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

Гайкевич Е.В.

Проверил:

Гриценко Н.Ю.

Минск 2023

СОДЕРЖАНИЕ

[Цель работы 3](#_Toc146680819)

[Краткие теоретические сведения 4](#_Toc146680820)

[Реализация программного продукта 5](#_Toc146680821)

[Результат выполнения программы 6](#_Toc146680822)

[Список использованных источников 7](#_Toc146680823)

[Приложение А 8](#_Toc146680824)

# 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

Для выбора скорости использовано текстовое поле, а для выбора траектории и объекта два выпадающих списка. Данные элементы интерфейса создаются при обработке сообщения WM\_CREATE.

Для начала и окончания анимации созданы две кнопки «Старт» и «Стоп». После начала анимации командой SetTimer запускается таймер, который каждые равные промежутки времени посылает сообщение WM\_TIMER.

В обработчике сообщения WM\_TIMER постоянно вычисляются новые координаты для объекта (было реализовано движение по прямой и по кривой, а именно функция синуса), после чего вызывается функция InvalidateRect, перерисовывающая окно.

Обработчик WM\_PAINT содержит в себе код, необходимый для подготовки к перерисовке (а именно заливка в пределах границ цвета фона, используя функции GetClientRect и FillRect, а также CreateSolidBrush для создания кисти определённого цвета) и самого отображения объекта (используя функции GDI, такие как Ellipse и Rectangle).

Для использования механизма перехвата сообщений была реализована функция KeyboardHookProc, которая является хуком, перехватывающим нажатие пробела вне окна процесса данного приложения. Данный хук подключается в обработчике сообщения WM\_CREATE функцией SetWindowsHookEx. Нажатием пробела была реализована возможность включения и выключения анимации объекта. В конце работы приложения в обработчике сообщения WM\_DESTROY данный хук удаляется командой UnhookWindowsHookEx.

# 4 РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ ПРОГРАММЫ

В результате выполнения лабораторной работы получилось приложение, способное управлять анимацией объекта. Разработанный интерфейс позволяет управлять траекторией и скоростью движения объекта. Также имеется возможность менять объект (рисунок 4.1, рисунок 4.2).

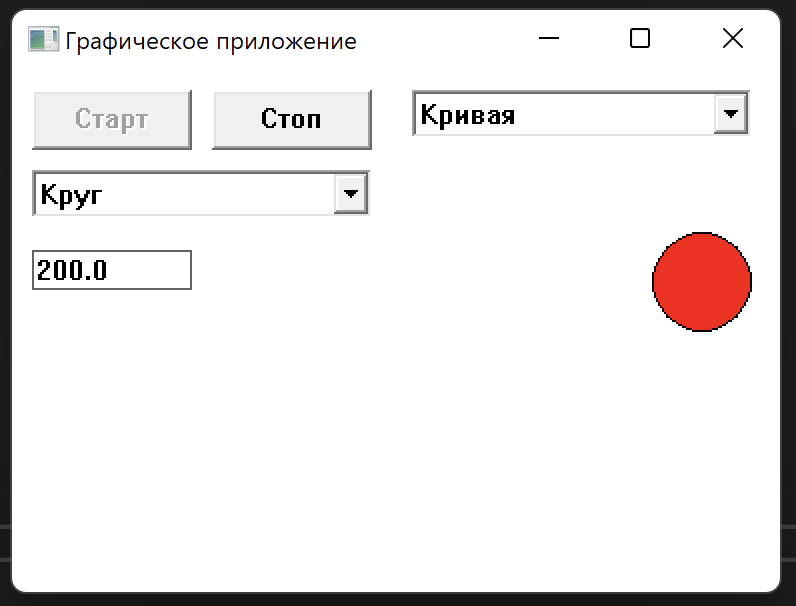


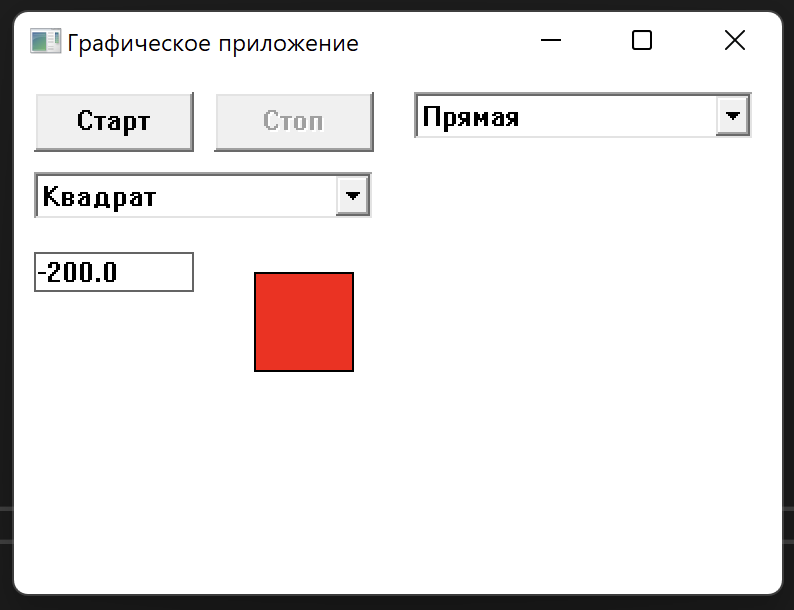
Рисунок 4.1 – Работа программы с заданными настройками

Рисунок 4.2 – Работа программы с изменёнными настройками

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Windows GDI [Электронный ресурс]. – Режим доступа: <https://learn.microsoft.com/ru-ru/windows/win32/api/_gdi/>

[2] Окна на чистом WinAPI. Или просто о сложном [Электронный ресурс]. – Режим доступа: <https://habr.com/ru/articles/352096/>.

[3] WinHook – легальный шпионаж [Электронный ресурс]. – Режим доступа: <https://codeby.net/threads/winhook-legalnyj-shpionazh.76505/>.

# ПРИЛОЖЕНИЕ А

**(обязательное)**

**Исходный код программы**

Листинг 1 – Файл Lab2.h:

#pragma once

#include "Resource.h"

#include <windows.h>

#include <cmath>

Листинг 2 – Файл Lab2.cpp:

#include "Lab2.h"

HINSTANCE hInst;

HWND hwndStartButton;

HWND hwndStopButton;

HWND hwndShapeComboBox;

HWND hwndSpeedEdit;

HWND hwndTrajectoryComboBox;

HWND hwndMainWnd;

HHOOK g\_keyboardHook = NULL;

bool isAnimating = false;

double currentX = 100;

double currentY = 100;

int objectWidth = 50;

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

if (nCode == HC\_ACTION) {

if (wParam == WM\_KEYDOWN) {

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

if (kbStruct->vkCode == VK\_SPACE) {

if (isAnimating)

{

isAnimating = false;

KillTimer(hwndMainWnd, 1);

EnableWindow(hwndStartButton, TRUE);

EnableWindow(hwndStopButton, FALSE);

}

else

{

isAnimating = true;

SetTimer(hwndMainWnd, 1, 50, NULL);

EnableWindow(hwndStartButton, FALSE);

EnableWindow(hwndStopButton, TRUE);

}

}

}

}

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;

hwndStartButton = CreateWindow(

L"BUTTON", L"Старт", WS\_CHILD | WS\_VISIBLE,

10, 10, 80, 30, hwnd, (HMENU)1, hInst, NULL

);

hwndStopButton = CreateWindow(

L"BUTTON", L"Стоп", WS\_CHILD | WS\_VISIBLE | WS\_DISABLED,

100, 10, 80, 30, hwnd, (HMENU)2, hInst, NULL

);

hwndShapeComboBox = CreateWindow(

L"COMBOBOX", L"", WS\_CHILD | WS\_VISIBLE | CBS\_DROPDOWNLIST,

10, 50, 170, 100, hwnd, (HMENU)3, hInst, NULL

);

SendMessage(hwndShapeComboBox, CB\_ADDSTRING, 0, (LPARAM)L"Круг");

SendMessage(hwndShapeComboBox, CB\_ADDSTRING, 0, (LPARAM)L"Квадрат");

SendMessage(hwndShapeComboBox, CB\_SETCURSEL, 0, 0);

hwndSpeedEdit = CreateWindow(

L"EDIT", L"200.0", WS\_CHILD | WS\_VISIBLE | WS\_BORDER,

10, 90, 80, 20, hwnd, (HMENU)4, hInst, NULL

);

hwndTrajectoryComboBox = CreateWindow(

L"COMBOBOX", L"", WS\_CHILD | WS\_VISIBLE | CBS\_DROPDOWNLIST,

200, 10, 170, 100, hwnd, (HMENU)6, hInst, NULL

);

SendMessage(hwndTrajectoryComboBox, CB\_ADDSTRING, 0, (LPARAM)L"Прямая");

SendMessage(hwndTrajectoryComboBox, CB\_ADDSTRING, 0, (LPARAM)L"Кривая");

SendMessage(hwndTrajectoryComboBox, CB\_SETCURSEL, 0, 0);

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

break;

case WM\_PAINT:

{

PAINTSTRUCT ps;

HDC hdc = BeginPaint(hwnd, &ps);

RECT rect;

GetClientRect(hwnd, &rect);

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

FillRect(hdc, &rect, hBackgroundBrush);

DeleteObject(hBackgroundBrush);

int selectedIndex = SendMessage(hwndShapeComboBox, CB\_GETCURSEL, 0, 0);

COLORREF color = RGB(255, 0, 0);

if (selectedIndex == 0)

{

HBRUSH hBrush = CreateSolidBrush(color);

SelectObject(hdc, hBrush);

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

DeleteObject(hBrush);

}

else

{

HBRUSH hBrush = CreateSolidBrush(color);

SelectObject(hdc, hBrush);

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

DeleteObject(hBrush);

}

EndPaint(hwnd, &ps);

}

break;

case WM\_COMMAND:

switch (LOWORD(wParam)) {

case 1:

if (!isAnimating) {

isAnimating = true;

SetTimer(hwnd, 1, 50, NULL);

EnableWindow(hwndStartButton, FALSE);

EnableWindow(hwndStopButton, TRUE);

}

break;

case 2:

isAnimating = false;

KillTimer(hwnd, 1);

EnableWindow(hwndStartButton, TRUE);

EnableWindow(hwndStopButton, FALSE);

break;

case 3:

InvalidateRect(hwnd, NULL, TRUE);

break;

}

break;

case WM\_DESTROY:

PostQuitMessage(0);

if (g\_keyboardHook != NULL) {

UnhookWindowsHookEx(g\_keyboardHook);

g\_keyboardHook = NULL;

}

break;

case WM\_TIMER:

if (isAnimating) {

int selectedTrajectory = SendMessage(hwndTrajectoryComboBox, CB\_GETCURSEL, 0, 0);

wchar\_t speedBuffer[32];

GetWindowText(hwndSpeedEdit, speedBuffer, sizeof(speedBuffer));

double speed = \_wtof(speedBuffer);

RECT clientRect;

GetClientRect(hwnd, &clientRect);

int clientWidth = clientRect.right - clientRect.left;

int clientHeight = clientRect.bottom - clientRect.top;

switch (selectedTrajectory) {

case 0:

currentX += speed \* 0.01;

if (currentX >= clientWidth) {

currentX = -objectWidth;

}

break;

case 1:

currentX += speed \* 0.01;

currentY += (clientHeight / 100) \* sin(currentX \* 0.05);

if (currentY < 0)

{

currentY = clientHeight;

}

if (currentY > clientHeight)

{

currentY = 0;

}

if (currentX >= clientWidth)

{

currentX = -objectWidth;

}

break;

}

InvalidateRect(hwnd, NULL, TRUE);

}

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"Графическое приложение", WS\_OVERLAPPEDWINDOW, CW\_USEDEFAULT, CW\_USEDEFAULT, 400, 300, NULL, NULL, hInstance, NULL);

ShowWindow(hwnd, nCmdShow);

UpdateWindow(hwnd);

MSG msg;

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

TranslateMessage(&msg);

DispatchMessage(&msg);

}

return msg.wParam;

}