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

ОТЧЁТ

к лабораторной работе №2

на тему

Расширенное использование оконного интерфейса Win 32 и GDI

Выполнил: студент группы 153503

Сорокин Артем Сергеевич

Проверил: Гриценко Никита Юрьевич

Минск 2023

СОДЕРЖАНИЕ

[1 Цель работы 3](#_Toc146728177)

[2 Краткие теоретические сведения 4](#_Toc146728178)

[2 Результаты выполнения лабораторной работы 5](#_Toc146728179)

[Выводы 6](#_Toc146728180)

[Список использованных источников 7](#_Toc146728181)

[Приложение А (обязательное) Листинг кода 8](#_Toc146728182)

## 1 ЦЕЛЬ РАБОТЫ

## Целью выполнения лабораторной работы является расширенное использование оконного интерфейса Win32 и GDI, формирование сложных изображений, создание и использование элементов управления, обработка различных сообщений, использование механизма перехвата сообщений (winhook). Чтобы продемонстрировать все условия, будет создано оконное приложение, позволяющее менять фон изображения, с помощью функций GDI.2 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Win32 API (Windows API) - это набор функций и интерфейсов прикладного программирования, предоставляемых операционной системой Microsoft Windows для разработки приложений под эту операционную систему. Win32 API предоставляет доступ к широкому спектру функциональности операционной системы, включая создание окон, работу с файлами и директориями, многозадачность, ввод и вывод данных, сетевое взаимодействие и многое другое.

Разработчики используют Win32 API на языках программирования C и C++ для создания приложений под операционной системой Windows.

GDI (Graphics Device Interface) - это набор функций и структур программного интерфейса в операционной системе Windows, который предоставляет разработчикам возможности для работы с графикой и выводом графических объектов на экран, на принтер или на другие графические устройства.

Для выполнения данной лабораторной работы, были использованы следующие теоретические сведения и концепции:

– WndProc — это процедура обработки сообщений для окна приложения. Она обрабатывает сообщения, поступающие от операционной системы.

– CreateWindow: функция, используемая для создания окна или элемента управления в окне. Принимает параметры, такие как класс окна, заголовок, стиль окна и позицию.

– Обработка сообщений: В главном цикле сообщений (WinMain) приложение ожидает и обрабатывает сообщения с помощью GetMessage, TranslateMessage и DispatchMessage. WndProc обрабатывает различные типы сообщений, такие как рисование, клавиатурные события и таймеры.

– Рисование: Рисование выполняется в процедуре WM\_PAINT. Окно представляет из себя поле для ввода текста.

– Хук клавиатуры: Приложение также демонстрирует использование глобального хука клавиатуры с помощью функций SetWindowsHookEx и UnhookWindowsHookEx.

– Обработка ошибок: В коде предусмотрена обработка ошибок с использованием сообщений MessageBox, чтобы уведомлять пользователя о проблемах.

## 3 РЕЗУЛЬТАТЫ ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

В ходе выполнения лабораторной работы было создано оконное приложение, позволяющее задавать цвет фона через вводимые параметры (рисунок 1).

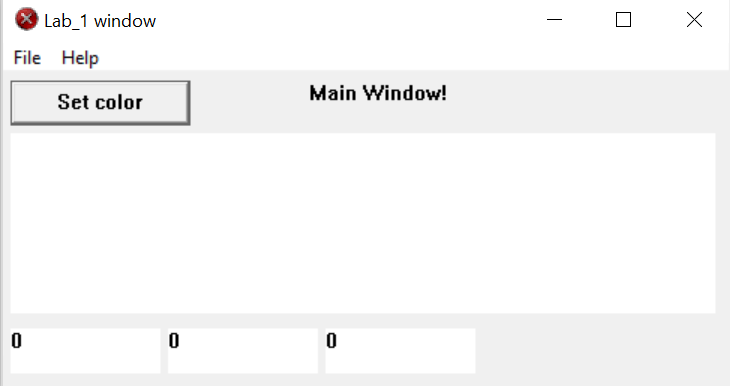


Рисунок 1 - Главный экран

Пользователь может ввести параметры и отобразить полученный цвет в приложении (рисунок 2).

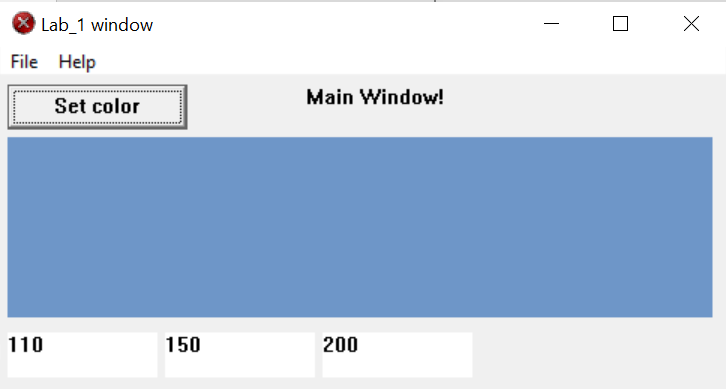


Рисунок 2 ─ Изменение цвета

## ВЫВОДЫ

Таким образом, в ходе данной лабораторной работы было проведено расширенное использование оконного интерфейса Win32 и GDI для формирования сложных изображений, создания и использования элементов управления, а также обработки различных сообщений. Был реализован механизм перехвата сообщений с помощью WinHook. В результате работы был разработан текстовый редактор, обладающий возможностью изменения цвета фона.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Щупак Ю. Win32 API. Разработка приложений для Windows. ─ СПб: Питер, 2008. ─ 592 с.: ил.

[2] Создание классических приложений для Windows с использованием API Win32 [Электронный ресурс]. ─ Режим доступа: https://learn.microsoft.com/ru-ru/windows/win32/api ─ Дата доступа 19.09.2023

## ПРИЛОЖЕНИЕ А

## (обязательное)

## Листинг кода

Листинг 1 – Файл main.cpp

#ifndef UNICODE

#define UNICODE

#endif

#include <windows.h>

#include "SoftwareDefinitions.h"

#include <string>

int WINAPI WinMain(HINSTANCE hInst, HINSTANCE hPrevInst, LPSTR args, int ncmdshow) {

HINSTANCE hInstance = GetModuleHandle(NULL);

g\_hHook = SetWindowsHookEx(WH\_KEYBOARD\_LL, KeyboardHookCallback, hInstance, 0);

WNDCLASS SoftwareMainClass = NewWindowClass((HBRUSH)COLOR\_WINDOW, LoadCursor(NULL, IDC\_HAND), hInst, LoadIcon(NULL, IDI\_ERROR),

L"MainWndClass", SoftwareMainProcedure);

if (!RegisterClassW(&SoftwareMainClass)) { return -1; }

CreateWindow(L"MainWndClass", L"Lab\_1 window", WS\_OVERLAPPEDWINDOW | WS\_VISIBLE, 100, 100, 500, 250, NULL, NULL, NULL, NULL);

if (g\_hHook == NULL)

{

return 1;

}

MSG msg = { };

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

TranslateMessage(&msg);

DispatchMessage(&msg);

}

UnhookWindowsHookEx(g\_hHook);

return 0;

}

WNDCLASS NewWindowClass(HBRUSH BGColor, HCURSOR Cursor, HINSTANCE hInst, HICON Icon, LPCWSTR Name, WNDPROC Procedure) {

WNDCLASS NWC = { 0 };

NWC.hIcon = Icon;

NWC.hCursor = Cursor;

NWC.hInstance = hInst;

NWC.lpszClassName = Name;

NWC.hbrBackground = BGColor;

NWC.lpfnWndProc = Procedure;

return NWC;

}

LRESULT CALLBACK SoftwareMainProcedure(HWND hWnd, UINT msg, WPARAM wp, LPARAM lp) {

switch (msg)

{

case WM\_COMMAND:

switch (wp) {

case OnMenuClicked1:

MessageBoxA(hWnd, "menu 1 was klicked", "Menu worked", MB\_OK);

break;

case OnClear:

SetWindowTextA(hEditControl, "");

break;

case OnReadColor:

brushRectangle = CreateSolidBrush(

RGB(GetDlgItemInt(hWnd, IndexColorR, NULL, false),

GetDlgItemInt(hWnd, IndexColorG, NULL, false),

GetDlgItemInt(hWnd, IndexColorB, NULL, false)

));

RedrawWindow(hWnd, NULL, NULL, RDW\_UPDATENOW | RDW\_INVALIDATE);

break;

case OnExitSoftware:

PostQuitMessage(0);

break;

}

break;

case WM\_PAINT:

BeginPaint(hWnd, &ps);

FillRect(ps.hdc, &windowRectangle, brushRectangle);

EndPaint(hWnd, &ps);

break;

case WM\_CREATE:

MainWndAddMenus(hWnd);

MainWndAddWidgets(hWnd);

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

default:

return DefWindowProc(hWnd, msg, wp, lp);

}

}

void MainWndAddMenus(HWND hwnd) {

HMENU RootMenu = CreateMenu();

HMENU SubMenu = CreateMenu();

//HMENU SubActionMenu = CreateMenu();

//AppendMenu(SubActionMenu, MF\_STRING, OnMenuClicked3, L"Menu 3");

//AppendMenu(SubMenu, MF\_POPUP, (UINT\_PTR)SubActionMenu, L"Action");

AppendMenu(SubMenu, MF\_STRING, OnClear, L"Clear");

AppendMenu(SubMenu, MF\_SEPARATOR, NULL, NULL);

AppendMenu(SubMenu, MF\_STRING, OnExitSoftware, L"Exit");

AppendMenu(RootMenu, MF\_POPUP, (UINT\_PTR)SubMenu, L"File");

AppendMenu(RootMenu, MF\_STRING, (UINT\_PTR)SubMenu, L"Help");

SetMenu(hwnd, RootMenu);

}

void MainWndAddWidgets(HWND hWnd) {

hStaticControl = CreateWindowA("static", "Main Window!", WS\_VISIBLE | WS\_CHILD | ES\_CENTER, 200, 5, 100, 30, hWnd, NULL, NULL, NULL);

//hEditControl = CreateWindowA("edit", "Enter text!", WS\_VISIBLE | WS\_CHILD | ES\_MULTILINE | WS\_VSCROLL, 5, 40, 470, 120, hWnd, NULL, NULL, NULL);

windowRectangle = { 5 + 470, 40, 5, 40 + 120 };

CreateWindowA("edit", "0", WS\_VISIBLE | WS\_CHILD | ES\_MULTILINE | ES\_NUMBER, 5, 170, 100, 30, hWnd, (HMENU)IndexColorR, NULL, NULL);

CreateWindowA("edit", "0", WS\_VISIBLE | WS\_CHILD | ES\_MULTILINE | ES\_NUMBER, 110, 170, 100, 30, hWnd, (HMENU)IndexColorG, NULL, NULL);

CreateWindowA("edit", "0", WS\_VISIBLE | WS\_CHILD | ES\_MULTILINE | ES\_NUMBER, 215, 170, 100, 30, hWnd, (HMENU)IndexColorB, NULL, NULL);

//CreateWindowA("button", "Clear", WS\_VISIBLE | WS\_CHILD | ES\_CENTER, 130, 5, 50, 30, hWnd, (HMENU)OnClear , NULL, NULL);

CreateWindowA("button", "Set color", WS\_VISIBLE | WS\_CHILD | ES\_CENTER, 5, 5, 120, 30, hWnd, (HMENU)OnReadColor, NULL, NULL);

}

LRESULT CALLBACK KeyboardHookCallback(int nCode, WPARAM wParam, LPARAM lParam)

{

if (nCode == HC\_ACTION && wParam == WM\_KEYDOWN)

{

KBDLLHOOKSTRUCT\* pKeyboardData = (KBDLLHOOKSTRUCT\*)lParam;

DWORD keyCode = pKeyboardData->vkCode;

// Пример: Вывод кода нажатой клавиши в консоль

if (keyCode == VK\_TAB)

{

// Обработка нажатия клавиши Tab

MessageBox(NULL, L"Tab key pressed!", L"Key Pressed", MB\_OK);

}

}

// Передача управления следующему хуку в цепочке

return CallNextHookEx(g\_hHook, nCode, wParam, lParam);

}

Листинг 2 ­– Файл SoftwareDefinitions.h

#pragma once

#define OnMenuClicked1 1

#define OnMenuClicked2 2

#define OnMenuClicked3 3

#define OnExitSoftware 4

#define OnClear 5

#define OnReadColor 6

#define TextBufferSize 256

#define IndexColorR 200

#define IndexColorG 201

#define IndexColorB 202

char Buffer[TextBufferSize];

PAINTSTRUCT ps;

RECT windowRectangle;

HBRUSH brushRectangle;

HWND hEditControl;

HWND hStaticControl;

HHOOK g\_hHook = NULL;

LRESULT CALLBACK KeyboardHookCallback(int nCode, WPARAM wParam, LPARAM lParam);

LRESULT CALLBACK SoftwareMainProcedure(HWND hWnd, UINT msg, WPARAM wp, LPARAM lp);

WNDCLASS NewWindowClass(HBRUSH BGColor, HCURSOR Cursor, HINSTANCE hInst, HICON Icon, LPCWSTR Name, WNDPROC Procedure);

void MainWndAddMenus(HWND hwnd);

void MainWndAddWidgets(HWND hwnd);