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**

Студент Д. С. Кончик

Преподаватель Н. Ю. Гриценко

Минск 2023

СОДЕРЖАНИЕ

[1 Цель работы 3](#_Toc146752068)

[2 Теоретические сведения 4](#_Toc146752069)

[3 Результат выполнения 5](#_Toc146752070)

[Заключение 6](#_Toc146752071)

[Список использованных источников 7](#_Toc146752072)

[Приложение А (обязательное) Листинг кода 8](#_Toc146752073)

1. **ЦЕЛЬ РАБОТЫ**

Цель работы ­ изучить и применить на практике знания о расширенном использовании Win32 и GDI для создания приложений на Windows, научиться формировать сложные изображения, обрабатывать различные сообщения, а также изучить механизм перехвата сообщений с использованием winhook. Для достижения цели будет создано графическое приложение стрелочных часов с возможностью изменения часового пояса.

1. **ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

Интерфейс графических устройств GDI операционной системы Microsoft Windows, как это можно предположить из названия, предназначен для взаимодействия приложений Windows с графическими устройствами, такими как видеомонитор, принтер или плоттер.

С точки зрения приложения GDI – это контекст отображения и инструменты для рисования. Контекст отображения можно сравнить с листом бумаги, на котором приложение рисует то или иное графическое изображение, а также пишет текст. Инструменты для рисования – это перья, кисти (а также шрифты и даже целые графические изображения), с помощью которых создается изображение. Кроме контекста отображения и инструментов для рисования, приложениям доступны десятки функций программного интерфейса GDI, предназначенные для работы с контекстом отображения и инструментами.

Если говорить более точно, контекст отображения является структурой данных, описывающей устройство отображения. В этой структуре хранятся различные характеристики устройства и набор инструментов для рисования, выбранный по умолчанию. Приложение может выбирать в контекст отображения различные инструменты (например, перья различной толщины и цвета). Поэтому если нужно нарисовать линию красного или зеленого цвета, перед выполнением операции следует выбрать в контекст отображения соответствующее перо.

Функции рисования не имеют параметров, указывающих цвет или толщину линии. Такие параметры хранятся в контексте отображения.

Приложение может создать контекст отображения не только для экрана монитора или окна, но и для любого другого графического устройства вывода, например, для принтера.

Операционная система Windows предоставляет программисту функции, позволяющие установить в приложении требуемое количество программных таймеров. С помощью таймеров приложение, например, может обеспечить временную синхронизацию и задание временных интервалов. Функция SetTimer() создает или видоизменяет системный таймер, который формирует сообщение WM\_TIMER.

Winhook – это механизм, позволяющий перехватывать и обрабатывать различные события в операционной системе Windows. Он может использоваться для мониторинга активности пользователя, анализа данных, защиты от вредоносного ПО и других целей.

1. **РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ**

В результате выполнения лабораторной работы было создано графическое приложение стрелочных часов (рисунок 1).

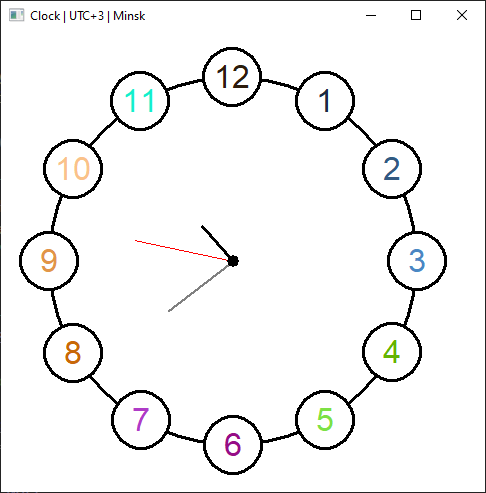


Рисунок 1 – Окно приложения

Для изменения часового пояса используется обработка нажатия клавиш *VK\_LEFT* и *VK\_RIGHT*. В случае, если часовой пояс выходит за указанный диапазон, то приложение выводит окно с ошибкой (рисунок 2).

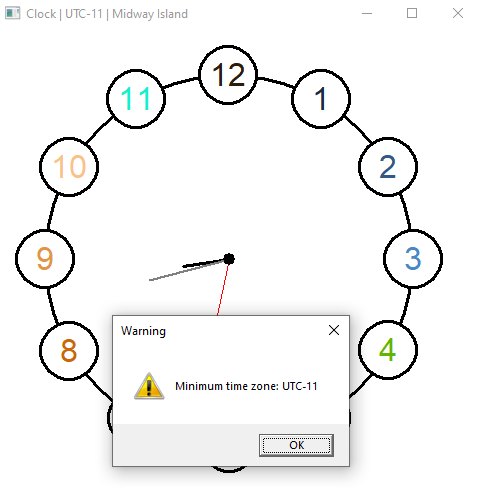


Рисунок 2– Выход за диапазон часового пояса

При нажатии пробела *VK\_SPACE* воспроизводится звук кукушки, нажатие обрабатывается с помощью хука.

# ЗАКЛЮЧЕНИЕ

В результате лабораторной работы были изучены основные принципы работы с GDI: работа с контекстом отображения, использование кистей, перьев и цветов, рисование геометрических фигур. Было создано приложение, отображающее реальное время на стрелочных часах с возможностью изменения часового пояса, обработкой выхода за диапазон допустимых значений часового пояса (UTC) и воспроизведению звука кукушки по нажатию пробела.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Основы программирования для Win32 API [Электронный ресурс]. – Режим доступа: <https://dims.karelia.ru/win32/>.
2. Графический интерфейс GDI в Microsoft Windows [Электронный ресурс]. – Режим доступа: https://www.frolov-lib.ru/books/bsp/v14/index.html.
3. Windows GDI [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/ru-ru/windows/win32/gdi/windows-gdi.

# ПРИЛОЖЕНИЕ А (обязательное) Листинг кода

Листинг 1 – Файл Source.cpp

#define \_USE\_MATH\_DEFINES

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <windows.h>

#include <string>

#include <string>

#include <ctime>

#include <map>

#include <thread>

#include "Constants.h"

using namespace std;

int utcOffset = 3;

HHOOK keyboardHook;

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

LRESULT CALLBACK MainWindowProcedure(HWND hWnd, UINT msg, WPARAM wParam, LPARAM lParam);

void PlaySoundAsync(wstring soundFile);

RECT reduceRect(RECT rect, const double alpha);

void DrawClock(HDC hdc, RECT rect, int hour, int minute, int second);

void DisplayTimeZone(HWND hWnd);

void UpdateClock(HWND hWnd);

int WINAPI WinMain(HINSTANCE hInst, HINSTANCE, LPSTR cmdline, int ss) {

keyboardHook = SetWindowsHookEx(WH\_KEYBOARD\_LL, KeyboardProc, hInst, 0);

// Создание класса главного окна

WNDCLASS wc;

wc.style = NULL;

wc.lpfnWndProc = MainWindowProcedure;

wc.cbClsExtra = NULL;

wc.cbWndExtra = NULL;

wc.hInstance = hInst;

wc.hIcon = NULL;

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

wc.hbrBackground = (HBRUSH)(COLOR\_WINDOW);

wc.lpszMenuName = NULL;

wc.lpszClassName = L"MainWindowClass";

if (!RegisterClass(&wc))

return EXIT\_FAILURE;

// Создание главного окна

HWND hWnd = CreateWindow(

L"MainWindowClass", L"Clock",

WS\_OVERLAPPEDWINDOW,

0, 0, 500, 500,

NULL, NULL, hInst, NULL

);

if (!hWnd)

return EXIT\_FAILURE;

ShowWindow(hWnd, ss);

SetTimer(hWnd, NULL, 1000, NULL);

MSG msg;

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

TranslateMessage(&msg);

DispatchMessage(&msg);

}

UnhookWindowsHookEx(keyboardHook);

return msg.wParam;

}

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

if (nCode >= 0) {

if (wParam == WM\_KEYDOWN) {

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

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

thread soundThread(PlaySoundAsync, L"audio/kukushka.wav");

soundThread.detach();

}

}

}

return CallNextHookEx(keyboardHook, nCode, wParam, lParam);

}

LRESULT CALLBACK MainWindowProcedure(HWND hWnd, UINT msg, WPARAM wParam, LPARAM lParam) {

switch (msg)

{

case WM\_CREATE:

{

thread soundThread(PlaySoundAsync, L"audio/start-windows.wav");

soundThread.detach();

break;

}

case WM\_SIZE:

{

UpdateClock(hWnd);

break;

}

case WM\_GETMINMAXINFO: {

MINMAXINFO\* minsize = (MINMAXINFO\*)lParam;

minsize->ptMinTrackSize.x = 400;

minsize->ptMinTrackSize.y = 400;

break;

}

case WM\_TIMER: {

UpdateClock(hWnd);

break;

}

case WM\_KEYDOWN:

switch (wParam) {

case VK\_LEFT:

if (utcOffset == -11) {

MessageBox(NULL, L"Minimum time zone: UTC-11", L"Warning", MB\_ICONWARNING | MB\_OK);

}

else {

utcOffset -= 1;

UpdateClock(hWnd);

}

break;

case VK\_RIGHT:

if (utcOffset == 12) {

MessageBox(NULL, L"Maximum time zone: UTC+12", L"Warning", MB\_ICONWARNING | MB\_OK);

}

else {

utcOffset += 1;

UpdateClock(hWnd);

}

break;

}

break;

case WM\_DESTROY:

{

PostQuitMessage(0);

break;

}

}

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

}

void PlaySoundAsync(wstring soundFile) {

PlaySound(soundFile.c\_str(), NULL, SND\_FILENAME);

}

// Уменьшить прямоугольик отрисовки

RECT reduceRect(RECT rect, const double alpha) {

RECT result;

// Исходные размеры

int width = rect.right - rect.left;

int height = rect.bottom - rect.top;

// Новые размеры

int newWidth = (int)(width \* alpha);

int newHeight = (int)(height \* alpha);

// Смещение по X и Y для центрирования уменьшенного прямоугольника

int xOffset = (width - newWidth) / 2;

int yOffset = (height - newHeight) / 2;

// Новые координаты всех границ прямоугольника

result.left = rect.left + xOffset;

result.top = rect.top + yOffset;

result.right = result.left + newWidth;

result.bottom = result.top + newHeight;

return result;

}

// Отрисовка часов

void DrawClock(HDC hdc, RECT rect, int hour, int minute, int second) {

// Вычисление в соответствии с размером окна

int width = rect.right - rect.left;

int height = rect.bottom - rect.top;

int diameter = min(width, height);

int radius = diameter / 2;

int centerX = rect.left + diameter / 2;

int centerY = rect.top + diameter / 2;

// Отрисовка большого круга

int penWidth = diameter / 100;

HPEN hPen = CreatePen(PS\_SOLID, penWidth, RGB(0, 0, 0));

HPEN hOldPen = (HPEN)SelectObject(hdc, hPen);

Ellipse(hdc, rect.left, rect.top, rect.left + diameter, rect.top + diameter);

// Установка размера шрифта в зависимости от диаметра

int fontSize = diameter / 10;

HFONT hFont = CreateFont(fontSize, 0, 0, 0, FW\_NORMAL, FALSE, FALSE, FALSE, DEFAULT\_CHARSET,

OUT\_OUTLINE\_PRECIS, CLIP\_DEFAULT\_PRECIS, CLEARTYPE\_QUALITY, DEFAULT\_PITCH, L"Arial");

HFONT hOldFont = (HFONT)SelectObject(hdc, hFont);

// Рисование цифр

for (int i = 1; i <= 12; i++) {

// Расчет координаты (x, y) для цифры

double angle = i \* (360.0 / 12.0);

double radian = angle \* M\_PI / 180.0;

int textX = centerX + (radius \* sin(radian));

int textY = centerY - (radius \* cos(radian));

// Создание кисти для заливки окружности

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

HBRUSH hOldBrush = (HBRUSH)SelectObject(hdc, hBrush);

// Отрисовка окружности для цифры

int radius = diameter \* 0.08;

Ellipse(hdc, textX - radius, textY - radius, textX + radius, textY + radius);

// Восстановка предыдущей кисти

SelectObject(hdc, hOldBrush);

DeleteObject(hBrush);

// Отрисовка цифры

SetBkMode(hdc, TRANSPARENT);

wstring digit = to\_wstring(i);

SIZE textSize;

GetTextExtentPoint32W(hdc, digit.c\_str(), digit.length(), &textSize);

SetTextColor(hdc, RGB((i \* 25) % 256, (i \* 45) % 256, (i \* 65) % 256));

TextOutW(hdc, textX - textSize.cx / 2, textY - textSize.cy / 2, digit.c\_str(), digit.length());

SetBkMode(hdc, OPAQUE);

}

// Восстановление предыдущего пера

SelectObject(hdc, hOldPen);

DeleteObject(hPen);

// Восстановление предыдущего шрифта

SelectObject(hdc, hOldFont);

DeleteObject(hFont);

// Отрисовка часовой стрелки

int hourAngle = (hour % 12) \* 30 + (minute / 2);

int hourLength = radius \* 0.25;

int hx = centerX + hourLength \* sin(hourAngle \* M\_PI / 180.0);

int hy = centerY - hourLength \* cos(hourAngle \* M\_PI / 180.0);

int hourPenWidth = diameter / 100;

HPEN hHourPen = CreatePen(PS\_SOLID, hourPenWidth, RGB(0, 0, 0));

hOldPen = (HPEN)SelectObject(hdc, hHourPen);

MoveToEx(hdc, centerX, centerY, NULL);

LineTo(hdc, hx, hy);

SelectObject(hdc, hOldPen);

DeleteObject(hHourPen);

// Отрисовка минутной стрелки

int minuteAngle = minute \* 6 + (second / 10);

int minuteLength = radius \* 0.45;

int mx = centerX + minuteLength \* sin(minuteAngle \* M\_PI / 180.0);

int my = centerY - minuteLength \* cos(minuteAngle \* M\_PI / 180.0);

int minutePenWidth = diameter / 150;

HPEN hMinutePen = CreatePen(PS\_SOLID, minutePenWidth, RGB(128, 128, 128));

hOldPen = (HPEN)SelectObject(hdc, hMinutePen);

MoveToEx(hdc, centerX, centerY, NULL);

LineTo(hdc, mx, my);

SelectObject(hdc, hOldPen);

DeleteObject(hMinutePen);

// Отрисовка секунднрй стрелки

int secondAngle = second \* 6;

int secondLength = radius \* 0.55;

int sx = centerX + secondLength \* sin(secondAngle \* M\_PI / 180.0);

int sy = centerY - secondLength \* cos(secondAngle \* M\_PI / 180.0);

int secondPenWidth = diameter / 200;

HPEN hSecondPen = CreatePen(PS\_SOLID, secondPenWidth, RGB(255, 0, 0));

hOldPen = (HPEN)SelectObject(hdc, hSecondPen);

MoveToEx(hdc, centerX, centerY, NULL);

LineTo(hdc, sx, sy);

SelectObject(hdc, hOldPen);

DeleteObject(hSecondPen);

// Отрисовка круга в центре часов

HBRUSH hBrush = CreateSolidBrush(RGB(0, 0, 0));

HBRUSH hOldBrush = (HBRUSH)SelectObject(hdc, hBrush);

int r = radius / 30;

Ellipse(hdc, centerX - r, centerY - r, centerX + r, centerY + r);

SelectObject(hdc, hOldBrush);

DeleteObject(hBrush);

}

void DisplayTimeZone(HWND hWnd) {

wstring new\_title = BASE\_TITLE;

if (utcOffset < 0)

new\_title += to\_wstring(utcOffset);

else if (utcOffset > 0)

new\_title += L"+" + to\_wstring(utcOffset);

new\_title += L" | " + TIME\_ZONE\_MAP[utcOffset];

SetWindowText(hWnd, new\_title.c\_str());

}

void UpdateClock(HWND hWnd) {

HDC hdc = GetDC(hWnd);

RECT clientRect;

GetClientRect(hWnd, &clientRect);

HBRUSH bgBrush = CreateSolidBrush(GetSysColor(COLOR\_WINDOW));

FillRect(hdc, &clientRect, bgBrush);

DeleteObject(bgBrush);

time\_t now = time(0);

tm\* localtm = localtime(&now);

tm\* gmtm = gmtime(&now);

DrawClock(hdc, reduceRect(clientRect, 0.8), gmtm->tm\_hour + utcOffset, gmtm->tm\_min, gmtm->tm\_sec);

DisplayTimeZone(hWnd);

ReleaseDC(hWnd, hdc);

}

Листинг 2 – Файл Constants.h

#pragma once

#include <string>

#include <map>

using namespace std;

wstring BASE\_TITLE = L"Clock | UTC";

map<int, wstring> TIME\_ZONE\_MAP = {

{-11, L"Midway Island"},

{-10, L"Honolulu"},

{-9, L"Anchorage"},

{-8, L"Los Angeles"},

{-7, L"Denver"},

{-6, L"Chicago"},

{-5, L"New York"},

{-4, L"Caracas"},

{-3, L"Buenos Aires"},

{-2, L"South Georgia"},

{-1, L"Azores"},

{0, L"London"},

{1, L"Paris"},

{2, L"Athens"},

{3, L"Minsk"},

{4, L"Kabul"},

{5, L"Islamabad"},

{6, L"Almaty"},

{7, L"Bangkok"},

{8, L"Beijing"},

{9, L"Tokyo"},

{10, L"Sydney"},

{11, L"Solomon Islands"},

{12, L"Suva"}

};