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»

Выполнил:

студент гр. 153504

Прескурел Я.Ю.

Проверил:

Гриценко Н.Ю.

Минск 2023

СОДЕРЖАНИЕ

[1 Цели работы 3](#_Toc146631498)

[2 Краткие теоретические сведения 4](#_Toc146631499)

[3 Полученные результаты 5](#_Toc146631500)

[Выводы 6](#_Toc146631501)

[Список использованных источников 7](#_Toc146631502)

[Приложение А 8](#_Toc146631503)

## 1 ЦЕЛИ РАБОТЫ

Изучить основные элементы интерфейса Win 32 и GDI. Изучить механизмы обработки различных сообщений в оконном интерфейсе, включая обработку сообщений мыши и клавиатуры. Изучение механизма перехвата сообщений (winhook). Реализовать графическое приложение для анимации движения объектов с возможностью настройки траектории и скорости.

## 2 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Оконный интерфейс Win32 и графический драйвер интерфейса (GDI) являются основой для разработки приложений под операционные системы Windows. Win32 API – это набор функций и структур, которые обеспечивают доступ к ресурсам операционной системы и позволяют программе работать с графическим пользовательским интерфейсом.

GDI основан на модели устройства с точки зрения рисования, где каждый элемент, например, кнопка или окно, представлен набором объектов GDI. Эти объекты могут быть созданы и изменены с помощью соответствующих функций API. Взаимодействие с GDI может быть осуществлено как в библиотеке DLL, так и непосредственно из окна.

В Win32 API управление элементами интерфейса осуществляется с помощью сообщений. Сообщения - это события, которые происходят в приложении, например, клик по кнопке или перемещение мыши. Они могут быть обработаны с помощью функции оконной процедуры. Для создания элементов управления, таких как кнопки, текстовые поля или ползунки, используются структуры, определенные в библиотеках Win32 API и GDI.

Чтобы обеспечить более гибкое и мощное управление сообщениями, можно использовать механизм перехвата сообщений, такой как WinHook. Этот механизм позволяет отслеживать и перехватывать сообщения, отправляемые любому приложению, и обрабатывать их внутри программы. Это может быть полезно, например, для реализации горячих клавиш или фильтрации входящих сообщений.

В целом, Win32 API и GDI предоставляют разработчикам большое количество инструментов для создания не только простых, но и сложных приложений с графическим интерфейсом. Они позволяют создавать элементы управления, обрабатывать сообщения и создавать сложные изображения, а также использовать механизм перехвата сообщений для гибкого управления взаимодействием пользователей с приложением.

## 3 РЕЗУЛЬТАТЫ ВЫПОЛНЕНИЯ ПРОГРАММЫ

В ходе выполнения лабораторной работы было реализовано графическое приложение для анимации движения объектов с возможностью настройки траектории и скорости. Результат работы программы показан на рисунках 3.1 и 3.2

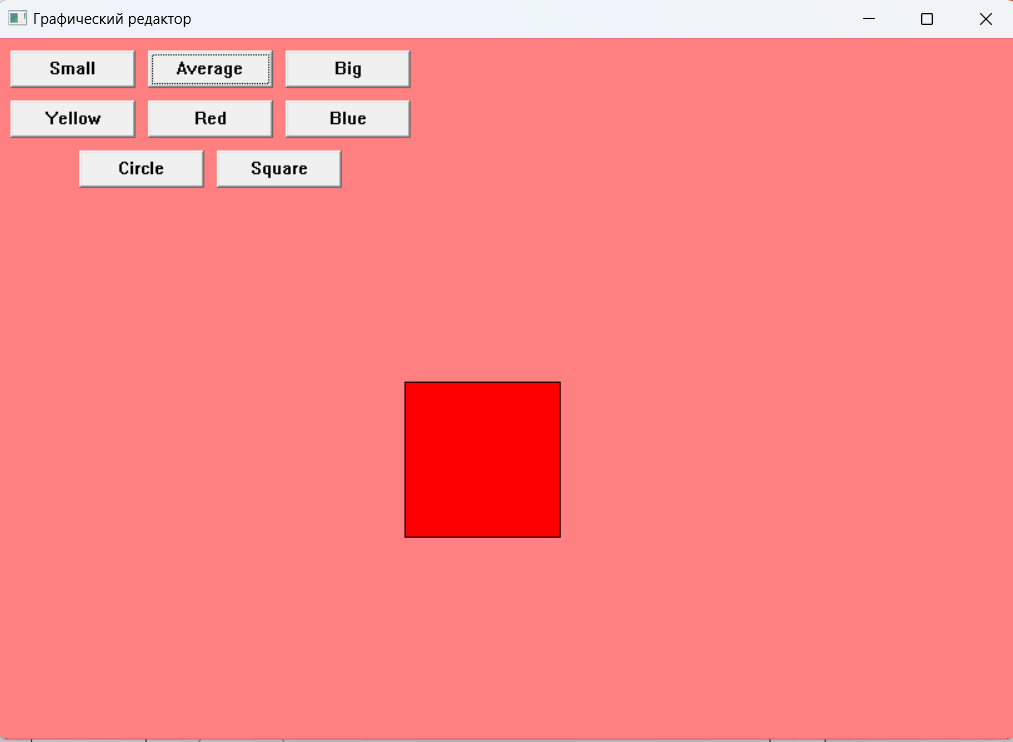


Рисунок 3.1 – Результат работы программы(1)

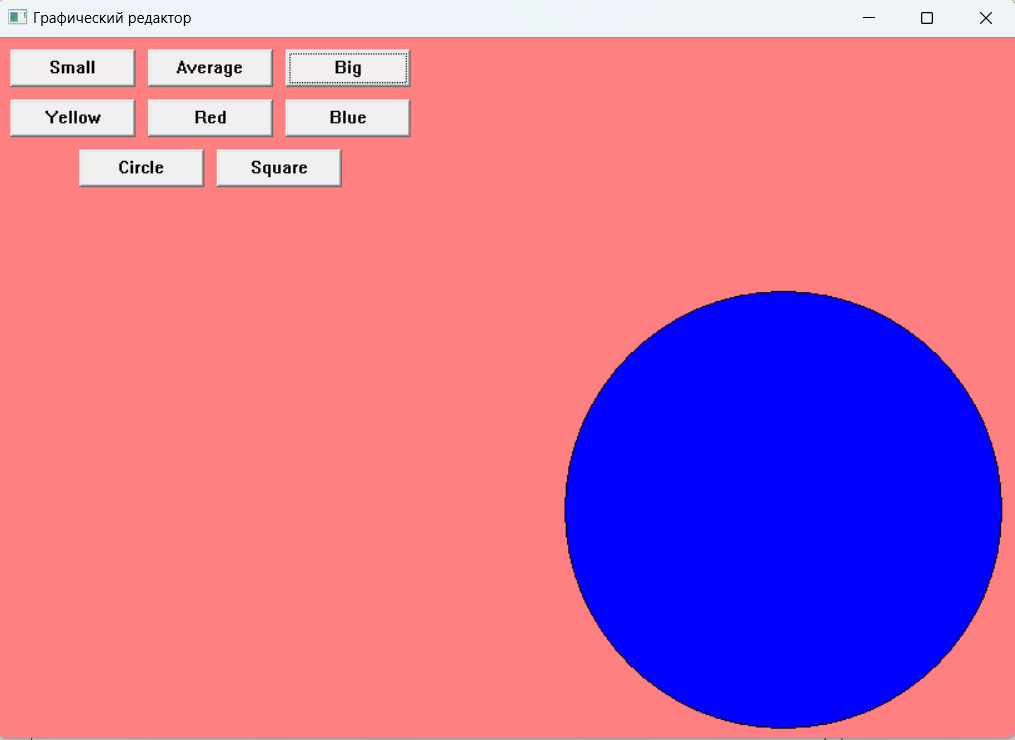


Рисунок 3.2 – Результат работы программы(2)

## ВЫВОДЫ

В ходе выполнения данной лабораторной работы были изучены особенности расширенного использования оконного интерфейса Win32 и GDI. Были изучены принципы создания и использования элементов управления. Были также изучены механизмы обработки различных сообщений и механизм перехвата сообщений с использованием Winhook.

## 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.

**ПРИЛОЖЕНИЕ А**

## (обязательное)

## Листинг кода

**Lab2.cpp**

#include <windows.h>

HINSTANCE hInst;

HWND hwndMainWnd;

HHOOK g\_keyboardHook = NULL;

bool isAnimating = false;

double currentX = 325;

double currentY = 275;

int objectSize = 100;

int objectWidth = objectSize;

int selectedShape = 0;

COLORREF selectedColor = RGB(100, 100, 100);

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

{

if (nCode == HC\_ACTION)

{

if (wParam == WM\_KEYDOWN)

{

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

if (kbStruct->vkCode == 0x53)

{

if (isAnimating)

{

isAnimating = false;

KillTimer(hwndMainWnd, 1);

}

else

{

isAnimating = true;

SetTimer(hwndMainWnd, 1, 50, NULL);

}

}

}

}

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

}

LRESULT CALLBACK WndProc(HWND hwnd, UINT msg, WPARAM wParam, LPARAM lParam)

{

switch (msg)

{

case WM\_CREATE:

hwndMainWnd = hwnd;

g\_keyboardHook = SetWindowsHookEx(WH\_KEYBOARD\_LL, KeyboardHookProc, GetModuleHandle(NULL), 0);

CreateWindow(L"BUTTON", L"Circle", WS\_CHILD | WS\_VISIBLE, 65, 90, 100, 30, hwnd, (HMENU)1, hInst, NULL);

CreateWindow(L"BUTTON", L"Square", WS\_CHILD | WS\_VISIBLE, 175, 90, 100, 30, hwnd, (HMENU)2, hInst, NULL);

CreateWindow(L"BUTTON", L"Yellow", WS\_CHILD | WS\_VISIBLE, 10, 50, 100, 30, hwnd, (HMENU)3, hInst, NULL);

CreateWindow(L"BUTTON", L"Red", WS\_CHILD | WS\_VISIBLE, 120, 50, 100, 30, hwnd, (HMENU)4, hInst, NULL);

CreateWindow(L"BUTTON", L"Blue", WS\_CHILD | WS\_VISIBLE, 230, 50, 100, 30, hwnd, (HMENU)5, hInst, NULL);

CreateWindow(L"BUTTON", L"Small", WS\_CHILD | WS\_VISIBLE, 10, 10, 100, 30, hwnd, (HMENU)6, hInst, NULL);

CreateWindow(L"BUTTON", L"Average", WS\_CHILD | WS\_VISIBLE, 120, 10, 100, 30, hwnd, (HMENU)7, hInst, NULL);

CreateWindow(L"BUTTON", L"Big", WS\_CHILD | WS\_VISIBLE, 230, 10, 100, 30, hwnd, (HMENU)8, hInst, NULL);

SendMessage(GetDlgItem(hwnd, 1), BM\_SETCHECK, BST\_CHECKED, 0);

SendMessage(GetDlgItem(hwnd, 3), BM\_SETCHECK, BST\_CHECKED, 0);

SendMessage(GetDlgItem(hwnd, 6), BM\_SETCHECK, BST\_CHECKED, 0);

break;

case WM\_PAINT:

{

PAINTSTRUCT ps;

HDC hdc = BeginPaint(hwnd, &ps);

RECT rect;

GetClientRect(hwnd, &rect);

HBRUSH hBackgroundBrush = CreateSolidBrush(RGB(255, 128, 128));

FillRect(hdc, &rect, hBackgroundBrush);

DeleteObject(hBackgroundBrush);

HBRUSH hBrush = CreateSolidBrush(selectedColor);

SelectObject(hdc, hBrush);

if (selectedShape == 0)

{

Ellipse(hdc, (int)currentX, (int)currentY, (int)(currentX + objectWidth), (int)(currentY + objectWidth));

}

else

{

Rectangle(hdc, (int)currentX, (int)currentY, (int)(currentX + objectWidth), (int)(currentY + objectWidth));

}

DeleteObject(hBrush);

EndPaint(hwnd, &ps);

}

break;

case WM\_TIMER:

if (isAnimating)

{

POINT cursorPos;

GetCursorPos(&cursorPos);

ScreenToClient(hwnd, &cursorPos);

currentX = cursorPos.x - objectWidth / 2;

currentY = cursorPos.y - objectWidth / 2;

RECT clientRect;

GetClientRect(hwnd, &clientRect);

int clientWidth = clientRect.right - clientRect.left;

int clientHeight = clientRect.bottom - clientRect.top;

if (currentX < 0)

{

currentX = 0;

}

if (currentY < 0)

{

currentY = 0;

}

if (currentX + objectWidth > clientWidth)

{

currentX = clientWidth - objectWidth;

}

if (currentY + objectWidth > clientHeight)

{

currentY = clientHeight - objectWidth;

}

InvalidateRect(hwnd, NULL, TRUE);

}

break;

case WM\_COMMAND:

switch (LOWORD(wParam))

{

case 1:

selectedShape = 0;

InvalidateRect(hwnd, NULL, TRUE);

break;

case 2:

selectedShape = 1;

InvalidateRect(hwnd, NULL, TRUE);

break;

case 3:

selectedColor = RGB(255, 255, 0);

InvalidateRect(hwnd, NULL, TRUE);

break;

case 4:

selectedColor = RGB(255, 0, 0);

InvalidateRect(hwnd, NULL, TRUE);

break;

case 5:

selectedColor = RGB(0, 0, 255);

InvalidateRect(hwnd, NULL, TRUE);

break;

case 6:

objectSize = 50;

objectWidth = objectSize;

InvalidateRect(hwnd, NULL, TRUE);

break;

case 7:

objectSize = 125;

objectWidth = objectSize;

InvalidateRect(hwnd, NULL, TRUE);

break;

case 8:

objectSize = 350;

objectWidth = objectSize;

InvalidateRect(hwnd, NULL, TRUE);

break;

}

break;

case WM\_DESTROY:

PostQuitMessage(0);

if (g\_keyboardHook != NULL)

{

UnhookWindowsHookEx(g\_keyboardHook);

g\_keyboardHook = NULL;

}

break;

default:

return DefWindowProc(hwnd, msg, wParam, lParam);

}

return 0;

}

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

{

hInst = hInstance;

WNDCLASSEX wc = { sizeof(WNDCLASSEX), CS\_HREDRAW | CS\_VREDRAW, WndProc, 0, 0, GetModuleHandle(NULL), NULL, NULL, NULL, NULL, L"MyClass", NULL };

RegisterClassEx(&wc);

HWND hwnd = CreateWindow(L"MyClass", L"Графический редактор", WS\_OVERLAPPEDWINDOW, CW\_USEDEFAULT, CW\_USEDEFAULT, 800, 600, NULL, NULL, hInstance, NULL);

ShowWindow(hwnd, nCmdShow);

UpdateWindow(hwnd);

MSG msg;

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

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

return msg.wParam;

}