Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей Кафедра информатики

Дисциплина: Операционные среды и системное программирование

**ОТЧЕТ**

к лабораторной работе № 2

на тему «Расширенное использование оконного интерфейса Win 32 и GDI. Формирование сложных изображений, создание и использование элементов управления, обработка различных сообщений, механизм перехвата сообщений (winhook).»

Выполнил:

студент гр. 153504

Князев Н.Д.

Проверил:

Гриценко Н.Ю.

Минск 2023

СОДЕРЖАНИЕ

[1 Цель Работы 3](#_Toc147936442)

[2 Краткие Теоретические Сведения 4](#_Toc147936443)

[3 Реализация Программного Продукта 5](#_Toc147936444)

[Список Использованных Источников 8](#_Toc147936445)

[ПРИЛОЖЕНИЕ А(обязательное)Исходный код программы 9](#_Toc147936446)

# 1 ЦЕЛЬ РАБОТЫ

Необходимо реализовать графическое приложение для анимации движения объектов с возможностью настройки траектории и скорости. Основной целью является процесс создания и размещения элементов управления, с последующей обработкой сообщений, работа с GDI и использование механизма перехвата сообщений (winhook).

# 2 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

GDI выполняет важную функцию в представлении и передаче графических объектов на различные устройства отображения, такие как мониторы и принтеры. Этот интерфейс отвечает за разнообразные графические операции, включая отрисовку линий и кривых, вывод шрифтов, а также управление цветовой палитрой.

Путем связи приложений с графическим оборудованием, GDI обеспечивает высокоуровневый способ визуализации данных и взаимодействия пользователя с приложениями. Этот компонент позволяет создавать пользовательские интерфейсы и эффективно манипулировать графическими данными, обеспечивая при этом совместимость с различными видами оборудования.

Для создания элемента управления используется функция CreateWindow или CreateWindowEx. При создании элемента управления указывается его класс и другие параметры, такие как положение, размер, стиль и идентификатор.

Для реагирования на события, происходящие с элементами управления (например, нажатие кнопки), необходимо обработать сообщения, которые они генерируют. Это можно сделать, добавив обработчик сообщений (WndProc) и в нем обрабатывать сообщения, связанные с элементами управления.

Можно настраивать различные свойства элемента управления, такие как текст, цвет, активность и многое другое, с использованием соответствующих функций, например, SetWindowText для установки текста кнопки.

Механизм перехвата сообщений (WinHook) является важной частью Windows API и позволяет программам мониторить и перехватывать сообщения, отправляемые системой или другими приложениями.

Механизм перехвата сообщений позволяет программам создавать так называемые хуки (hooks), которые выполняются в ответ на определенные события и сообщения в системе. Эти события могут включать в себя клавишные нажатия, мышиные действия, изменения в оконной системе, а также сообщения об изменениях в ресурсах и состоянии окон.

Одним из наиболее распространенных применений механизма перехвата сообщений является создание приложений, управляющих поведением и интерактивностью других программ, таких как утилиты для записи макросов, системы мониторинга активности пользователя и средства обеспечения безопасности. Он также играет важную роль в разработке программ для автоматизации задач и создания плагинов для сторонних приложений.

# 3 РЕАЛИЗАЦИЯ ПРОГРАММНОГО ПРОДУКТА

Этот код представляет собой простой текстовый редактор на платформе Windows, созданный с использованием WinAPI и GDI+. Вот, что делают функции в этом коде:

1. **WinMain()**: Эта функция является точкой входа в приложение. Она инициализирует GDI+ и создает главное окно редактора.
2. **CreateMainWin()**: Создает главное окно приложения, включая текстовое поле и кнопки для выбора шрифта, цвета фона, открытия и сохранения файла.
3. **WndProc()**: Обработчик сообщений главного окна. Эта функция обрабатывает различные сообщения, такие как изменение размера окна, отрисовку фона и текста, а также команды от кнопок на главном окне.
4. **UpdateFont()**: Обновляет шрифт текстового поля на основе выбранного пользователем шрифта.
5. **OpenFile()**: Открывает выбранный пользователем текстовый файл и отображает его содержимое в текстовом поле.
6. **SaveFile()**: Сохраняет содержимое текстового поля в выбранный пользователем файл.
7. **DrawBackground()**: Рисует фон главного окна приложения с использованием выбранного цвета фона.
8. **DrawText()**: Рисует текст в текстовом поле с использованием выбранного шрифта и цвета текста.

Данный код также включает в себя обработку различных сообщений и команд от пользователя, таких как изменение размера окна, выбор шрифта и цвета фона, а также открытие и сохранение файлов.

**4 РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ ПРОГРАММЫ**

В результате выполнения лабораторной работы получился текстовый редактор, функционал которого позволяет изменить цвет фона окна, изменить шрифт текста, а также открыть и сохранить написанный текст. (рисунок 4.1, рисунок 4.2).

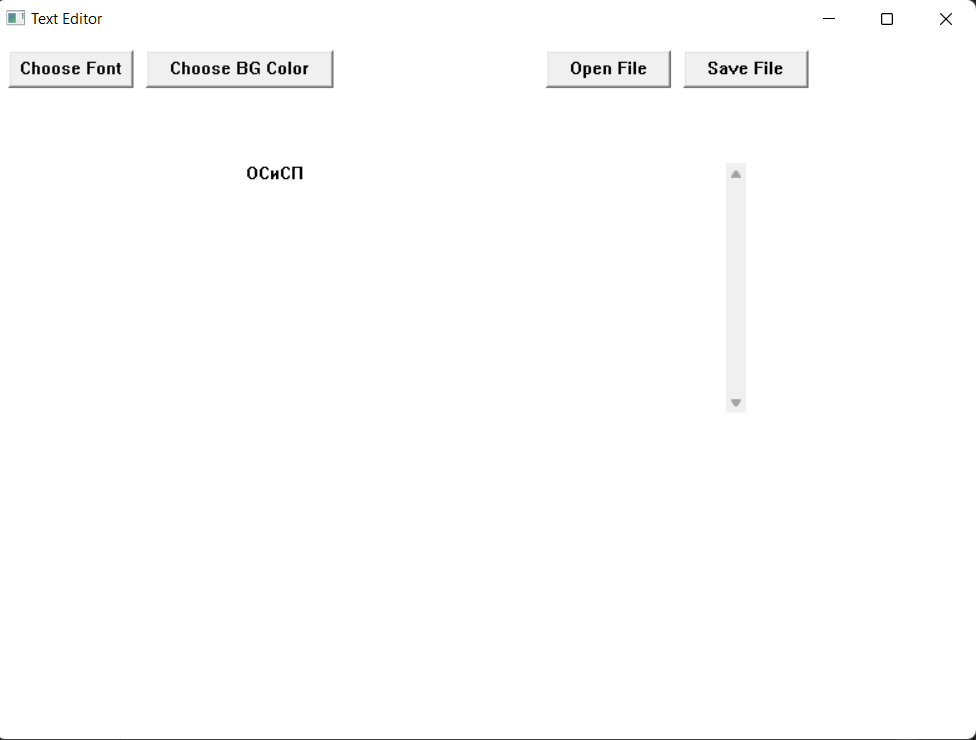


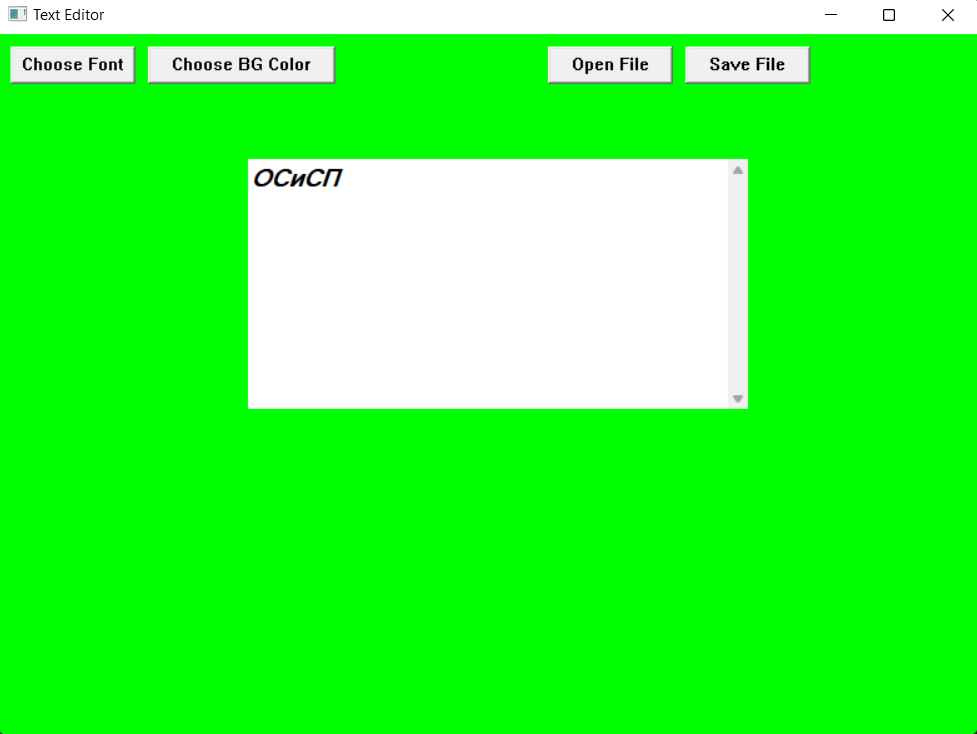
Рисунок 4.1 – Работа программы с заданными настройками

Рисунок 4.2 – Работа программы с изменёнными настройками

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Марапулец Ю. В. Системное программирование в WIN API. Учебное пособие, 2021. – 256 с

[2] Графический интерфейс GDI в Microsoft Windows [Электронный ресурс]. – Режим доступа: https://www.frolov-lib.ru/books/bsp/v14/ch3\_2.htm.

[3] Интерфейс графического устройства (GDI) [Электронный ресурс]. – Режим доступа: https://documentation.help/Win32/GDI.htm.

[4] Системная палитра [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/ru-ru/windows/win32/gdi/system-palette.

[5] Использование перехватчиков [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/ru-ru/windows/win32/winmsg/using-hooks.

# 

# ПРИЛОЖЕНИЕ А

**(обязательное)**

**Исходный код программы**

Листинг 1 – Файл TxtEditor.h:

#pragma once

#include "Resource.h"

#include <windows.h>

#include <cmath>

Листинг 2 – Файл TxtEditor.cpp:

#include <windows.h>

#include <richedit.h>

#include <commdlg.h>

#include <gdiplus.h>

#include <string>

#include <richedit.h>

#pragma comment (lib, "Gdiplus.lib")

HINSTANCE hInst;

HWND hMainWnd;

HWND hEdit;

LOGFONT lf;

COLORREF textColor = RGB(0, 0, 0);

COLORREF bgColor = RGB(255, 255, 255);

HBRUSH hBrush = NULL;

#define IDC\_CHOOSE\_FONT 1001

#define IDC\_CHOOSE\_BG\_COLOR 1003

#define IDC\_OPEN\_FILE 1004

#define IDC\_SAVE\_FILE 1005

LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam);

void CreateMainWin(HINSTANCE hInstance, int nCmdShow);

void UpdateFont();

void OpenFile();

void SaveFile();

void DrawBackground(HWND hwdn);

void DrawText();

int APIENTRY WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)

{

hInst = hInstance;

Gdiplus::GdiplusStartupInput gdiplusStartupInput;

ULONG\_PTR gdiplusToken;

Gdiplus::GdiplusStartup(&gdiplusToken, &gdiplusStartupInput, NULL);

CreateMainWin(hInstance, nCmdShow);

MSG msg;

while (GetMessage(&msg, NULL, 0, 0))

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

if (hBrush)

{

DeleteObject(hBrush);

}

Gdiplus::GdiplusShutdown(gdiplusToken);

return (int)msg.wParam;

}

void CreateMainWin(HINSTANCE hInstance, int nCmdShow)

{

LoadLibrary(TEXT("Msftedit.dll"));

WNDCLASSEX wcex;

memset(&wcex, 0, sizeof(wcex));

wcex.cbSize = sizeof(WNDCLASSEX);

wcex.style = CS\_HREDRAW | CS\_VREDRAW;

wcex.lpfnWndProc = WndProc;

wcex.hInstance = hInstance;

wcex.hCursor = LoadCursor(NULL, IDC\_ARROW);

wcex.hbrBackground = (HBRUSH)(COLOR\_WINDOW + 1);

wcex.lpszClassName = L"TextEditorClass";

RegisterClassEx(&wcex);

hMainWnd = CreateWindow(L"TextEditorClass", L"Text Editor", WS\_OVERLAPPEDWINDOW,

CW\_USEDEFAULT, CW\_USEDEFAULT, 800, 600, NULL, NULL, hInstance, NULL);

if (!hMainWnd)

{

MessageBox(NULL, L"Window creation failed", L"Error", MB\_ICONERROR);

return;

}

ShowWindow(hMainWnd, nCmdShow);

UpdateWindow(hMainWnd);

hEdit = CreateWindowExW(0, L"EDIT", L"", WS\_CHILD | WS\_VISIBLE | WS\_VSCROLL | ES\_MULTILINE | ES\_AUTOVSCROLL | ES\_WANTRETURN | WS\_EX\_LAYERED,

200, 100, 400, 200, hMainWnd, NULL, hInstance, NULL);

CreateWindow(L"BUTTON", L"Choose Font", WS\_CHILD | WS\_VISIBLE, 10, 10, 100, 30, hMainWnd, (HMENU)IDC\_CHOOSE\_FONT, hInstance, NULL);

CreateWindow(L"BUTTON", L"Choose BG Color", WS\_CHILD | WS\_VISIBLE, 120, 10, 150, 30, hMainWnd, (HMENU)IDC\_CHOOSE\_BG\_COLOR, hInstance, NULL);

CreateWindow(L"BUTTON", L"Open File", WS\_CHILD | WS\_VISIBLE, 440, 10, 100, 30, hMainWnd, (HMENU)IDC\_OPEN\_FILE, hInstance, NULL);

CreateWindow(L"BUTTON", L"Save File", WS\_CHILD | WS\_VISIBLE, 550, 10, 100, 30, hMainWnd, (HMENU)IDC\_SAVE\_FILE, hInstance, NULL);

}

LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)

{

switch (message)

{

case WM\_SIZE:

MoveWindow(hEdit, 0, 0, LOWORD(lParam), HIWORD(lParam), TRUE);

break;

case WM\_PAINT:

DrawBackground(hWnd);

DrawText();

break;

case WM\_COMMAND:

switch (LOWORD(wParam))

{

case IDC\_CHOOSE\_FONT:

{

CHOOSEFONT cf;

ZeroMemory(&cf, sizeof(cf));

cf.lStructSize = sizeof(cf);

cf.hwndOwner = hWnd;

cf.lpLogFont = &lf;

cf.Flags = CF\_SCREENFONTS | CF\_EFFECTS | CF\_INITTOLOGFONTSTRUCT;

if (ChooseFont(&cf))

{

UpdateFont();

}

}

break;

case IDC\_CHOOSE\_BG\_COLOR:

{

CHOOSECOLOR cc;

static COLORREF acrCustClr[16];

ZeroMemory(&cc, sizeof(cc));

cc.lStructSize = sizeof(cc);

cc.hwndOwner = hWnd;

cc.lpCustColors = (LPDWORD)acrCustClr;

cc.rgbResult = bgColor;

cc.Flags = CC\_FULLOPEN | CC\_RGBINIT;

if (ChooseColor(&cc))

{

bgColor = cc.rgbResult;

if (hBrush)

{

DeleteObject(hBrush);

}

hBrush = CreateSolidBrush(bgColor);

RECT rcClient;

GetClientRect(hWnd, &rcClient);

HDC hdc = GetDC(hWnd);

FillRect(hdc, &rcClient, hBrush);

ReleaseDC(hWnd, hdc);

InvalidateRect(hWnd, NULL, TRUE);

}

}

break;

case IDC\_OPEN\_FILE:

OpenFile();

break;

case IDC\_SAVE\_FILE:

SaveFile();

break;

}

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

default:

return DefWindowProc(hWnd, message, wParam, lParam);

}

return 0;

}

void UpdateFont()

{

HFONT hFont = CreateFontIndirect(&lf);

if (hFont)

{

SendMessage(hEdit, WM\_SETFONT, (WPARAM)hFont, TRUE);

SetTextColor(GetDC(hEdit), textColor);

InvalidateRect(hEdit, NULL, TRUE);

}

}

void OpenFile()

{

OPENFILENAME ofn;

wchar\_t szFileName[MAX\_PATH] = L"";

ZeroMemory(&ofn, sizeof(ofn));

ofn.lStructSize = sizeof(ofn);

ofn.hwndOwner = hMainWnd;

ofn.lpstrFilter = L"Text Files\0\*.txt\0All Files\0\*.\*\0";

ofn.lpstrFile = szFileName;

ofn.nMaxFile = MAX\_PATH;

ofn.Flags = OFN\_FILEMUSTEXIST | OFN\_PATHMUSTEXIST;

if (GetOpenFileName(&ofn))

{

HANDLE hFile = CreateFile(ofn.lpstrFile, GENERIC\_READ, 0, NULL, OPEN\_EXISTING, FILE\_ATTRIBUTE\_NORMAL, NULL);

if (hFile != INVALID\_HANDLE\_VALUE)

{

DWORD dwFileSize = GetFileSize(hFile, NULL);

if (dwFileSize != INVALID\_FILE\_SIZE)

{

char\* pBuffer = new char[dwFileSize + 1];

DWORD dwRead;

if (ReadFile(hFile, pBuffer, dwFileSize, &dwRead, NULL))

{

pBuffer[dwRead] = '\0';

SetWindowTextA(hEdit, pBuffer);

}

delete[] pBuffer;

}

CloseHandle(hFile);

}

}

}

void SaveFile()

{

OPENFILENAME ofn;

wchar\_t szFileName[MAX\_PATH] = L"";

ZeroMemory(&ofn, sizeof(ofn));

ofn.lStructSize = sizeof(ofn);

ofn.hwndOwner = hMainWnd;

ofn.lpstrFilter = L"Text Files\0\*.txt\0All Files\0\*.\*\0";

ofn.lpstrFile = szFileName;

ofn.nMaxFile = MAX\_PATH;

ofn.Flags = OFN\_OVERWRITEPROMPT | OFN\_PATHMUSTEXIST;

if (GetSaveFileName(&ofn))

{

std::wstring filePath = ofn.lpstrFile;

if (filePath.find\_last\_of(L".") == std::wstring::npos)

{

filePath += L".txt";

}

HANDLE hFile = CreateFile(filePath.c\_str(), GENERIC\_WRITE, 0, NULL, CREATE\_ALWAYS, FILE\_ATTRIBUTE\_NORMAL, NULL);

if (hFile != INVALID\_HANDLE\_VALUE)

{

int nTextLength = GetWindowTextLength(hEdit);

if (nTextLength > 0)

{

char\* pBuffer = new char[nTextLength + 1];

GetWindowTextA(hEdit, pBuffer, nTextLength + 1);

DWORD dwWritten;

WriteFile(hFile, pBuffer, nTextLength, &dwWritten, NULL);

delete[] pBuffer;

}

CloseHandle(hFile);

}

}

}

void DrawBackground(HWND hWnd)

{

PAINTSTRUCT ps;

HDC hdc = BeginPaint(hWnd, &ps);

RECT rect;

GetClientRect(hWnd, &rect);

FillRect(hdc, &rect, hBrush);

EndPaint(hWnd, &ps);

}

void DrawText()

{

PAINTSTRUCT ps;

HDC hdc = BeginPaint(hEdit, &ps);

SetTextColor(hdc, textColor);

SetBkColor(hdc, bgColor);

int nTextLength = GetWindowTextLength(hEdit);

if (nTextLength > 0)

{

char\* pBuffer = new char[nTextLength + 1];

GetWindowTextA(hEdit, pBuffer, nTextLength + 1);

TextOutA(hdc, 0, 0, pBuffer, nTextLength);

delete[] pBuffer;

}

EndPaint(hEdit, &ps);

}