Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Комсомольский-на-Амуре государственный университет»

Факультет компьютерных технологий

Кафедра ПУРИС

Лабораторная работа №2

по дисциплине «Компьютерная графика»

Студент группы 0ВТб-1 Н.Д. Малышев

Преподаватель В.А. Тихомиров

2023

1 Разработка приложения

Программно реализовать целочисленные алгоритмы Брезенхема для вывода на экран фигуру, заданную на рисунке 1.1.

Рисунок 1.1 – Фигура

В листинге 1.1 показан код, который реализует графическое приложение с использованием WinAPI для отображения окружности и диагональных линий, используя алгоритм Брезенхема для рисования.

Листинг 1.1 – Lab_2.cpp

```
#include "framework.h"
#include "Lab_2.h"
#include <cmath>

#define MAX_LOADSTRING 100

// Глобальные переменные:
HINSTANCE hInst;                                // Текущий
экземпляр приложения
WCHAR szTitle[MAX_LOADSTRING];                  // Текст
заголовка окна
WCHAR szWindowClass[MAX_LOADSTRING];            // Имя класса
главного окна

ATOM MyRegisterClass(HINSTANCE hInstance);
BOOL InitInstance(HINSTANCE, int);
LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);
INT_PTR CALLBACK About(HWND, UINT, WPARAM, LPARAM);

// Прототипы функций для рисования линии и окружности
void DrawLine(HDC hdc, int x1, int y1, int x2, int y2, COLORREF
color);
void DrawCircle(HDC hdc, int x, int y, int r, COLORREF color);
```

```
// Главная функция приложения
int APIENTRY wWinMain(_In_ HINSTANCE hInstance,
    _In_opt_ HINSTANCE hPrevInstance,
    _In_ LPWSTR    lpCmdLine,
    _In_ int       nCmdShow)
{
    UNREFERENCED_PARAMETER(hPrevInstance);
    UNREFERENCED_PARAMETER(lpCmdLine);

    // Загрузка строк из ресурсов
    LoadStringW(hInstance, IDS_APP_TITLE, szTitle,
MAX_LOADSTRING);
    LoadStringW(hInstance, IDC_LAB2, szWindowClass,
MAX_LOADSTRING);
    MyRegisterClass(hInstance);

    // Инициализация экземпляра приложения и создание главного
окна
    if (!InitInstance(hInstance, nCmdShow))
    {
        return FALSE;
    }

    HACCEL hAccelTable = LoadAccelerators(hInstance,
MAKEINTRESOURCE(IDC_LAB2));

    MSG msg;

    // Основной цикл сообщений
    while (GetMessage(&msg, nullptr, 0, 0))
    {
        if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))
        {
            TranslateMessage(&msg);
            DispatchMessage(&msg);
        }
    }

    return (int)msg.wParam;
}

// Регистрация класса окна
ATOM MyRegisterClass(HINSTANCE hInstance)
{
    WNDCLASSEXW wcex;

    wcex.cbSize = sizeof(WNDCLASSEX);

    wcex.style = CS_HREDRAW | CS_VREDRAW;
    wcex.lpfnWndProc = WndProc;
    wcex.cbClsExtra = 0;
    wcex.cbWndExtra = 0;
    wcex.hInstance = hInstance;
```

```
    wcex.hIcon = LoadIcon(hInstance, MAKEINTRESOURCE(IDI_LAB2));
    wcex.hCursor = LoadCursor(nullptr, IDC_ARROW);
    wcex.hbrBackground = (HBRUSH)(COLOR_WINDOW + 1);
    wcex.lpszMenuName = 0;
    wcex.lpszClassName = szWindowClass;
    wcex.hIconSm = LoadIcon(wcex.hInstance,
MAKEINTRESOURCE(IDI_SMALL));

    return RegisterClassExW(&wcex);
}

// Инициализация главного окна приложения
BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)
{
    hInst = hInstance; // Сохранение маркера экземпляра в
глобальной переменной

    // Создание окна с заданными параметрами
    HWND hWnd = CreateWindowW(szWindowClass, szTitle,
WS_OVERLAPPEDWINDOW,
        (GetSystemMetrics(SM_CXSCREEN) - 800) / 2, //
Центрирование по X
        (GetSystemMetrics(SM_CYSCREEN) - 800) / 2, //
Центрирование по Y
        800, 800, nullptr, nullptr, hInstance, nullptr);

    if (!hWnd)
    {
        return FALSE;
    }

    ShowWindow(hWnd, nCmdShow);
    UpdateWindow(hWnd);

    return TRUE;
}

// Обработчик сообщений главного окна
LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam,
LPARAM lParam)
{
    switch (message)
    {
    case WM_PAINT:
    {
        PAINTSTRUCT ps;
        HDC hdc = BeginPaint(hWnd, &ps);

        // Получаем размеры клиентской области окна
        RECT clientRect;
        GetClientRect(hWnd, &clientRect);
        int clientWidth = clientRect.right - clientRect.left;
        int clientHeight = clientRect.bottom - clientRect.top;
```

```
            // Вызываем функцию DrawCircle для рисования окружности,
            // учитывая размеры клиентской области
            int circleRadius = min(clientWidth, clientHeight) / 2;
            int circleX = clientWidth / 2;
            int circleY = clientHeight / 2;
            DrawCircle(hdc, circleX, circleY, circleRadius, RGB(255,
0, 0)); // Пропорциональное рисование окружности

            EndPaint(hWnd, &ps);
        }
        break;
        case WM_SIZE:
            // При изменении размера окна, запрашиваем перерисовку
окна
            InvalidateRect(hWnd, NULL, TRUE);
            break;
        case WM_DESTROY:
            PostQuitMessage(0);
            break;
        default:
            return DefWindowProc(hWnd, message, wParam, lParam);
        }
        return 0;
}

// Функция для рисования окружности с алгоритмом Брезенхема
void DrawCircle(HDC hdc, int x, int y, int r, COLORREF color)
{
    int x1, y1, yk = 0; // Инициализация переменных для
алгоритма Брезенхема
    int sigma, delta, f; // Дополнительные переменные для
расчетов

    x1 = 0; // Начальные координаты на оси X
    y1 = r; // Начальные координаты на оси Y
    delta = 2 * (1 - r); // Начальное значение дельты

    do
    {
        // Установка пикселей для всех четырех четвертей
окружности
        SetPixel(hdc, x + x1, y + y1, color);
        SetPixel(hdc, x - x1, y + y1, color);
        SetPixel(hdc, x + x1, y - y1, color);
        SetPixel(hdc, x - x1, y - y1, color);

        f = 0; // Флаг для проверки условия завершения
        if (y1 < yk)
            break;
        if (delta < 0)
        {
            sigma = 2 * (delta + y1) - 1;
```

```
            if (sigma <= 0)
            {
                x1++;
                delta += 2 * x1 + 1;
                f = 1; // Установка флага, чтобы
инкрементировать x1
            }
        }
        else if (delta > 0)
        {
            sigma = 2 * (delta - x1) - 1;
            if (sigma > 0)
            {
                y1--;
                delta += 1 - 2 * y1;
                f = 1; // Установка флага, чтобы
декрементировать y1
            }
        }
        if (!f)
        {
            x1++;
            y1--;
            delta += 2 * (x1 - y1 - 1);
        }
    } while (1); // Повторяем до завершения рисования окружности

    // Рассчитываем длину линии для диагональных линий
    int lineLength = int(r / sqrt(2.0)); // Длина линии равна
половине диаметра окружности

    // Рисуем две диагональные линии, чтобы получить круг
    DrawLine(hdc, x - lineLength, y - lineLength, x +
lineLength, y + lineLength, color);
    DrawLine(hdc, x - lineLength, y + lineLength, x +
lineLength, y - lineLength, color);
}

// Функция для рисования линии с алгоритмом Брезенхема
void DrawLine(HDC hdc, int x1, int y1, int x2, int y2, COLORREF
color)
{
    int dx = abs(x2 - x1); // Разница между x координатами
    int dy = abs(y2 - y1); // Разница между y координатами
    int sx = (x1 < x2) ? 1 : -1; // Направление движения по оси
X
    int sy = (y1 < y2) ? 1 : -1; // Направление движения по оси
Y
    int err = dx - dy; // Ошибка
    int err2;

    while (true)
    {
```

```
            SetPixel(hdc, x1, y1, color); // Устанавливаем пиксель

            if (x1 == x2 && y1 == y2)
                break; // Завершаем, если достигли конечных
координат

            err2 = 2 * err;

            if (err2 > -dy)
            {
                err -= dy;
                x1 += sx;
            }
            if (err2 < dx)
            {
                err += dx;
                y1 += sy;
            }
        }
}
```

Результаты работа представлены на рисунке 1.2.

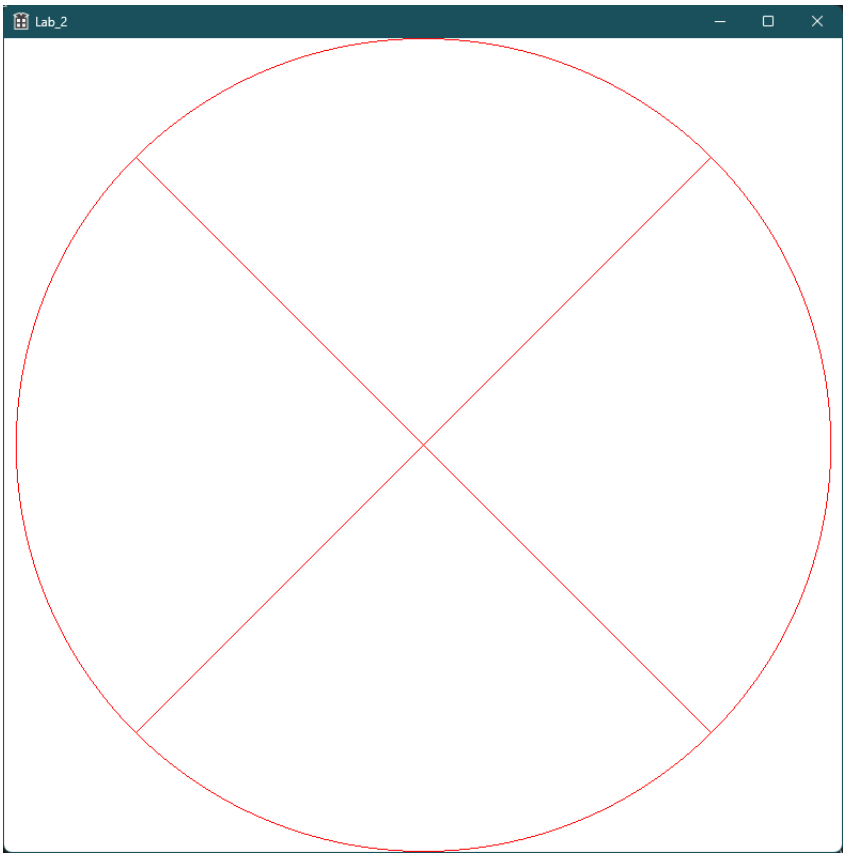

Рисунок 1.2 – Пример работы приложения

Список использованных источников

1 РД ФГБОУ ВО «КнАГТУ» 013-2016. Текстовые студенческие работы. Правила оформления. – Введ. 2016-03-10. – Комсомольск-на-Амуре: ФГБОУ ВО «КнАГТУ», 2016. – 55 с.