;*

*DialOpt2.InputQ = grad2;*

*DialOpt2.df = df2;*

*DialOpt2.da = da2;*

*if (DialOpt2.DoModal() == IDOK)*

*{*

*PenLine.PenColor = DialOpt2.COLORLine;*

*A2 = DialOpt2.InputA;*

*B2 = DialOpt2.InputB;*

*a2 = DialOpt2.Inputa;*

*b2 = DialOpt2.Inputb;*

*grad2 = DialOpt2.InputQ;*

*IfAnim2 = DialOpt2.IfAnim;*

*kcouts2 = DialOpt2.kcouts;*

*df2 = DialOpt2.df;*

*da2 = DialOpt2.da;*

*if (IfAnim2)*

*{*

*SetTimer(3, 0, NULL);*

*}*

*Animate2();*

*}*

*}*

*void CChildView::Prop3()*

*{*

*CDialogOptions DialOpt3;*

*DialOpt3.COLORLine = PenLine.PenColor;*

*DialOpt3.InputA = A3;*

*DialOpt3.InputB = B3;*

*DialOpt3.Inputa = a3;*

*DialOpt3.Inputb = b3;*

*DialOpt3.kcouts = kcouts3;*

*DialOpt3.InputQ = grad3;*

*DialOpt3.df = df3;*

*DialOpt3.da = da3;*

*if (DialOpt3.DoModal() == IDOK)*

*{*

*PenLine.PenColor = DialOpt3.COLORLine;*

*A3 = DialOpt3.InputA;*

*B3 = DialOpt3.InputB;*

*a3 = DialOpt3.Inputa;*

*b3 = DialOpt3.Inputb;*

*grad3 = DialOpt3.InputQ;*

*IfAnim3 = DialOpt3.IfAnim;*

*kcouts3 = DialOpt3.kcouts;*

*df3 = DialOpt3.df;*

*da3 = DialOpt3.da;*

*if (IfAnim3)*

*{*

*SetTimer(4, 0, NULL);*

*}*

*Animate3();*

*}*

*}*

*void CChildView::Prop4()*

*{*

*CDialogOptions DialOpt4;*

*DialOpt4.COLORLine = PenLine.PenColor;*

*DialOpt4.InputA = A4;*

*DialOpt4.InputB = B4;*

*DialOpt4.Inputa = a4;*

*DialOpt4.Inputb = b4;*

*DialOpt4.kcouts = kcouts4;*

*DialOpt4.InputQ = grad4;*

*DialOpt4.df = df4;*

*DialOpt4.da = da4;*

*if (DialOpt4.DoModal() == IDOK)*

*{*

*PenLine.PenColor = DialOpt4.COLORLine;*

*A4 = DialOpt4.InputA;*

*B4 = DialOpt4.InputB;*

*a4 = DialOpt4.Inputa;*

*b4 = DialOpt4.Inputb;*

*grad4 = DialOpt4.InputQ;*

*IfAnim4 = DialOpt4.IfAnim;*

*kcouts4 = DialOpt4.kcouts;*

*df4 = DialOpt4.df;*

*da4 = DialOpt4.da;*

*if (IfAnim4)*

*{*

*SetTimer(5, 0, NULL);*

*}*

*Animate4();*

*}*

*}*

*void CChildView::Prop5()*

*{*

*CDialogOptions DialOpt5;*

*DialOpt5.COLORLine = PenLine.PenColor;*

*DialOpt5.InputA = A5;*

*DialOpt5.InputB = B5;*

*DialOpt5.Inputa = a5;*

*DialOpt5.Inputb = b5;*

*DialOpt5.kcouts = kcouts5;*

*DialOpt5.InputQ = grad5;*

*DialOpt5.df = df5;*

*DialOpt5.da = da5;*

*if (DialOpt5.DoModal() == IDOK)*

*{*

*PenLine.PenColor = DialOpt5.COLORLine;*

*A5 = DialOpt5.InputA;*

*B5 = DialOpt5.InputB;*

*a5 = DialOpt5.Inputa;*

*b5 = DialOpt5.Inputb;*

*grad5 = DialOpt5.InputQ;*

*IfAnim5 = DialOpt5.IfAnim;*

*kcouts5 = DialOpt5.kcouts;*

*df5 = DialOpt5.df;*

*da5 = DialOpt5.da;*

*if (IfAnim5)*

*{*

*SetTimer(6, 0, NULL);*

*}*

*Animate5();*

*}*

*}*

*void CChildView::Prop6()*

*{*

*CDialogOptions DialOpt6;*

*DialOpt6.COLORLine = PenLine.PenColor;*

*DialOpt6.InputA = A6;*

*DialOpt6.InputB = B6;*

*DialOpt6.Inputa = a6;*

*DialOpt6.Inputb = b6;*

*DialOpt6.kcouts = kcouts6;*

*DialOpt6.InputQ = grad6;*

*DialOpt6.df = df6;*

*DialOpt6.da = da6;*

*if (DialOpt6.DoModal() == IDOK)*

*{*

*PenLine.PenColor = DialOpt6.COLORLine;*

*A6 = DialOpt6.InputA;*

*B6 = DialOpt6.InputB;*

*a6 = DialOpt6.Inputa;*

*b6 = DialOpt6.Inputb;*

*grad6 = DialOpt6.InputQ;*

*IfAnim6 = DialOpt6.IfAnim;*

*kcouts6 = DialOpt6.kcouts;*

*df6 = DialOpt6.df;*

*da6 = DialOpt6.da;*

*if (IfAnim6)*

*{*

*SetTimer(7, 0, NULL);*

*}*

*Animate6();*

*}*

*}*

*void CChildView::OpenSix()*

*{*

*}*

*double CChildView::qradtorad(int grad)*

*{*

*double intpart,rad;*

*rad=0.0;*

*rad=modf(grad/360,&intpart);*

*if (abs(intpart)>0)*

*{*

*if (grad>0) grad=grad-360\*intpart;*

*else grad=grad+360\*intpart;*

*}*

*if (grad<0) grad=(grad+360);*

*if ((grad>=0) && (grad<180)) rad=grad\*pi/180;*

*if ((grad>=180) && (grad<360)) rad=(grad-180)\*pi/180;*

*return rad;*

*}*

*double CChildView::qradtorad1(int grad1)*

*{*

*double intpart, rad;*

*rad = 0.0;*

*rad = modf(grad1 / 360, &intpart);*

*if (abs(intpart) > 0)*

*{*

*if (grad1 > 0) grad1 = grad1 - 360 \* intpart;*

*else grad1 = grad1 + 360 \* intpart;*

*}*

*if (grad1 < 0) grad1 = (grad1 + 360);*

*if ((grad1 >= 0) && (grad1 < 180)) rad = grad1 \* pi / 180;*

*if ((grad1 >= 180) && (grad1 < 360)) rad = (grad1 - 180) \* pi / 180;*

*return rad;*

*}*

*double CChildView::qradtorad2(int grad2)*

*{*

*double intpart, rad;*

*rad = 0.0;*

*rad = modf(grad2 / 360, &intpart);*

*if (abs(intpart) > 0)*

*{*

*if (grad2 > 0) grad2 = grad2 - 360 \* intpart;*

*else grad2 = grad2 + 360 \* intpart;*

*}*

*if (grad2 < 0) grad2 = (grad2 + 360);*

*if ((grad2 >= 0) && (grad2 < 180)) rad = grad2 \* pi / 180;*

*if ((grad2 >= 180) && (grad2 < 360)) rad = (grad2 - 180) \* pi / 180;*

*return rad;*

*}*

*double CChildView::qradtorad3(int grad3)*

*{*

*double intpart, rad;*

*rad = 0.0;*

*rad = modf(grad3 / 360, &intpart);*

*if (abs(intpart) > 0)*

*{*

*if (grad3 > 0) grad3 = grad3 - 360 \* intpart;*

*else grad3 = grad3 + 360 \* intpart;*

*}*

*if (grad3 < 0) grad3 = (grad3 + 360);*

*if ((grad3 >= 0) && (grad3 < 180)) rad = grad3 \* pi / 180;*

*if ((grad3 >= 180) && (grad3 < 360)) rad = (grad3 - 180) \* pi / 180;*

*return rad;*

*}*

*double CChildView::qradtorad4(int grad4)*

*{*

*double intpart, rad;*

*rad = 0.0;*

*rad = modf(grad4 / 360, &intpart);*

*if (abs(intpart) > 0)*

*{*

*if (grad4 > 0) grad4 = grad4 - 360 \* intpart;*

*else grad4 = grad4 + 360 \* intpart;*

*}*

*if (grad4 < 0) grad4 = (grad4 + 360);*

*if ((grad4 >= 0) && (grad4 < 180)) rad = grad4 \* pi / 180;*

*if ((grad4 >= 180) && (grad4 < 360)) rad = (grad4 - 180) \* pi / 180;*

*return rad;*

*}*

*double CChildView::qradtorad5(int grad5)*

*{*

*double intpart, rad;*

*rad = 0.0;*

*rad = modf(grad5 / 360, &intpart);*

*if (abs(intpart) > 0)*

*{*

*if (grad5 > 0) grad5 = grad5 - 360 \* intpart;*

*else grad5 = grad5 + 360 \* intpart;*

*}*

*if (grad5 < 0) grad5 = (grad5 + 360);*

*if ((grad5 >= 0) && (grad5 < 180)) rad = grad5 \* pi / 180;*

*if ((grad5 >= 180) && (grad5 < 360)) rad = (grad5 - 180) \* pi / 180;*

*return rad;*

*}*

*double CChildView::qradtorad6(int grad6)*

*{*

*double intpart, rad;*

*rad = 0.0;*

*rad = modf(grad6 / 360, &intpart);*

*if (abs(intpart) > 0)*

*{*

*if (grad6 > 0) grad6 = grad6 - 360 \* intpart;*

*else grad6 = grad6 + 360 \* intpart;*

*}*

*if (grad6 < 0) grad6 = (grad6 + 360);*

*if ((grad6 >= 0) && (grad6 < 180)) rad = grad6 \* pi / 180;*

*if ((grad6 >= 180) && (grad6 < 360)) rad = (grad6 - 180) \* pi / 180;*

*return rad;*

*}*

**MainFmr.h**

// MainFrm.h : интерфейс класса CMainFrame

//

#pragma once

#include "ChildView.h"

class CMainFrame : public CFrameWnd

{

public:

CMainFrame();

protected:

DECLARE\_DYNAMIC(CMainFrame)

// Атрибуты

public:

int inputk;

// Операции

public:

// Переопределение

public:

virtual BOOL PreCreateWindow(CREATESTRUCT& cs);

virtual BOOL OnCmdMsg(UINT nID, int nCode, void\* pExtra, AFX\_CMDHANDLERINFO\* pHandlerInfo);

// Реализация

public:

virtual ~CMainFrame();

#ifdef \_DEBUG

virtual void AssertValid() const;

virtual void Dump(CDumpContext& dc) const;

#endif

protected: // встроенные члены панели элементов управления

CStatusBar m\_wndStatusBar;

CChildView m\_wndView;

// Созданные функции схемы сообщений

protected:

afx\_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);

afx\_msg void OnSetFocus(CWnd \*pOldWnd);

DECLARE\_MESSAGE\_MAP()

};

**MainFmr.cpp**

// MainFrm.cpp : реализация класса CMainFrame

//

#include "stdafx.h"

#include "2.h"

#include "MainFrm.h"

#ifdef \_DEBUG

#define new DEBUG\_NEW

#endif

// CMainFrame

IMPLEMENT\_DYNAMIC(CMainFrame, CFrameWnd)

BEGIN\_MESSAGE\_MAP(CMainFrame, CFrameWnd)

ON\_WM\_CREATE()

ON\_WM\_SETFOCUS()

END\_MESSAGE\_MAP()

static UINT indicators[] =

{

ID\_SEPARATOR, // индикатор строки состояния

ID\_INDICATOR\_CAPS,

ID\_INDICATOR\_NUM,

ID\_INDICATOR\_SCRL,

};

// создание/уничтожение CMainFrame

CMainFrame::CMainFrame()

{

inputk=0;

}

CMainFrame::~CMainFrame()

{

}

int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)

{

if (CFrameWnd::OnCreate(lpCreateStruct) == -1)

return -1;

// создать представление для размещения рабочей области рамки

if (!m\_wndView.Create(NULL, NULL, AFX\_WS\_DEFAULT\_VIEW, CRect(0, 0, 0, 0), this, AFX\_IDW\_PANE\_FIRST, NULL))

{

TRACE0("Не удалось создать окно представлений\n");

return -1;

}

if (!m\_wndStatusBar.Create(this))

{

TRACE0("Не удалось создать строку состояния\n");

return -1; // не удалось создать

}

m\_wndStatusBar.SetIndicators(indicators, sizeof(indicators)/sizeof(UINT));

return 0;

}

BOOL CMainFrame::PreCreateWindow(CREATESTRUCT& cs)

{

if( !CFrameWnd::PreCreateWindow(cs) )

return FALSE;

// TODO: изменить класс Window или стили посредством изменения

// CREATESTRUCT cs

cs.cx = 880;

cs.cy = 630;

cs.x = 10;

cs.y = 10;

cs.dwExStyle &= ~WS\_EX\_CLIENTEDGE;

cs.lpszClass = AfxRegisterWndClass(0);

return TRUE;

}

// диагностика CMainFrame

#ifdef \_DEBUG

void CMainFrame::AssertValid() const

{

CFrameWnd::AssertValid();

}

void CMainFrame::Dump(CDumpContext& dc) const

{

CFrameWnd::Dump(dc);

}

#endif //\_DEBUG

// обработчики сообщений CMainFrame

void CMainFrame::OnSetFocus(CWnd\* /\*pOldWnd\*/)

{

// передача фокуса окну представления

m\_wndView.SetFocus();

}

BOOL CMainFrame::OnCmdMsg(UINT nID, int nCode, void\* pExtra, AFX\_CMDHANDLERINFO\* pHandlerInfo)

{

// разрешить ошибки в представлении при выполнении команды

if (m\_wndView.OnCmdMsg(nID, nCode, pExtra, pHandlerInfo))

return TRUE;

// в противном случае выполняется обработка по умолчанию

return CFrameWnd::OnCmdMsg(nID, nCode, pExtra, pHandlerInfo);

}